Maastricht University

Faculty of Science and Engineering

Department of Advanced Computing Sciences (DACS)

Contents

Academic Calendar 2024-2025

Bachelor's Programme Data Science and Artificial Intelligence (year 1, 2 and 3)
Bachelor's Programme Computer Science (year 1 and 2) and
Master's Programmes Artificial Intelligence and Data Science for Decision Making (year 1 and 2)

🟦

Inkom Maastricht University
19-22 August 2024

🟧

Education periods
Period 1: 2 September - 11 October 2024
Period 2: 28 October - 6 December 2024
Period 3: project weeks, see below
Period 4: 27 January - 14 March 2025
Period 5: 31 March - 16 May 2025
Period 6: project weeks, see below

🟨

Preparation weeks
assignments are possible, but there will be no scheduled activities on site.
Period 1: 21 October – 25 October 2024
Period 4: 24 March – 28 March 2025

🟩

Project weeks
Period 3: 6 January - 17 January 2025 (BY1)
Period 3: 6 January - 24 January 2025 (all other)
BY1: Final presentation: January 2025
BY2: Final presentation: January 2025
BY3: Final presentation: January 2025
MY1: Project seminar: January 2025
Period 6: 26 May - 13 June 2025
BY1: Final presentation: June 2025
BY2: Final presentation: June 2025
MY1: Project seminar: June 2025

Bachelor Thesis Winter Conference:
2-6 December 2024

Resit Bachelor Thesis Winter Conference:
20-24 January 2025

Bachelor Thesis Summer Conference:
10-13 June 2025

Resit Bachelor Thesis Summer Conference:
25-29 August 2025

🟩

Introduction days

27 August 2024	(Bachelor DSAI)
28 August 2024	(Bachelor CS)
29 August 2024	(Masters, premasters, exchange)
29 August 2024	(BBQ event for all new students)
30 August 2024	(Mentor event for all new bachelor students)
24 January 2025	(February intake)

🟥

Exam periods

14 - 18 October 2024:	Exams period 1 -> all
9 - 13 December 2024:	Exams period 2 -> all
17 - 21 March 2025:	Exams period 4 -> all
19 - 23 May 2025:	Exams period 5 -> all

Resits period

9 - 13 December 2024:	Resit period 1 -> BY2, 3 and Masters
20 - 24 January 2025:	Resit period 1 and 2 -> BY1
17 - 21 March 2025:	Resit period 2 -> BY2, 3 and Masters
19 - 23 May 2025:	Resit period 4 -> Masters
16 - 20 June 2025:	Resit period 4 and 5 -> BY1, 2, 3 and Masters

🟦

Graduation
26 September 2024 (Masters)
End November 2025 (Bachelor and Master)

🟪

(Public) Holiday, no courses
Christmas: 16 December 2024 - 3 January 2025
Carnival break: 3 - 7 March 2025
Good Friday: 18 April 2025
Easter Monday: 21 April 2025
Liberation Day: 5 May 2025
Ascension Day and Bridging Day: 29 - 30 May 2025
Whit Monday: 9 June 2025

DACS bachelor students will have to register for their own courses and resits through the Student Portal. DACS master students will have to register their courses through a form which will be sent by Student Affairs.

August

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

September

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

October

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

November

M	T	W	T	F	S	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

December

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31					

January

M	T	W	T	F	S	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

February

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28		

March

M	T	W	T	F	S	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31						

April

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30				

May

M	T	W	T	F	S	S
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

June

M	T	W	T	F	S	S
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

July

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

1 Education: the profile of the study Data Science & Artificial Intelligence, Computer Science, Artificial Intelligence, and Data Science for Decision Making

1.1 What are the programmes about?

The Bachelor's Programmes Computer Science and Data Science & Artificial Intelligence, and the Master's Programmes Artificial Intelligence and Data Science for Decision Making of the Faculty of Science and Engineering at Maastricht University are embedded within the Department of Advanced Computing Sciences.

At the Department of Advanced Computing Sciences (DACS), research and education are tightly linked. Our academic staff consists of experienced lecturers and researchers that are well known in the international scientific community. You will be taught by lecturers from DACS, and you will become part of a tight-knit community consisting of approximately 1200 bachelor's and master's students and 100 staff members. Together, we come from over 50 different countries. The department has its own dedicated study association, MSV Incognito, of which all students automatically become a member.

The department's research activities span the disciplines and interfaces of artificial intelligence, data science, computer science, applied mathematics, and robotics, covering the entire spectrum from curiosity-driven research to responsible societal applications, often taking on an interdisciplinary character. Contributions to areas such as multi-agent systems, (medical) signal and image processing, machine learning, explainable AI, FAIR principles, game theory, intelligent search techniques, computer vision, cyber-security, and affective computing are internationally recognized. You will be exposed to the department's research through several semester projects, and your thesis project, among all.

1.2 Study System

1.2.1 Bachelor Data Science and Artificial Intelligence

The bachelor's programme in Data Science & Artificial Intelligence is a three-year programme. We chose for a broad setup of the curriculum, so that students can decide on the way they would like to specialize during the final stage.

Year 1

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Procedural Programming;	Objects in Programming;	PROJECT	Data Structures and Algorithms;	Computational and Cognitive Neuroscience;	PROJECT
Discrete Mathematics;	Calculus;		Linear Algebra;	Numerical Methods;	
Introduction to Data Science and Artificial Intelligence	Logic		Principles of Data Science	Software Engineering	
PROJECT			PROJECT		

Year 2

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Databases;	Machine Learning;	PROJECT	Human Computer Interaction & Affective Computing;	Philosophy and Artificial Intelligence;	PROJECT
Probability and Statistics;	Simulation and Statistical Analysis;		Mathematical Modelling;	Linear Programming;	
Graph Theory	Reasoning Techniques		1 out of the electives: * Theoretical Computer Science * Multi-variable Calculus	Natural Language Processing	
PROJECT			PROJECT		

Year 3

Period 1*	Period 2*	Period 3	Period 4	Period 5	Period 6
Digital Society;	Computer Security;	PROJECT	Operations Research Case Studies **	BACHELOR'S THESIS	
Game Theory;	Software and Systems Verification;		Intelligent Systems;		
Semantic Web;	Logic for Artificial Intelligence;		Data Analysis		
Recommender Systems;	Parallel Programming;				
Robotics and Embedded Systems;	Large Scale IT and Cloud Computing;				
Introduction to Quantum Computing	Introduction to Bio-Informatics				
PROJECT			BACHELOR'S THESIS		

** Third year students choose 6 optional courses (3 in period 3.1 and 3 in period 3.2) in addition to the semester project in semester 1 of year 3. In case students have passed both electives of period 2.4, either the course Theoretical Computer Science or Multivariable Calculus can replace 1 of the third year electives. In semester 1 of year 3, student can also choose (1) elective courses at other UM bachelor programmes of at most 18 ECTS (2) the minor Entrepreneurship or (3) the educational minor. (4) Alternatively, students can study abroad for a semester at one of our exchange partners (see internationalization section below). Please contact the student adviser for more information. Also, check the Study Abroad section in the "My Organisations" section of Canvas.*

***) The course has capacity of 80 participants (students).*

Periods 1, 2, 4 and 5 last seven weeks in total. During week 1-6 there are classes and in week 7 exams. Three courses are offered during each period, each course is good for 4 credits (ECTS). Per course, five to seven hours of class are offered each week in year 1, and about six hours in year 2 and 3. At the end of periods 1 and 4 there is one week planned for study and/or preparation for upcoming periods.

Next to these courses, you participate in a group project of 6 credits that will last the whole semester. Skill classes and project meetings are mandatory: you are expected to be present during 100% of the skill classes and 100% of the project meetings in each academic year. For more details about attendance of skill classes and project meetings, please check the Education and Examination Regulations (EER) and the Rules and Regulations (R&R) that are published on the Student Portal.

If a student does not contribute sufficiently to the project, the project examiners may deviate from the group grade for this individual student. The project of semester 1 runs during period 1, 2 and 3. The project of semester 2 runs during period 4, 5 and 6. For specific details on the project curriculum in year 1 see section Project 1-1 and Project 1-2 with the course descriptions of year 1.

Periods 3 and 6 last three weeks (except for period 3 of year 1, which lasts two weeks), and during these three weeks, students work full time to finish their project assignment. During the three-week project periods in periods 3 and 6, you will work full-time on a project assignment. This project assignment is announced in the beginning of periods 1 or 4, along with the group composition. At the end of each period, groups are assessed on the progress or final results of their work. The form of assessment is specific to each project. The examiners give their feedback to the content and progress of a project. The assessment in principle results in the same mark for all group members. However, there can be diversification, see the Education and Examination Regulations (EER) and the Rules and Regulations (R&R).

The final stage of your bachelor's programme, period 5 and 6 of year 3 is reserved for writing your bachelor's thesis that equals 18 ECTS. Every student has to conduct a short scientific research focussed on a relevant topic. This can be empirical or theoretical research. Students have acquired information on these different research domains throughout their educational programme. Each student has to hand in a signed bachelor thesis project plan to the Bachelor's thesis coordinator. Each student is supervised by a thesis supervisor. In the second period of this semester, the students conduct their own research. In the end of the last period of the semester, each student must present their results.

1.2.2 Bachelor Computer Science

The bachelor's programme in Computer Science is a three-year programme. The programme is designed to provide a solid background in fundamental computer science, software development, and mathematics.

Year 1

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Procedural Programming;	Objects in Programming;	PROJECT	Linear Algebra;	Databases;	PROJECT
Discrete Mathematics;	Calculus;		Data Structures and Algorithms;	Statistics;	
Introduction to Computer Science	Logic		Object-Oriented Modelling	Algorithmic Design	
PROJECT			PROJECT		

Year 2

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Computer Networks;	Software Engineering & Architectures;	PROJECT	Embedded Programming;	IT Management & Privacy	PROJECT
Introduction to Artificial Intelligence;	Principles of Programming Languages		Computer Security;	Numerical Methods	
Intelligent User Interfaces	1 out of the two elective modules:		Parallel Programming	1 out of the two elective modules	
	- **Intelligent Interaction:** course Image & Video Processing + Project 2-1: Human-Computer Interaction - **Artificial Intelligence & Machine Learning:** course Machine Learning + Project 2-1: Adaptive Systems			- **High-Performance Computing:** course High-Performance Computing + Project 2-2: High-Performance Computing - **Cyber-security and IoT:** choose one course between Information Security or Ubiquitous Computing & IoT + Project 2-2: Cybersecurity & IoT	
	PROJECT		PROJECT		

Periods 1, 2, 4 and 5 last seven weeks in total. During week 1-6 there are classes and in week 7 exams. Three courses are offered during each period, each course is good for 4 credits (ECTS). Per course, five to seven hours of class are offered each week in year 1, and about six hours in year 2 and 3. At the end of periods 1 and 4 there is one week planned for study and/or preparation for upcoming periods.

Next to these courses, you participate in a group project of 6 credits that will last the whole semester. Skill classes and project meetings are mandatory: you are expected to be present during 100% of the skill classes and 100% of the project meetings in each academic year. For more details about attendance of skill classes and project meetings, please check the Education and Examination Regulations (EER) and the Rules and Regulations (R&R) that are published on the Student Portal.

If a student does not contribute sufficiently to the project, the project examiners may deviate from the group grade for this individual student. The project of semester 1 runs during period 1, 2 and 3. The project of semester 2 runs during period 4, 5 and 6. For specific details on the project curriculum in year 1 see section Project 1-1 and Project 1-2 with the course descriptions of year 1.

Periods 3 and 6 last three weeks (except for period 3 of year 1, which lasts two weeks), and during these three weeks, students work full time to finish their project assignment. During the three-week project periods in periods 3 and 6, you will work full-time on a project assignment. This project assignment is announced in the beginning of periods 1 or 4, along with the group composition. At the end of each period, groups are assessed on the progress or final results of their work. The form of assessment is specific to each project. The examiners give their feedback to the content and progress of a project. The assessment in principle results in the same mark for all group members. However, there can be diversification, see the Education and Examination Regulations (EER) and the Rules and Regulations (R&R).

The final stage of your bachelor's programme, period 5 and 6 of year 3 is reserved for writing your bachelor's thesis that equals 18 ECTS. Every student has to conduct a short scientific research focused on a relevant topic. This can be empirical or theoretical research. Students have acquired information on these different research domains throughout their educational programme. Each student has to hand in a signed bachelor thesis project plan to the Bachelor's thesis coordinator. Each student is supervised by a thesis supervisor. In the second period of this semester, the students conduct their own research. In the end of the last period of the semester, each student must present their results.

1.2.3 Masters

AI year 1:

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Intelligent Search and Games;	**Advanced Concepts in Machine Learning;**	R E S E A R C H P R O J E C T	**Agents & Multi-Agent Systems;**	**Autonomous Robotic Systems;**	R E S E A R C H P R O J E C T
1 of the electives:	*1 of the electives:*		*1 of the electives:*	*1 of the electives:*	
Stochastic Decision Making	Network Science		Building and Mining Knowl-edge Graphs	Information Retrieval and Text Mining	
Data Mining	Advanced Natural Language Processing		Planning and Scheduling	Computer Vision	
Signal and Image Processing			Dynamic Game Theory	Reinforcement Learning	
			Explainable AI (*)	Introduction to Quantum Computing for AI and Data Science (**)	
PROJECT			PROJECT		

AI year 2:

Semester 1	Semester 2

Elective Semester:
 	
Courses and Research Project;
 	
Research Internship;
 	
Professional Internship;
 	
Quantum Computing specialization (pre-requisite: Intro QC) (**);
 	
Study Abroad
	THESIS

(*) The course has capacity of 60 participants (students).

(**) The course Introduction to Quantum Computing for AI and Data Science is a prerequisite for the elective courses Quantum Algorithms, Quantum AI and Quantum Information and Security. These four courses, together with a dedicated research project on quantum computing, form the specialization in Quantum Computing for AI and Data Sciences.

Master Artificial Intelligence

DSDM year 1:

Period 1	Period 2	Period 3	Period 4	Period 5	Period 6
Data Mining	**Model Identification and Data Fitting**	R E S E A R C H P R O J E C T	**Computational Statistics**	**Algorithms for Big Data**	R E S E A R C H P R O J E C T
1 of the electives:	*1 of the electives:*		*1 of the electives:*	*1 of the electives:*	
Signal and Image Processing	Advanced Concepts in Machine Learning		Dynamic Game Theory	Symbolic Computation and Control	
Mathe-matical Optimiza-tion	Network Science		Planning and Scheduling	Information Retrieval and Text Mining	
Stochastic Decision Making	Advanced Natural Language Processing		Building and Mining Knowledge Graphs	Computer Vision	
			Data Fusion	Introduction to Quantum Computing for AI and Data Science (**)	
			Explainable AI (*)		
PROJECT			PROJECT		

DSDM year 2:

Semester 1	Semester 2

Elective Semester:
 	
Courses and Research Project;
 	
Research Internship;
 	
Professional Internship;
 	
Quantum Computing specialization (pre-requisite: Intro QC) (**);
 	
Study Abroad
	THESIS

(*) The course has capacity of 60 participants (students)

(**) The course Introduction to Quantum Computing for AI and Data Science is a prerequisite for the elective courses Quantum Algorithms, Quantum AI and Quantum Information and Security. These four courses, together with a dedicated research project on quantum computing, form the specialization in Quantum Computing for AI and Data Sciences.

Periods 1, 2, 4 and 5 last seven weeks in total. During weeks 1-6 there are classes and in week 7 exams. Several courses are offered during each period, each course equals 6 credits (ECTS). Per course, six hours of classes/tutorials are offered, on average, per week, in which a teacher will explain the theory of the subject or in which you do some practical training. At the end of periods 1 and 4 there is one week dedicated entirely for project work.

Depending on the master´s programme you enrolled in, you are required to take and pass several mandatory courses (4 for the AI master, and 4 for the DSDM master). These courses are underlined in the tables above. Besides these courses, there is a selection of electives in year 1. You are required to take 1 elective in each period in year 1. There are possibilities to take other Master courses taught at the Department of Advanced Computing Sciences as electives in year 2.

Next to the courses, in year 1 of your studies you participate in a group project of 6 credits every semester, that will last the whole semester.
As a student, you are expected to participate actively in doing tasks with respect to the project skills training and project meetings. In addition, students are expected to cooperate actively with their group in order to successfully finish their project assignment. If a student fails to do so, the student might not be allowed to participate in the examination of the project, or the project examiners may deviate from the group grade for this individual student (see the Education and Examination Regulations (EER) and the Rules and Regulations (R&R) for details). The project of semester 1 runs during periods 1, 2, and 3. The project of semester 2 runs during periods 4, 5, and 6. Periods 3 and 6 last three weeks, and students work fulltime to finish their project assignment. The topics for the projects are announced at the beginning of each semester. Students cast their preferences for the projects. The project assignment and the group composition will be announced shortly afterwards, based on student preference in so far as possible. During the semester, each group will provide a research and business plan/report (end of period 1 or period 4), give an interim presentation on the progress, their approach and schedule for the remaining time (during period 2 or period 5), a social-media post (end of period 2 or period 5), and a final product together with report and public oral presentation. Each of these milestones is assessed by the supervisor/examiner. The assessment will - in principle - yield the same mark for all the group members. However, there can be diversification, see the Education and Examination Regulations (EER) and the Rules and Regulations (R&R).

Choose your own curriculum

During the first semester of the second year of the master’s programme, you can choose your own curriculum from available options, enabling you to pursue your personal interests. During this semester, you can obtain 30 credits by choosing elective courses of the other master’s programmes offered at the Department of Advanced Computing Sciences (i.e., AI or DSDM). In addition, you can also:

- Take a combination of elective courses from the master’s AI or DSDM and courses at another master’s programme of Maastricht University;
- Participate in a research project (a research internship) of the academic staff of the Department of Advanced Computing Sciences or at another research university;
- Participate in an internship at a company;
- Follow an exchange programme at one of our partner universities abroad.

The final stage of your master’s programme, during the second semester of the second year, is reserved for conducting and writing your master’s thesis that counts for 30 credits. The thesis is produced individually and is the result of a master research project on a topic that you will be working on under the supervision of one of the academic staff members of the programme. In the preliminary phase, the emphasis is on self-study, subject determination, approaching a supervisor, planning, and some preliminary research. After approval of the thesis research plan by the Board of Examiners, the actual research starts. In this phase, the student carries out his/her own research.
The senior researcher that acts as the supervisor of this research process will guide the student during a series of regular appointments. The final phase is used to accomplish, i.e. write, the master’s thesis. The master’s thesis project is completed by an individual presentation and discussion of the results at the department. Assessment will be based on the research, the thesis itself, the process, the software and the presentation and discussion of this thesis (i.e., public defence).

Note that all individual curriculum choices are guided by our student counsellors and are subject to approval by the Board of Examiners.

1.2.4 Project Centred Learning

All of our bachelor's and master's programmes employ project centered learning (PCL), a variant of Maastricht University's signature problem-based learning. The PCL educational model is small-scale and student-oriented. You work in small groups on two complex and challenging projects per year (amounting to one per semester). These projects last around five months each and run parallel to courses, which follow a lecture- and tutorial-based setup. The level of the projects, and the products students need to deliver, matches the students' study progress by requiring knowledge obtained from coursework. In addition to technical skill development, this educational model trains employability skills such as teamwork, project planning, documenting, and presenting.

We base projects on real-life research and/or societal challenges provided by our staff and by companies. This gives our students the opportunity to gain invaluable experience by applying the learned knowledge to finding solutions to real-world problems. Together with fellow students, you research on what existing approaches can be adapted, and design and evaluate new approaches for the problem at hand. At the end of each project, you deliver a functional product and present your findings to your fellow students, the teachers and/or the client.

Project Centered Learning has advantages:

- from the beginning you find out what teamwork means
- you learn project-related skills in a natural way
- you will be continuously placed in an active role
- you will be able to match theory with its applications
- PCL increases the student's motivation

Projects for instance require students to design a quadcopter platform for an autonomous swarm, to detect fraud behavior in bank transactions, to optimize timetables, to model human decision process from intercranial EEG, to automate the recognition of facial expressions, or to design and implement a traffic simulator.

1.2.5 Shortened Academic Year pilot at FSE

Starting September 1st 2024, the Faculty of Science and Engineering (FSE) will participate in the UM pilot for the Shortened Academic Year. This pilot shortens the Academic Year by about two weeks in the summer and one week before Christmas as compared to the standard UM frame schedule. In this way, the length of our academic year becomes more closely aligned with international practice. This gives students dedicated time to further explore own research, work on projects, or use some extra rest at the end of each semester.

The pilot is implemented by all FSE bachelor and masters programmes and remains based on the UM's 6-period structure. The biggest change in the pilot is a longer three-week Christmas break, and the summer break that starts two to three weeks earlier. The Business Engineering programme will not join the pilot at this moment due to the entwinement with the School of Business and Economics. The pilot will run for at least the next three academic years. At the end of the first year of the pilot in September 2025, a first survey will be conducted to evaluate the outcomes of the shortened academic year.

1.3 Study Abroad

In the two bachelors Data Science and Artificial Intelligence and Computer Science and the two masters Data Science for Decision Making and Artificial Intelligence we have one of the highest ratios of international students. More than 75% of the scientific staff and 75% of the students are non-Dutch, giving rise to an international study environment. Additionally, we host a number of international exchange students each year and offer our own students a number of opportunities for international experiences themselves. For example, we offer students the opportunity to study abroad for a semester during the elective semester. For this purpose, we collaborate with well-established partner universities.

For bachelor students there is an option to participate in an international study abroad programme during the fifth (elective) semester of the bachelor's programme. For master students there is an option to participate in an international study abroad during the third (elective) semester of the master's programmes.

Please note that for students entering the Master programmes during the February intake, fewer universities may be available for study abroad. More documentation and information about the participating exchange partner universities is available on Canvas under the "Going Abroad" section (see the application form). Also, here you can find the Study Abroad Guide.
For additional information, you can contact the international relations officer or your study advisor.

1.4 Degree

A successful conclusion of the bachelor's programme will provide you with a bachelor's certificate according to Dutch law, that is, a 'Bachelor of Science'. A successful conclusion of a master's programme will provide you with a master's certificate according to Dutch law, that is, a 'Master of Science'.

1.5 Study Feasibility and Quality Assurance

We strive for continuous improvement of the quality and feasibility of our study programmes. Student evaluations of each course help us in maintaining a high standard of educational quality and keeping the study programmes feasible. The study programme's feasibility means that a student with an appropriate background should be able to finish the study within the set number of years.

To maintain this standard, the quality assurance officer collects information about teaching, learning and assessment at the end of each period. The quality assurance officer then reports the outcomes of the student evaluations to the Education Programme Committee (EPC) of the respective bachelor's or master's programme. The EPC consists of representatives of staff and students (6 staff representatives and five student representatives -- two students of Bachelor DSAI, one student of Bachelor CS, and two master students, one for each Master's programme). If the outcomes are unsatisfactory, the EPC will take action to improve the quality of a specific course (or project) or of the study as a whole. Therefore, student feedback for courses is essential in pointing out strong aspects and aspects for improvement of all educational activities.

As a student in one of our bachelor's or master's programmes, you are encouraged to give your opinion about each course, as it may help improve that course. Moreover, future students may benefit from the results and comments of your evaluation just as you may benefit from course evaluations of fellow students. Also, if you are interested, you are encouraged to become a student member of the EPC.

1.6 Courses at other Faculties or universities

If a student from one of our bachelor's or master's programmes would like to participate in courses at other programmes of Maastricht University or other universities, approval from the Board of Examiners is needed in advance. Students should consider for themselves what the implications are of the mixed schedules at FSE and other faculties as a result of the Shortened Academic Year pilot at FSE. The student counsellors will upload further information on Canvas in November 2024.
A special position among electives from outside the bachelor's programme is the Educational Minor. The Educational Minor leads to a limited second-degree teaching qualification. Upon successful completion of the minor, you are qualified to teach in the first, second and third year of VWO, HAVO and VMBO-tl (MAVO) level in the Netherlands. Students in the BSc Data Science and Artificial Intelligence can - upon successful completion - acquire a teaching qualification for the main subject of Mathematics. The Educational Minor is organised in close cooperation with the Fontys Leraren Opleiding (Teacher Training) in Sittard (FLOS) and Tilburg (FLOT). The language of the Educational Minor is Dutch.

The programme contains several pedagogical-didactic courses in semester 5, along with education aimed at teaching methodology. There is also a mandatory practical internship, in the form of work placements, which is spread out over semesters 5 and 6. The education meetings mostly take place at UM and occasionally at the Fontys Leraren Opleiding in Sittard. The practical internship will be done at several secondary schools in the whole province of Limburg and will continue until the end of semester 6. During the practical internship, the student spends one day a week at a secondary school for the course of a full school year. In this way, the necessary teaching experience is obtained. Knowledge and practice are closely connected in the Educational Minor.

Successful completion of the educational minor yields 35 ECTS of which five are extracurricular. This means that these 5 ECTS cannot be used to replace any other components of the original bachelor program. Students in the BSc Data Science and Artificial Intelligence who wish to participate in the program should have accumulated, by the end of their second year, all 60 ECTS from the first-year components and at least 52 ECTS from the second year components. Prior to their enrolment in this minor, a motivated request for participation has to be submitted in Dutch to the Board of Examiners dacs-boe). Enrolment is dependent on selection and prior permission of the Board of Examiners.

If you have any questions regarding the contents of this educational minor, then please contact Dr. Stefan Maubach (s.maubach@maastrichtuniversity.nl).

1.7 Department of Advanced Computing Sciences Honours Programmes Bachelor

The Department of Advanced Computing Sciences offers its talented and top-performing bachelor's students the possibility to participate in our Honours Programme. This programme offers of two different tracks: a research track (MaRBLe 2.0) and a practical track (CS@Work or KE@Work, depending on the bachelor you are enrolled in, CS or DSAI respectively). If you successfully complete an honours programme track, this will be certified on an honour's diploma supplement.

MaRBle 2.0 – Research Track

In MaRBle 2.0, you get the opportunity to work on a state-of-the-art research project. Work will be organized in a similar way as in professional research institutes where participants work together as individual experts on a team project. Each student specializes in a task, gets assigned a team role and receives a self-contained research project. Together they work towards a common demonstration where individual research projects integrate towards a journal publication. For the individual subprojects, each student receives an individual supervisor from DACS.

For more information on MaRBLe 2.0, please contact the MaRBLe 2.0 coordinator, Rachel Cavil, via rachel.cavill@maastrichtuniversity.nl.

KE@Work – Practical Track

Students admitted to the KE@Work track (Knowledge Engineering at Work) are placed at a company or organisation through a careful selection and matching process. Over the full second and third year of the bachelor program, KE@Work students spend 50% of the time in class and 50% at the company, where they work on solving real world challenges and complex business problems using an academic approach, under supervision of dedicated business and university supervisors. For more information, please contact the coordinator via kework@maastrichtuniversity.nl

CS@Work – Practical Track

Students admitted to the CS@Work track (Computer Science at Work) are placed at a company or organisation through a careful selection process based on the elective choices made by the student during the second year. Over the span of the third year of the bachelor program, starting with three weeks of the summer break between the second and third year, CS@Work students spend 40% of the time in class and 60% at the company, where they work as part of a development team, solving real world challenges challenges and complex business problems using an academic approach, under supervision of dedicated business and university supervisors. For more information, please contact the coordinator via cswork@maastrichtuniversity.nl.

Eligibility Requirements

Eligibility requirements for the honours programmes entail that students:

- *Have passed all courses/components of their Bachelor's programme at first opportunity;*
- *Are maintaining a GPA of at least 7.5 up to the point of selection;*
- *Have not been convicted of fraud and have not been reprimanded for a violation of house rules or code of conduct.*

You can find the exact criteria and leniency in the Rules and Regulations.

1.8 Regulations

There are a number of Rules and Regulations that we expect you to be familiar with. An overview of the regulations can be found here

For examination, The Education and Examination Regulations (EER), the Rules and Regulations (RR) and the Rules of Procedure for Examination are of particular interest:

https://intranet.maastrichtuniversity.nl/en/dacs-students/exams-rules-and-regulations/examination

2 Bachelors

2.1 Curriculum of the first year of the Bachelor Programme Data Science and Artificial Intelligence

In order to learn how the transformation of raw data into useful information and knowledge is achieved, and how this transformation can be automated with the help of artificial intelligence, a thorough basic knowledge of specific mathematics and computer science subjects is required. This means that the first year is largely filled with mathematics and computer science subjects. Additionally, you will also follow courses on topics that help the understanding and broadening of the fields of Data Science and Artificial Intelligence, such as Computational & Cognitive Neuroscience.

The year is divided into four periods of seven weeks with three courses each, and two periods of three weeks during which you will work on a project. Each project is preceded by partial project assignments during the other periods. The week schedule works with two-hour clusters. In the overview below, the courses are indicated, as well as the study load in credits (ECTS). One ECTS credit stands for about 28 hours of study time (lectures, meetings and self-study). Besides the lectures that are given on the subjects, there will also be practical and skills training.

Year 1		ECTS
Period 1.1	Introduction to Data Science and Artificial Intelligence (KEN1110) Procedural Programming (KEN1120) Discrete Mathematics (KEN1130) Project 1-1 (KEN1300)	4 4 4
Period 1.2	Objects in Programming (KEN1220) Calculus (KEN1440) Logic (KEN 1530) Project 1-1 (KEN1300)	4 4 4
Period 1.3	Project 1-1 (KEN1300)	6
Period 1.4	Linear Algebra (KEN1410) Data Structures and Algorithms (KEN1420) Principles of Data Science (KEN1435) Project 1-2 (KEN1600)	4 4 4
Period 1.5	Computational and Cognitive Neuroscience (KEN1210) Software Engineering (KEN1520) Numerical Methods (KEN1540) Project 1-2 (KEN1600)	4 4 4
Period 1.6	Project 1-2 (KEN1600)	6

(*) Project 1-1 will start in period 1.1 and will run until period 1.3; Project 1-2 will start in period 1.4 and will run until period 1.6. The credits for the projects will become available at the end of period 1.3 and period 1.6, respectively. Please see the course description section Project 1-1 and Project 1-2 for more details on the project curriculum. For each period, we will give a short explanation of the various parts. Before the start of each period, the students will receive detailed information about the content, the study material, the teaching form, the schedule, and the examination method.

Period 1.1

Introduction to Data Science and Artificial Intelligence (KEN1110)

Examiner: Dr. Rachel Cavill, Dr. Pietro Bonizzi, and Prof. dr. Anna Wilbik

Desired Prior Knowledge: The course appears as desired prior knowledge for the courses Reasoning Techniques and Theoretical Computer Science.

Prerequisites: None.

Course description: The course Introduction to Data Science and Artificial Intelligence offers a comprehensive overview of the core topics in Data Science and Artificial Intelligence, both from a mathematical and from a computational perspective. Particular emphasis is put on the basic classes of techniques and methods, the theoretical underpinnings of data science and computational intelligence, and some example application domains of data science and artificial intelligence.
As such, the course provides an overview of many topics that are addressed in much more detail throughout the Bachelor's Data Science and Artificial Intelligence programme.

Knowledge and understanding: After successful completion of the course, students will be able to recognise what real world problems require the use of data science, and approach their solution by using a data science process, namely: explore the data, model the data, and perform simulations if required. Moreover, they will exhibit knowledge in the basic concepts of artificial intelligence, such as agents, search, artificial intelligence, decision trees.

Applying knowledge and understanding: Students learn to recognise applications of data science and artificial intelligence in different domains and apply the basic techniques they have learnt from both.

Making judgements: Upon completion of the course, students are able to recognise the relevant domains of data science and artificial intelligence when confronted with data science and artificial intelligence problems.

Communication: Students are able to explain the process they used to generate results and communicate the meaning of those results in context.

Learning skills: Students will be able to recognise small-scale data science problems and autonomously and critically reflect upon the appropriateness of data science and artificial intelligence methods for tackling those, and propose a primary solution.

Study material: Material will be provided during the course.

Recommended literature:
- S. Russell and P. Norvig (2010): Artificial Intelligence, A Modern Approach. Third edition, Pearson Education, ISBN 978-0-13-207148-2.
- C.D. Manning, P. Raghavan and H. Schütze (2008) Introduction to Information Retrieval. Cambridge University Press. ISBN 0521865719

Exam: There will be a closed book written exam at the end of the course.

ECTS: 4

Procedural Programming (KEN1120)

Coordinator: Dr. Enrique Hortal

Examiners: Dr. Enrique Hortal & Charis Kouzinopoulis

Desired prior knowledge: None.

Prerequisites: None.

Description: The course provides the basics of computer science and computer programming. After a short introduction to computer organization, the principles of programming are presented. The main topics of the course are: data types, variables, methods, parameters, decision structures, iteration, arrays, recursion and a branching application (related to the semester project). Programming skills will be acquired during practical sessions using the object-oriented programming language Java.

Knowledge and understanding: The course offers preliminary methodological and theoretical bases for studying and applying computers and computer programming on which the rest of the curriculum builds.

Applying knowledge and understanding: Whenever a computer system or a programming system has to be designed and implemented the knowledge and insights acquired during the course can be used and applied.

Making judgements: After successful completion of the course, students will be able to judge the quality and correctness of simple non-object-oriented programs.

Communication: The skills acquired during the course will enable students to communicate about standard programming constructs and algorithmic basics.

Learning skills: After successful completion of the course, students will be able to formalize, analyse and program solutions to simple software problems.

Study material: Lecture slides, example code and multimedia material that are made available before and after each lecture.

Assessment: Closed-book written exam (90%) + Assignments (10%)

Recommended literature: H. Schildt, Java: A Beginner's Guide, Eighth Edition, ISBN: 1260440214, McGraw-Hill Education

Additional literature: C. Horstmann (2016). Java Concepts (8th Edition). John Wiley & Sons, New York, ISBN: 978-1-1190-5645-4 or C.Horstmann (2012). Big Java Late Objects. John Wiley & Sons, New York, ISBN 978-1-1180-8788-6

ECTS: 4

Discrete Mathematics (KEN1130)

Examiner: Dr. Marieke Musegaas, Dr. Otti D'Huys, and Dr. Stefan Maubach

Desired Prior Knowledge: None.

Prerequisites: None.

Description: In this course, we build a mathematical framework that is based on logic and reason. The main objective of the course is to make students familiar with the language of mathematics. Students will learn how to make sound arguments and to detect where and why certain arguments go wrong. For this purpose, we will discuss the basic principles of logic and, closely related, the basic types of mathematical proofs. In doing so, we will encounter numbers such as integers, natural numbers and real numbers and we shall examine what makes these numbers special. After that, we will use basic logic to discuss, among other things, the following mathematical concepts: infinity, sets, relations, functions, permutations and combinations. Our fundamental tool in all of this is plain common sense. You really do not need your toolbox of mathematical formulas learned in previous studies and neither do you need a calculator. Pen and paper are the basic instruments needed. After completing each topic, exercises will be provided to be completed in class or at home, since mathematics is mainly learned by practising repeatedly.

Knowledge and understanding: Students will be able to read, interpret and manipulate basic mathematical terminology (propositional logic, quantifiers, set theory, relations, functions, and combinatorics). Students will also be able to read and interpret several different types of mathematical proofs and identify whether a purported proof is mathematically sound.

Applying knowledge and understanding: Upon completion of the course students will know how to read, interpret, write and manipulate rigorous mathematical statements using propositional logic, quantifiers, set theory, relations, functions and combinatorics. Students will be able to select, from a range of mathematical tools, which is appropriate to prove or disprove a given mathematical statement, and apply the chosen tools, rigorously and clearly in order to achieve the desired goal.

Making judgements: Students will be able to distinguish between mathematically sound and unsound statements and defend the rigour of their own mathematical arguments.

Communication: Students will be able to write clear, rigorous and explicit mathematical arguments using standardized mathematical terminology and such that each step in the argument is a logical consequence of earlier steps.

Learning skills: By the end of the course, students will be able to autonomously and critically reflect upon the mathematical correctness of their own arguments.

Study material: A. Chetwynd & P. Diggle: Discrete Mathematics. Butterworth- Heinemann, Oxford, ISBN 0 340 61047 6. Lecture notes will also be provided.

Recommended literature: None

Exam: Closed book written exam.

ECTS: 4

Period 1.2

Objects in Programming (KEN1220)

Coordinator: Dr. Thomas Bitterman

Examiners: Dr. Thomas Bitterman, Dr. Evgueni Smirnov & Theodor Schnitzler

Tutors: Prianikov, Nikola; Barta, Lázár; Doss, Heinz; Bams, Guillaume; Balan, Alexandra; Goldie, Samuel; Buiter Sanchez, Arantxa; Gójska, Maja; Timmermans, Derrick; Straka, Filip; Goffinet, Arthur

Desired prior knowledge: Procedural Programming

Prerequisites: None.

Description: This course is a follow-up to the course Procedural Programming. It teaches object-oriented programming in Java. The main topics covered in the course are objects and classes, interfaces and polymorphism, event handling, inheritance, graphic user interfaces, exception handling, and streams.

Knowledge and understanding: After successful completion of the course, students will be able to explain the methodological and theoretical principles of object-oriented programming.

Applying knowledge and understanding: Students will be able to implement basic object-oriented computer programs. They will be able to design and describe simple object-oriented computer systems.

Making judgements: Students will be able to judge the quality and correctness of simple object-oriented programs.

Communication: Students will be able to communicate about object-oriented programming constructs and algorithmic basics.

Learning skills: Students will be able to recognize their own lack of knowledge and understanding and take appropriate action such as consulting additional material or other sources of help.

Study material: Course notes, slides, and other information made available.

Assessment: Written exam (80%) + practical assignments (20%).

Recommended literature: C. Horstmann (2016). Java Concepts (8th Edition). John Wiley & Sons, New York, ISBN: 978-1-1190-5645-4.
C. Horstmann (2012). Big Java Late Objects. John Wiley & Sons, New York, ISBN 978-1-1180-8788-6

ECTS: 4

Calculus (KEN1440)

Examiner: Dr. Otti D'Huys & Dr. Gijs Schoenmakers

Prerequisites: None.

Description: The following subjects will be discussed in Calculus: limits and continuity, differential calculus, integral calculus, and an introduction to sequences and series and multivariable calculus. In addition to the main facts and concepts, problem-solving strategies will be discussed. Both the intuition behind the concepts and their rigorous definitions will be presented along with simple examples of formal mathematical proofs.

Knowledge and understanding: Student can define, write and explain key facts and concepts involving limits and continuity, can interpret and solve differential calculus, integral calculus, and understand the basics of multivariable calculus.

Applying knowledge and understanding: Students are able to solve problems applying the concepts learned in the course, using standard problem-solving strategies.

Making judgements: Students are able to analyse a simple problem within the course content and justify the solution methodology they choose. They can summarize this methodology mathematically.

Communication: Students are able to explain their solution strategy in written form and defend their solution strategy in discussion with others

Learning skills: After successful completion of the course the students will be able both to solve standard problems (constructing graphs of functions, finding extrema of functions, computing limits, summing infinite series etc.) and to apply their knowledge in solving and analysing more complex problems (e.g. in analysis of numerical algorithms).

Study material: Calculus, a complete course, any edition, by R.A. Adams, Addison Wesley Longman and materials provided during the lectures.

Exam: Intermediate bonus assignments and a final written exam.

ECTS: 4

Logic (KEN1530)

Coordinators: Dr. Tjitze Rienstra & Dr. Stefan Maubach

Examiners: Dr. Tjitze Rienstra, Dr. Stefan Maubach & Dr. Nico Roos

Description: This course deals with three logical systems, namely propositional logic, first-order predicate logic and dynamic logic. The course covers notation systems, syntax and semantics, valid consequences, deduction, semantic tableaux, and proof systems.

Knowledge and understanding: Students need to get accustomed to the fundamental concepts of mathematical logical systems (propositional logic and predicate logic) to able to describe information in a logical framework and to reason and prove correctly. Students will get accustomed to the basic concepts of some advanced logical systems (dynamic logic and Hoare logic).

Applying knowledge and understanding: Student will apply the reasoning and proof methods learned to small-scale problems and some more complex situations.

Making judgements: Students will learn to judge how to reason correctly using mathematical proofs and how to judge which logical system is suitable to solve the problem at hand.

Communication: The chosen syntax of the logical language used must be easily understandable by peers and other experts the logical proofs given must be correct, concise and easily understandable.

Learning skills: Having learned basic logical concepts and reasoning techniques the students are able to apply them to larger-scale problems.

Study material: Johan van Benthem, Hans van Ditmarsch, Jan van Eijck, Jan Jaspars, Logic in Action. Edition of February 2014 or later. This is a freely available e-book. Check your Canvas for the link.

Exam: Written exam.

ECTS: 4

Project 1-1 (KEN1300)

Coordinator: Dr. ir. Martijn Boussé

Description: Students work on a project assignment in small groups of six to seven students. The group composition stays the same for the whole project and is announced shortly before the project opening in period 1.1. The students are guided through the project by tutors (for project management) and mediators (for team dynamics). The project assignment is divided into three subtasks (one per period) and is strongly related to the course content from period 1.1 and 1.2. In period 1.1, after receiving the assignment for the whole project at the end of week 4, the students can start working on the project, and work full-time on the project in week 8 after the exams. In period 1.2, the students continue working on the projects in parallel to the other courses of that period. In period 1.3, the students work two weeks full-time on the project. The students meet their tutor about 3 times per period. A plenary Q&A with the examiners will be organized in period 1.1 and 1.2. The students will engage in several feedback and graded moments with the examiners during the project.

Knowledge and understanding: Interpret constraint-satisfaction problems arising in practice and translate this to discrete-mathematical algorithmic models capable of solving the problem. Gain insight into practical use of basic software design and development principles. Recognise and relate user-computer interactions to concepts from graphics and user-interface frameworks. Strengthen knowledge of basic algorithms and methods for efficiently solving constraint-satisfaction problems arising in applied mathematics (especially: discrete mathematics) and artificial intelligence.

Applying knowledge and understanding: Design an answer strategy for scientific questions using analytical thinking and logical reasoning. Translate discrete-mathematical algorithmic models to software code. Implement software to efficiently solve constraint-satisfaction problems arising in applied mathematics (especially: discrete mathematics) and artificial intelligence by finding, designing and applying appropriate algorithms. Formulate computational experiments, and analyse and interpret the results. Apply basic design and development principles in the construction of software systems. Use existing software application frameworks for graphics and user interfaces. Use tools for software project management such as version control systems and issue trackers. Identify project goals, deliverables, and constraints. Use simple project management tools. Work in a team such that the workload is balanced. Plan teamwork by setting deadlines and distributing tasks.

Making judgements: Evaluate different mathematical and computational models with respect to their suitability, efficiency and correctness for a specific task. Elicit and evaluate relevant scientific background information. Evaluate the group's progress during the project.

Communication: Give a clear and well-constructed presentation, including a demonstration of the product, and with appropriate use of illustrations and/or videos. Offer and respond to questions on and constructive criticism of presentations. Write a project report according to the structure of an academic article. Submit arguments in exact sciences, with appropriate use of formulae and figures. Cite published sources in the project report according to the academic guidelines. Structurally inform stakeholders on project progress. Effectively communicate with project group members about task division, planning and project deadlines. Effectively communicate with group members by listening to others' ideas; be contactable and include others in the discussion. Cooperate in a group to reach a consensus view. Give constructive feedback to team members. Communicate in the English language.

Learning skills: Reflect on one's own academic abilities and functioning in a team.

Study material: Project manual project 1-1, Maastricht University.

Assessment: The final grade will be composed of a project grade and a skill class grade. The project grades consists of several components such as project management, deliverables, presentation, and peer feedback. The skill class grade will depend on the total number of passed skill classes. (NG for the project is given if a student failed more than 2 skill classes).
Students not complying with attendance and participation requirements during the project meetings or examination moments may not be allowed to attend examination moments or may receive an NG.

Skill classes: Students will engage with a series of skill classes that prepare and support them for project work such as project management, team dynamics, and communication.

ECTS: 6

Period 1.4

Linear Algebra (KEN1410)

Examiner: Dr. Marieke Musegaas, dr. ir. Philippe Dreesen, & dr. Steve Chaplick.

Desired Prior Knowledge: None.

Prerequisites: None.

Course description: This course introduces the fundamental concepts of linear algebra, and examines them from both an algebraic and a geometric point of view. First, we address what can be recognized without doubt as the most frequently occurring mathematical problem in practical applications: how to solve a system of linear equations. Then we discuss linear functions and mappings, which can be studied naturally from a geometric point of view. Vectors spaces are then introduced as a common framework that brings all themes together. Next, we shift from the geometric point of view to the dynamic perspective, where the focus is on the effects of iterations (i.e., the repeated application of a linear mapping). This involves a basic theory of eigenvalues and eigenvectors, which have many applications in various branches of science as for instance in problems involving dynamics and stability, in control theory, and in optimization problems found in data science. Key concepts in the course are vectors, matrices, systems of linear equations, eigenvalues, eigenvectors, linear transformations, and orthogonality. The software package Matlab is introduced in the accompanying computer classes, where emphasis is put on the application of linear algebra to solve real world problems.

Knowledge and understanding: Students are able to recognize and explain the fundamental concepts of Linear Algebra: systems of linear equations, vectors and vector spaces, basis and coordinates, matrices and matrix-vector computations, linearity and orthogonality, linear independence, rank, fundamental spaces (row space, column space, and null space), determinants and invertibility, eigenvalues and Eigen spaces, diagonalization.

Applying knowledge and understanding: Students are able to analyse a linear algebra problem from both an algebraic and a geometrical point of view. Students can solve systems of linear equations, compute determinants and rank, compute eigenvalues and Eigen spaces, make use of complex numbers, diagonalizable matrices, and perform change of coordinates.

Making judgements: Students are able to look at the same problem from different angles and to switch their point of view (from geometric to algebraic and vice versa).

Communication: Students are able to motivate both from an algebraic and a geometric point of view the solution set of a system of linear equations, the linear independence and orthogonality of a set of vectors, the linear transformation between two coordinate systems, the fundamental spaces associated with a matrix, the invertibility of a matrix, and the diagonalization of a matrix in terms of the properties of its eigenvalues and eigenvectors.

Learning skills: Students have acquired the skills to autonomously recognize elements of practical problems, which can be addressed and solved with linear algebra, and use Matlab to solve larger scale problems.

Study material: David C. Lay, Linear algebra and its applications, 6th ed., Pearson, ISBN: 978-1-292-35121-6.

Recommended literature: None.

Exam: Closed book written exam

ECTS: 4

Data Structures and Algorithms (KEN1420)

Coordinator: Tom Pepels, M.Sc.

Examiners: Dr. Francesco Barile, Tom Pepels, M.Sc. & Dr. Bastian Küppers

Tutor(s): TBA

Desired prior knowledge: Programming in Java, Procedural Programming (KEN1120), Objects in Programming (KEN1220)

Prerequisites: None.

Description: The Data Structures and Algorithms course introduces the students to the design and application of data structures and algorithms. Abstract datatypes will be used as a central topic in this course. Together with the basic abstract data types such as trees, lists, and graphs, the associated algorithms and their complexity are discussed. The differences between the best, expected, and worst behaviour of an algorithm is explained. Supported by the concepts of complexity bounds and big O notation complexity is illustrated on several algorithms such as search, and string & graph algorithms. After completing this course, students will be able to determine the appropriate data structures and algorithms for simple problems.

Knowledge and understanding: Students will acquire a thorough understanding of both fundamental and complex data structures—ranging from arrays and linked lists to trees and graphs—alongside the principles of algorithm design, such as recursion, sorting, and graph algorithms. The curriculum emphasizes the importance of complexity analysis, teaching students to evaluate algorithm performance using Big O notation and other measures. Students will explore various algorithmic strategies, including dynamic programming and greedy algorithms, to solve computational problems efficiently.

Applying knowledge and understanding: Students will directly apply theoretical concepts through hands-on coding tutorials, designing and implementing algorithms to address specific problems. The primary learning goals include mastering the selection and application of appropriate data structures for optimizing software performance, conducting complexity analysis to evaluate and improve algorithm efficiency, and developing solutions for software development challenges. This approach aims to enhance students' problem-solving skills and prepare them for advanced computational tasks in their academic and professional futures. By the end of this course section, students will have gained experience in applying theoretical knowledge to practical scenarios, demonstrating their ability to navigate complex problems and develop efficient, effective solutions.

Making judgements: Students are tasked with developing the ability to critically assess the efficiency and effectiveness of different data structures and algorithms in solving computing problems. This involves comparing various algorithmic approaches based on their time and space complexities, understanding the trade-offs involved in algorithm selection, and justifying the choice of specific data structures for given scenarios.

Communication: Students will be able to explain how data structures and algorithms are to be included in program designs.

Learning skills: Students are encouraged to develop autonomous learning habits and critical thinking abilities. The focus is on fostering the capacity to independently acquire new computational techniques, adapt to evolving programming paradigms, and apply problem-solving strategies in unfamiliar contexts.

Study material: The course follows the Algorithms Fourth Edition book. Next to the book, weekly l ecture videos and short introduction videos to key topics are provided.

Exam: 'Closed Book' written exam

Recommended literature: Sedgewick and Wayne (2011) Algorithms Fourth Edition. Addison Wesley. ISBN: 978-0321573513

Additional literature: A Y Bhargava (2016). Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People. Manning. ISBN: 978-1617292231

ECTS: 4

Principles of Data Science (KEN1435)

Examiners: dr.ir. Marijn ten Thij & Dr. Anirudh Wodeyar

Desired prior knowledge: Procedural Programming

Prerequisites: None.

Description: Nowadays data science is at the core of modern society. We collect large amounts of data with the goal to make better decisions. We need to make sense of the data and leverage it in effective ways.

In this course, we will start with where data comes from—controlled experiments and observational studies. We will look at potential biases that can affect conclusions that we make from data. We will focus on what kind of causal statement one can draw based on data coming from experiments versus observational studies.

We will then summarize and visualize data using histograms and scatter plots. As we will see, there are some interesting recurring patterns when we summarize data. For example, the distribution of the average follows the bell shape curve. We will also consider deviations from the bell shape curve in case of outliers, and how to deal with real world and possible "unclean" data. Scatter plots will help us study the regression line and correlations.

This course will build the foundation for subsequent courses: probability and statistics, simulation and statistical analysis, and machine learning. You will learn how to convert data into tables and use them for subsequent analysis and plotting. We will focus on the principles of modern reproducible science, that is, to build analysis workflows that can easily be understood and re-run by others. We will learn how to keep track of analysis decisions and parameter choices. We will summarize all the uncertainties in an accessible way and see that this is crucial for effective decision making in the modern world.

During the labs, we will learn Python—one of the main programming languages used in data science— and how to use it to write analysis reports using literate programming—mixing code, plots, and narrative in the same document. We will analyze and visualize real datasets.

Knowledge and understanding: Students learn to organize, analyze, and visualize data. They understand what type of distribution to expect after summarizing the data.

Applying knowledge and understanding: Students analyze real datasets. They apply their knowledge about summaries of the data—and other tools in data science—and determine where they can be appropriately applied. They translate their understanding into conclusions for domain experts. Students develop skills to generalize data analyses to unseen contexts.

Making judgements: Students decide the limits on what can be learned from data. They judge the data based on the design of the experiment and the final goal of the analysis. They also predict the consequences of data misuse when making causal claims.

Communication: Students communicate their findings in written analysis reports. They write reports in Jupyter notebooks and compile them to html reports, so that they are accessible for domain experts.

Learning skills: Students develop skills to turn an abstract question into an actionable decision to gain insights.

Study material: Statistics (fourth edition) by Freedman, Pisani, and Purves, ISBN 9780393929720. Additional selected material from data science textbooks and other resources.

Assessment: 20% homework assignments and 80% written final exam

Recommended literature: None

ECTS: 4

Period 1.5

Computational and Cognitive Neuroscience (KEN1210)

Examiner: Dr. Alard Roebroeck and Dr. Michael Capalbo

Desired Prior Knowledge: None

Prerequisites: None.

Description: The course Computational and Cognitive Neuroscience presents an overview of the core topics in cognitive and biological psychology. These topics include (human) perception, learning, memory, planning, problem solving, reasoning, language, speech, and action. Both the functional and neuroanatomical foundations of cognitive faculties are addressed. Several models of cognition and theories of brain function that are of relevance to knowledge engineering will be outlined. Several skills trainings will be given to train understanding in biological functioning of neuronal communication, and functioning of neural networks and genetic algorithms.

Knowledge and understanding: The student can recount the main points of the domain of cognitive science
- The student can describe the main points of the domain of cognitive science
- The student can explain the following (human) behaviours while using these points: perception, learning, memory, planning, problem solving, reasoning, language, speech, and action.
- The student can identify the computational aspects and computational applications of these fields

Applying knowledge and understanding:
This knowledge is applied in in two practical assignments in which the students are asked to create a genetic algorithm and a neural network

Making judgements:
- Upon completion of the course, students are able to interpret data and literature about a subject in (or related to) the domain of cognitive and biological psychology.
- Using the data and literature, they can support judgements about the societal, scientific or ethical aspects of the subject.

Communication:
Students are able to communicate ideas and solutions to an audience of non-experts and experts.

Learning skills:
Students have acquired the skill to translate theoretical models into computational models.

Study material: Material will be provided during the course.

Recommended literature: Sternberg, R.J. (1999). Cognitive psychology (latest edition). Fort Worth: Harcourt Brace. Kalat, J.W. (2007) 9th edition Biological psychology. Pacific Grove, California; London: Brooks Cole. Gazzaniga, M. (2009). Cognitive Neuroscience (third edition).

Exam: Written exam.

ECTS: 4

Software Engineering (KEN1520)

Examiners: Tom Pepels, M.Sc. & Dr. Bastian Küppers

Tutor(s): TBA

Desired prior knowledge: Programming in Java, Procedural Programming (KEN1120), Objects in Programming (KEN1220)

Prerequisites: None.

Description: This course is designed to equip students with the foundational concepts and practices of software engineering, encompassing areas such as software design, management, testing, and maintenance.
Weekly sessions include a lecture complemented by a tutorial/problem-solving session, where students will engage in hands-on assignments. These activities aim to facilitate the application of class-taught concepts. In collaboration with a peer, students will undertake a small project tailored to highlight critical facets of software engineering.
Upon completion, it is anticipated that students will have acquired a comprehensive understanding of software engineering principles and practices, positioning them well to implement these skills in subsequent software development endeavors.

Knowledge and understanding: This course offers students an in-depth grasp of the key principles and methodologies in software engineering, including software design, development, testing, and maintenance. It covers the theoretical underpinnings of software engineering practices and introduces the core concepts needed to navigate the software development lifecycle effectively. Students will learn about various software architectures, programming paradigms, and the importance of quality assurance and software maintenance strategies.

Applying knowledge and understanding: Students will apply theoretical knowledge through practical assignments, collaborative projects, and problem-solving sessions. These activities are designed to reinforce the concepts learned in lectures, with a focus on real-world application. By engaging in hands-on tasks, students will develop the skills necessary to design, test, and maintain software systems, working both individually and as part of a team.

Making judgements: The course encourages students to critically evaluate different software engineering methodologies, tools, and practices. Students will learn to assess the appropriateness of various software development models for specific projects, make informed decisions regarding testing strategies, and determine the most effective maintenance approaches. This critical analysis aims to cultivate a nuanced understanding of how to navigate complex software engineering challenges.

Communication: Students will learn to articulate complex software design concepts and project requirements clearly, both verbally and in writing. Collaborative projects and presentations will further enhance their ability to communicate technical information effectively to diverse audiences, including team members and non-technical stakeholders.

Learning skills: The course is designed to foster lifelong learning skills, preparing students to adapt to new technologies, methodologies, and changes in the software engineering field. Through self-directed learning, reflective practice, and feedback incorporation, students will enhance their ability to independently acquire new knowledge and skills. This foundation will support their ongoing professional development and adaptability in the evolving landscape of software engineering.

Study material: Next to the recommended materials, weekly lecture videos and short introduction videos to key topics are provided.

Assessment:
- Written "closed-book" exam at the end of the course worth 80% of the final grade.
- During the course, students receive several graded assignments in the form of a project that count for 20% of the final grade.
- Attending 6 out of 7 tutorial sessions rewards 0.5 bonus points toward the final grade.

Recommended literature:
- Goldman and Miller, MIT 6.031: Software Construction, http://web.mit.edu/6.031/
- Martin, Clean Code: A Handbook of Agile Software Craftsmanship (2008)

ECTS: 4

Numerical Mathematics (KEN1540)

Coordinator: Dr. Pieter Collins

Examiners: Dr. Pieter Collins & Dr. Ir. Martijn Boussé

Desired prior knowledge: Calculus, Linear Algebra

Prerequisites: None.

Description: Numerical mathematics is the art of solving mathematical problems with the aid of a digital computer. In this course we will cover the fundamental concepts of numerical mathematics, including the floating-point representation of real numbers, truncation and round-off errors, iterative methods and convergence. We will study the simplest and most important algorithms for core problems of numerical mathematics, namely the solution of algebraic equations and differential equations, interpolating data by polynomials, numerically estimating derivatives and integrals, approximating functions by polynomials and trigonometric series, solving systems of linear algebraic equations and computing eigenvalues. There will be a strong practical component, with students being expected to write their own numerical code and test the performance and suitability of different methods on various problems.

Knowledge and understanding: By the end of this course, students will have knowledge of the fundamental problems of numerical mathematics and basic techniques for their solution. You will understand issues of efficiency and numerical accuracy, will be able to analyse which numerical methods are likely to perform best on different types of problem, and evaluate whether the results of a given computation are trustworthy. You will be able to write your own code (in MATLAB) implementing basic numerical algorithms. Advanced students will have the skills necessary to adapt existing numerical algorithms and develop new algorithms.

Applying knowledge and understanding: Students will be expected to implement the algorithms covered in the lectures themselves, apply them to practical problems, and explain the performance of different algorithms in terms of theoretical analyses.

Making judgements: Students will learn how to analyse which numerical methods are likely to perform best on different types of problem, and to evaluate whether the results of a given computation are trustworthy.

Communication: Students will learn the terminology required to discuss numerical algorithms and the results of numerical computations with mathematicians, (social) scientists and engineers.

Learning skills: Students will learn to design, analyse, implement and apply numerical methods.

Assessment: Written examination with formula sheet (100%). Preparatory exercises (+10% bonus).

Recommended literature: J.D. Faires & R. Burden, "Numerical Methods", International 4th Edition, Cengage, 2012; ISBN: 978-0-495-38569-1.

Additional literature: C.F. Gerald & P.O. Wheatley, "Applied Numerical Analysis", Seventh Edition, Pearson, 2003; ISBN: 0-321-13304-8.
T. Siauw & A.M. Bayen, "An Introduction to Matlab Programming and Numerical Methods for Engineers", Academic Press, 2015; ISBN 978-0-12-520228-3.

ECTS: 4

Project 1-2 (KEN1600)

Coordinator: Dr. Otti D'Huys

Prerequisites: In order to participate in this project the student has to have passed two out of four courses from the set: Discrete Mathematics, Calculus, Procedural Programming and Objects in Programming.

Description: Students work on a project assignment in small groups of about seven students. The group composition stays the same for the whole project and is announced before the project opening in period 1.4. The students are guided through the project by a fixed tutor. The project assignment is related to the content of the courses from year 1. In period 1.4, after receiving the assignment for the whole project at the end of week 5, the students start working on the project in parallel to their courses. They meet their tutor approximately once every week. In period 1.6, the students work three weeks full-time on the project and meet their tutor twice a week.

At the beginning of period 1.5, the students hand in a planning, along with a short summary about the work completed so far, and receive feedback from the examiners. By the end of period 1.5 the students have a midway evaluation as formative assessment, and hand in a draft report. In period 1.6, they submit a final report on their project and attend a final examination.

Knowledge and understanding: Interpret the meaning of mathematical models of real-world processes. Gain insight into practical use of software design and development principles. Recognise and relate user-computer interactions to concepts from graphics and user-interface frameworks. Strengthen knowledge of basic algorithms and methods for specific problems in artificial intelligence and applied mathematics.

Applying knowledge and understanding: Students will be able to design an answer strategy for scientific questions using analytical thinking and logical reasoning and to translate mathematical models to software code. Furthermore, students will be able to implement software to solve problems in applied mathematics by applying numerical methods and artificial intelligence algorithms, formulate computational experiments, and analyse and interpret the results, apply design and development principles in the construction of software systems and use existing software application frameworks for graphics and user interfaces. Even more so, students will learn to use tools for software project management such as version control systems and issue trackers, identify project goals, deliverables, and constraints. Lastly they will learn how to plan and chair meetings, create notes for minutes, work in a team such that the workload is balanced and plan teamwork by setting deadlines and distributing tasks.

Making judgements: Students will learn to evaluate different mathematical and computational models with respect to their suitability, efficiency and correctness for a specific task.

Communication: Students will be able to give a clear and well-constructed presentation, including a demonstration of the product, and with appropriate use of illustrations and/or videos, to offer and respond to questions on and constructive criticism of presentations. Furthermore, they will learn to write a project report according to the structure of an academic article, submit arguments in exact sciences, with appropriate use of formulae and figures. They learn to cite published sources in the project report according to the academic guidelines. Additionally, students will learn to structurally inform stakeholders on project progress and effectively communicate with project group members about task division, planning and project deadlines, effectively communicate with group members by listening to others' ideas; be contactable include others in the discussion. It will be important to cooperate in a group to reach a consensus view, communicate in the English language, elicit and evaluate relevant scientific background information.

Learning skills: Reflect on one's own academic abilities and functioning in a team.

Study material: Project manual project 1-2, Maastricht University.

Assessment:

$$FinalGrade=0.9\cdot(ProjectGrade \vee IndividualGrade)+skillClassGrade$$

The *individualGrade* is given due to either outstanding or not enough contribution of a student to the project. By passing skill classes, the students can get a reward called *skillClassGrade*, which is 1 if the students passes sufficiently many components of all skill classes, 0.5 if the students passed most components, and 0 if the students fails for a critical amount of the skill class tasks. Failing all components of more than 2 skill classes will lead to an NG in the project. Missing mandatory project events such as project meetings and examination moments will lead to a reduction of the grade or even to receiving an NG for the project.

Skill classes: There will be skill classes on the following topics (each with components spread over periods 1.4-1.5)

Information Research: Systematic Literature Search
These skill classes will give the students an introduction to which databases, search strings and

settings can be used to systematically search for literature, and are guided in drafting a search plan for the relevant literature in the project.

Team Dynamics
The team dynamics workshops aim to provide the students with a deeper awareness, insight and practice in effective team collaboration & co-creation. In a later stage, students evaluate the team collaboration and communication by means of interactive exercises.

Academic Writing
In the project skills components you will explore the key structure of your report, as well as key points of Academic Writing at Maastricht University. Areas of focus include: structure of paper; linguistic aspects of writing in English, presenting information logically and citation and reference procedures.

ECTS: 6

2.2 Curriculum of the second year of the Bachelor Programme Data Science and Artificial Intelligence

Year 2		ECTS
Period 2.1	Databases (KEN2110) Probability and Statistics (KEN2130) Graph Theory (KEN2220) Project 2-1 (KEN2300)	4 4 4
Period 2.2	Reasoning Techniques (KEN2230) Machine Learning (KEN2240) Simulation and Statistical Analysis (KEN2530) Project 2-1 (KEN2300)	4 4 4
Period 2.3	Project 2-1 (KEN2300)	6
Period 2.4	Human Computer Interaction and Affective Computing (KEN2410) Theoretical Computer Science (KEN2420) (**) or Multivariable Calculus (KEN3250) (**) Mathematical Modelling (KEN2430) Project 2-2 (KEN2600)	4 4 4
Period 2.5	Philosophy and Artificial Intelligence (KEN2120) Linear Programming (KEN2520) Natural Language Processing (KEN2570) Project 2-2 (KEN2600)	4 4 4
Period 2.6	Project 2-2 (KEN2600)	6

(*) Project 2-1 will start in period 2.1 and will run until period 2.3 with weekly meetings; Project 2-2 will start in period 2.4 and will run until period 2.6 with weekly meetings. The credits for the projects will become available at the end of period 2.3 and 2.6 respectively.

(**) Elective: in case students have passed both electives of period 2.4, Theoretical Computer Science or Multivariable Calculus can replace 1 of the third year electives.

For each period, we will give a short explanation of the various parts. Before the start of each period, the students will receive detailed information about the content, the study material, the teaching form, the schedule, and the examination method.

Period 2.1

Databases (KEN2110)

Coordinator & examiner: Tom Pepels, M.Sc.

Tutor(s): TBA

Desired prior knowledge: Programming in Java or other comparable language, Data Structures and Algorithms (KEN/BCS1420), Software Engineering (KEN1520)

Prerequisites: None.

Description: This course delves into the principles and practical applications of database systems, emphasizing relational databases and data modeling aimed at developing robust, data-intensive software applications. Students will learn to utilize Structured Query Language (SQL) for effective data manipulation, ensuring database transactions are Atomic, Consistent, Isolated, and Durable (ACID). The curriculum expands to cover alternative data storage solutions, including distributed databases and NoSQL technologies, offering insights into various object persistence techniques. Throughout the course, participants will gain hands-on experience with different database management systems (DBMS), learning how to select and employ these systems in the construction of sophisticated software solutions.

Knowledge and understanding: Students will acquire a solid understanding of database fundamentals, including the core concepts of database management systems, query languages, data modeling, and database programming. The course aims to build a comprehensive foundation, enabling students to articulate the principles underpinning relational and distributed database technologies and their application in real-world scenarios.

Applying knowledge and understanding: Participants will apply their knowledge to design and implement efficient database solutions tailored to specific system requirements. They will explore the capabilities and limitations of various database types, integrating software architectures to devise and develop comprehensive database applications. Practical exercises will reinforce the theoretical concepts, enhancing students' competency in database design and implementation.

Making judgements: Students will develop the ability to critically analyze database design challenges, comparing different modeling approaches to optimize database structures based on practical use cases. This includes the capacity to refine database models, propose enhancements to existing designs, and evaluate the integrity and effectiveness of database implementations critically. Through case studies and project work, students will practice making informed decisions to solve complex database problems.

Communication: The course will equip students with the skills to effectively communicate database concepts, designs, and solutions to a varied audience, including developers, database administrators, and end-users. Emphasis will be placed on articulating the key entities, relationships, and processes involved in database systems, fostering clear and concise communication in technical and non-technical contexts.

Learning skills: By engaging with the course content, students will cultivate the ability to independently explore and understand advanced topics in database management and development. This includes identifying relevant literature and resources beyond the course materials, encouraging continuous learning and adaptation to emerging database technologies and methodologies.

Study material: Weekly lecture slides and extra materials

Exam: Written exam (75%) + practical assignment (25%)

Recommended literature: Alan Beaulieu, 2020. Learning SQL, (3rd ed.). O'Reilly Media, Inc.

Additional literature: Martin Kleppmann, 2017. Designing Data-Intensive Applications. O'Reilly Media, Inc.

ECTS: 4

Probability and Statistics (KEN2130)

Examiner: Dr. Christof Seiler

Desired Prior Knowledge: Discrete Mathematics and Calculus

Prerequisites: None.

Course Description: This course is a first introduction to probability and statistics. We will start by learning how to count and define a notion of probability. We will then move on to the concept of conditional probability, random variables and their distributions, expectation, continuous random variables, moments, joint distributions, and inequalities and limit theorems. This will provide us with the necessary language to study central topics of importance in statistics, such as the difference between a population and a sample, confidence intervals, parameter estimation, and hypothesis testing.

Knowledge and understanding: In this course, the students obtain tools to define random variables and identify probability distributions in a wide range of probabilistic experiments. Furthermore, they know which procedure is most appropriate to find an answer to a given statistical question.

Applying knowledge and understanding: Students are capable of calculating probabilities, expectations, variances and related quantities in a wide range of probabilistic experiments. Furthermore, they can estimate statistical quantities and perform statistical tests to extract information from data sets.

Making judgements: Students can critically analyze probabilistic experiments and statistical inferences and can decide whether to accept or reject statistical hypotheses.

Communication: The students will be able to communicate their conclusions and the underlying rationale to expert and non-expert audiences.

Learning Skills: Students are able to use elements from probability theory and statistics in other domains in order to increase one's knowledge.

Study material: Hwang and Blitzstein, Introduction to Probability (2019, second edition)

Exam: 20% homework assignments and 80% written final exam

ECTS: 4

Graph Theory (KEN2220)

Examiner: Dr. Matus Mihalák

Desired Prior Knowledge: Discrete Mathematics; Data Structures and Algorithms

Prerequisites: None

Description: A graph is simply a collection of points, some of which are joined by lines. This deceptively simple structure is one of the cornerstones of both theoretical and applied computer science. A great many problems that arise in the real world can be modeled as graph problems. Several classical examples include the problem of finding the shortest route between two cities, of maximizing flow in a network of pipelines, or of finding an optimal pairing between producers and consumers. In this course we will look at both the algorithmic/applied side of graph theory and its more abstract mathematical foundations, because the latter is often important for understanding the former. We will cover topics such as paths, tours, trees, matchings, flows and colorings.

Knowledge and understanding: Students will have a solid overview of the basic concepts and results of (applied) graph theory, including the main mathematical tools to argue about graphs. Students will have the tools to model and analyze various real-world problems using graphs.

Applying knowledge and understanding: Students will be able to recognize when a problem can be modeled with graphs, and whether the problem can be efficiently solved using standard or slightly adjusted graph-theoretic algorithms.

Making judgements: Students will be able to formulate a given (sub)problem as a graph-theoretic problem, argue why the formulation is correct, and they will be able to judge the feasibility of existing algorithmic solutions.

Communication: Students will be able to explain, in the language of graph theory, how a problem at hand can be modelled and solved.

Learning skills: Students will enhance their study skills such as time management, effective reading, critical thinking and reading, exact and unambiguous writing and formulation of ideas and statements, and reflection on marked (graded) work. Along the way, students will improve general learning skills such as self-motivation, careful listening and giving instructions, and openness to new knowledge.

Study material: Appropriate material will be provided during the course.

Recommended literature: None.

Exam: Written exam (80% of the final grade). Bi-weekly graded exercises (20% of the final grade).

ECTS: 4

Reasoning Techniques (KEN2230)

Coordinator: Dr. Tjitze Rienstra

Examiners: Prof. Dr. Mark Winands and Dr. Tjitze Rienstra

Desired Prior Knowledge: Introduction to Data Science and Artificial Intelligence; Logic.

Description: Central in this course is how, based on available data, new knowledge and information can be obtained using reasoning processes. The course will be supported by tutorials, in which the acquired techniques can be put into practice by using Prolog. The following four techniques are discussed:

1. Reasoning using logic: syntax, semantics, and inference in first-order logic, situation calculus, forward and backward reasoning, completeness, logic programming with Prolog.
2. Problem solving using search: problem types, blind-search methods, informed-search methods, comparison of search methods, games as search problems, minimax, alpha-beta pruning, Monte Carlo Tree Search, chance games, constraint satisfaction problems.
3. Planning: planning in situation calculus, representation of states, goals and operators, state space and plan space, algorithms for classic planning.
4. Reasoning with uncertainty: uncertainty and probability theory, conditional probability, the Rule of Bayes, semantics of belief networks, exact and approximate inference in belief networks.

Knowledge and understanding: Students learn to understand how problems can be represented as logical problems, as search problems, as planning problems or as problems involving uncertainty and get accustomed to reasoning methods to solve problems of all four types mentioned above.

Applying knowledge and understanding: Students learn to apply the reasoning methods learned to toy problems and some more complex situations.

Making judgements: Students learn to judge which type of knowledge representation is suitable for the problem at hand, and which reasoning technique is suitable to solve the problem at hand. Communication: students can explain the knowledge representation used and reasoning technique chosen to peers and other experts.

Communication: Students can explain the knowledge representation used and reasoning technique chosen to peers and other experts.

Learning skills: Students are able to critically reflect on their own and other's chosen representations and used reasoning methods.

Study material: Russell, S. and Norvig, P., Artificial Intelligence: A Modern Approach, 4th edition. Pearson, 2020. Bratko, I. (2012). Prolog: *Programming for Artificial Intelligence*, 4th edition. Addison-Wesley , 2011.

Assessment: Closed-book written exam (80% of final grade) and assignments during the course (20% of final grade).

Recommended literature: Luger, G.F., Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 6th edition. Pearson International Edition, 2009.

ECTS: 4

Machine Learning (KEN2240)

Examiners: Dr. Evgueni Smirnov and Dr. Enrique Hortal Quesada,

Desired prior knowledge: Procedural Programming, Calculus, Linear Algebra, Logic, Probability and Statistics

Prerequisites: None

Description: Machine learning is a major frontier field of artificial intelligence. It deals with developing computer systems that autonomously analyse data and automatically improve their performance with experience. This course presents basic and state-of-the-art techniques of machine learning. Presented techniques for automatic data classification, data clustering, data prediction, and learning include Decision Trees, Bayesian Learning, Linear and Logistic Regression, Recommender Systems, Artificial Neural Networks, Support Vector Machines, Instance-based Learning, Rule Induction, Clustering, and Reinforcement Learning. Lectures and practical assignments emphasize the practical use of the presented techniques and prepare students for developing real-world machine-learning applications.

Knowledge and understanding: After successful completion of the course, students will be able to describe and explain the basic machine learning algorithms. Students will understand the mathematical foundation of machine learning algorithms and how mathematical methods are successfully combined to obtain the variety of machine learning algorithms that are currently available.

Applying knowledge and understanding: Students will acquire the knowledge to apply, formulate, and validate techniques from machine learning and to apply basic machine learning algorithms to real-life problems. Students will be able to implement machine-learning algorithms in software and apply existing machine learning software implementation to datasets. Students will have the necessary knowledge to design, implement, and apply data processing systems that autonomously extract information from data, interpret results, and make decisions.

Making judgements: Students learn how to critically analyse real-world problems, select appropriate machine learning techniques according to the specific problem, and predict the consequences of their choices. After successful completion of the course, students gain the ability to judge which problems can be solved better and to which extend through the application of machine learning techniques. Students obtain an awareness of and responsibility for ethical and social consequences of developments in and application of machine learning.

Communication: The skills acquired during the course will allow students to present the results of different stages of the application of machine-learning techniques to specialists or non-specialists.

Learning skills: After successful completion of the course, students can analyse, adapt, design, implement, and critically reflect on machine-learning algorithms and tools. Students also obtain the critical fundamental skills and knowledge to study further advanced machine learning techniques in the professional literature.

Study material: Lecture material provided during the lecture.

Recommended literature:
- T. Mitchell (1997). Machine Learning, McGraw-Hill, ISBN-13: 978-0071154673.
- H. Blockeel, Machine Learning and Inductive Inference (course text), Uitgeverij ACCO, 2012.
- I.H. Witten and E. Frank (2011). Data Mining: Practical Machine Learning Tools and Techniques (Third Edition), Morgan Kaufmann, January 2011, ISBN-13: 978-0123748560.

Exam: Written “open-book” exam at the end of the course. During the course, students receive several graded assignments that can earn them a maximum bonus grade of 1.0.

ECTS: 4

Simulation and Statistical Analysis (KEN2530)

Examiners: Dr. Joel Karel and Dr. Marijn ten Thij

Prior Knowledge: Knowledge: Probability & Statistics, Calculus, Matlab, and Java.

Prerequisite: None.

Description: Mathematical simulation is concerned with studying processes and systems. Uncertainty can be an important factor and must be modelled properly. For modelling systems, all the available data must be analyzed. After modelling a complex system, various scenarios can be simulated, using Monte Carlo simulation, to gain insight. The results need to be properly interpreted and the experiments can be designed in such fashion that the downstream analysis is obeying certain assumptions. Furthermore, uncertainty has to be reduced and one must understand how uncertainty influences decisions. The modelling, implementation, analysis, and technical aspects will be discussed as an introduction in this field. Emphasis will be on discrete event simulation and the statistical analysis of the output of simulation studies, where topics are: modelling, Poisson processes, random number generators, selecting and testing input distributions, generating random variates, experiment design, statistical analysis of experiments, comparing experimental results and variance reduction. Practical exercises will be used to place the techniques in context.

Knowledge and understanding: Define concepts of simulation, discrete event simulation and statistical inference. Explain techniques underlying mathematical simulation. Explain methods for analyzing experimental results and efficient simulation including their assumptions, justify why they are important, and match them to simulation design.

Applying Knowledge and understanding: Being able to model a system in a structured manner, to design and implement simulators for systems, and to collect data from these simulations. In addition, you will be able to employ techniques underlying mathematical simulation and apply methods from statistics for analyzing experimental results.

Making judgements: Being able to choose and motivate alternative techniques underlying mathematical simulation. Choose, motivate, and contrast methods for statistical analysis.

Communication: Being able to convey the phases of a specific simulation study to non-experts. Being able to explain the assumptions and choices made when analyzing experimental data to experts and non-experts.

Learning Skills: The ability to independently learn to handle large-scale simulation. To identify shortcomings in data analysis.

Study material: Simulation Modeling and Analysis (5th edition) -Averill Law

Recommended literature: Object-Oriented Computer Simulation of discrete-event systems – Jerzy Tyszer, Design and Analysis of Experiments – Douglas C. Montgomery, Introduction to Probability Models – Sheldon M. Ross.

Exam: Written exam and assignments and/or bonus assignments.

ECTS: 4

Project 2-1 (KEN2300)

Coordinator: Dr. Katharina Schneider

Prerequisites: Students must have passed Project 1-1. Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms. The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 1. This project is a prerequisite for Project 3-1.

Description: Students work on a project assignment in small groups. The group composition stays the same for the whole project and is announced at the beginning of period 2.1. Throughout the project, the groups are guided by a tutor with respect to the project management. The project assignment is related to the content of the courses from period 2.1 and 2.2. In periods 2.1 and 2.2, the students work on the project, while also having to attend the courses of these periods. They meet their tutor approximately once every two weeks. In period 2.3, the students work three weeks full-time on the project and meet their tutor about once to twice a week.

The focus of this project lays on the software implementation/design and the product functionality. During the project, the students have to hand in several deliverables such as a project plan after a few weeks from the start or the implemented code at the end of the project. Peer feedback on implementations will add to the quality of feedback the students receive. Presentations throughout the project will be used to communicate the progress to the examiners.

Applying knowledge and understanding: Students will learn to concretize project assignment and construct and maintain a planning Furthermore, they will learn formulating, selecting and validating models for the problem chosen and collect and interpret experimental data with evaluation metrics. Lastly they will improve their ability to plan and chair meetings, create notes for minutes, work in a team such that the workload is balanced and plan teamwork by setting deadlines and distributing tasks.

Making judgements: After completing this project, students will be able to compare and criticize results, position them in terms of the literature diagnose limitations and formulate a discussion.

Communication: Students will be able to write a scientific paper that: describes the project, explains the methods, summarizes the outcomes, discusses them and makes the conclusions. Students will be able to present and defend project in English and coordinate project progress in project meetings

Learning skills: Students will be able to reflect on the progress of the project and study relevant literature to solve problem at hand.

Study material: Project manual project 2-1, Maastricht University.

Assessment: The assessment is composed of a grade for the following deliverables:
- Project plan
- Product
- Report

Furthermore, a grade for the project management and a peer feedback grade on the product functionality will be included in the final grade.

As the focus of this project lays on the software implementation/design and product functionality, the grade for the product has the highest weight.

The examiners can deviate from the group grade if a student shows either outstanding or not enough contribution to the project. Missing mandatory project events such as project meetings and examination moments will automatically lead to a reduction of the grade or even to receiving an NG for the project.

By passing skill classes, the students can get a reward called skillClassGrade, which is 1 if the students passed all skill classes, 0.5 if the students passed all but one skill classes, and 0 if the students passed all but two skill classes. Failing more than two skill classes will lead to an NG in the project.

Skill classes: The project comes along with skill classes that enhance the students' soft and hard skills. They are closely related to the deliverables of the project. Skill classes are mandatory to pass to complete the project.

ECTS: 6

Period 2.4

Human Computer Interaction and Affective Computing (KEN2410)

Coordinator: Konstantia Zarkogianni

Examiners: Konstantia Zarkogianni & Yusuf Can Semerci

Tutors: Konstantia Zarkogianni & Yusuf Can Semerci

Desired prior knowledge: Machine Learning, Probabilities and Statistics.

Prerequisites: None.

Description: Within the frame of this course the multidisciplinary field of Human-Computer Interaction and Affective Computing (HCI&AC) will be studied. HCI is the study of interaction between people (users) and computers. It is often regarded as the intersection of computer science, behavioural sciences, design, and several other fields of study. Interaction between users and computers occurs at the user interface, which includes both software and hardware; for example, characters or objects displayed on a personal computer's monitor and input received from users via hardware peripherals such as keyboard, mouse and web cameras. The cource will provide a comprehensive exploration of the field, preparing students to understand, analyze, and design interactive systems effectively. This course also covers Affective Computing, a new branch of HCI that places emphasis on user emotions and personality. Affective Computing attempts to bring emotions into intelligent interfaces that interact with humans and see how they can have a positive and constructive impact in human-machine interactions. Emphasis will also be placed on data analysis, from the perspectve of analyzing UX while interacting with interfaces. The course follows a project-based approach.

Knowledge and understanding: Upon completion of the course, students will be able to understand the multidisciplinary nature of the HCI while building know how in prototyping techniques and technologies for testing a user interface. They will understand the main usability directives - principles, standards, guidelines, and patterns – within the frame of designing a user interface. They will also get familiar with affective computing and how relevant technologies can help in designing an effective user-interface.

Applying knowledge and understanding: After successful completion of the course, students will be able to design and implement low-fidelity prototypes of UI to serve as a tool for requirements extraction. They will also have the capacity to implement functional prototypes of UI for usability testing and conduct relevant user studies involving human participants. Students will be able to perform statistical data analysis and analyze emotions and personality traits.

Making judgements: Students will develop critical skills towards designing, implementing, and assessing hard coded tasks that are appropriately linked with the usability principles.

Communication: Students will develop communication and presentation skills to effectively convey innovative and complicated user-interfaces within a short time interval.

Learning skills: By the end of the course, students will be able to recognize challenges in human computer interaction systems and design UI elements reflecting end users' emotions and personality traits.

Study material: Lecture notes and slides

Assessment: Assignment

Recommended literature:
- Shneiderman B, Plaisant C, Cohen M, Jacobs S, Elmqvist N, Diakopoulos N. (2016) Designing the user interface: strategies for effective human-computer interaction. Pearson, ISBN: 978-0134380384
- Calvo RA, D'Mello S, Gratch JM, Kappas A, (2015). The Oxford handbook of affective computing. Oxford University Press, ISBN: 978-0199942237
- Software tools for prototyping (e.g. Adobe XD, Justinmind, Mupixa)

Additional literature:
- Coursera video lectures of Scott Klemmer and accompanying slides.

ECTS: 4

Theoretical Computer Science (optional course) (KEN2420)

Examiner: Dr. Georgios Stamoulis

Desired Prior Knowledge: Introduction to Data Science and Artificial Intelligence, Discrete Mathematics, Data Structures and Algorithms.

Description: This course explores the theoretical underpinnings of computing by investigating algorithms and programs casted as language recognition problems. The influence of the theory on modern hardware and software system design is demonstrated. The following subjects will be treated: mathematical foundations, alphabets and languages, finite automata and regular languages, Turing machines, acceptance and decidability, recursive functions and grammars, time complexity classes, NP problems, NP-completeness

Knowledge and understanding: Students will learn to comprehend the inherent complexity of problems and be able to motivate why some problems are inherently more difficult than others are. They will learn to have insight into how complex problems can be solved efficiently and will be able to classify such problems into a language hierarchy and complexity classes. Furthermore, students will be able to apply the tools needed for such classification

Applying knowledge and understanding: students will be able to apply the theory learned to solve small-scale problems

Making judgements: Students will learn to judge which problems are decidable and efficiently solvable and to judge which technique is suitable to solve the problem at hand.

Communication: The knowledge representation used and technique from complexity theory chosen must be easily understandable by peers and others experts

Learning skills: The student will learn to reflect on own one's and other's thoughts on complexity and solvability of problems.

Study material: Elaine Rich (2008), Automata, Computability and Complexity, Prentice Hall, New Jersey, ISBN 0-13-228806-0.

Exam: Written exam; during the course the students will receive three assignments, that, if they receive a sufficient grade, may earn them up to a total of one bonus point.

ECTS: 4

Mathematical Modelling (KEN2430)

Examiner: Dr. Joel Karel and Prof. Dr. ir. Ralf Peeters.

Desired Prior Knowledge: Linear Algebra, Calculus, Matlab.

Description: Mathematical modelling is of great importance for solving practical problems by casting them into a form suitable for the use of mathematical techniques. In this course, a number of basic topics are discussed. First, attention is paid to a framework for mathematical modelling. Then we focus on some widely used model classes from engineering, in particular on the class of linear time-invariant dynamical models. These are described by linear difference equations (in discrete time) or linear differential equations (in continuous time). Alternative model descriptions that are discussed are transfer functions (in the frequency domain) obtained with the z-transform and the Laplace transform respectively; and state-space models, which may or may not involve canonical forms. Some further topics receiving attention are the concepts of stability, sinusoidal fidelity, Bode diagrams, the interconnection of subsystems, and the technique of pole placement by means of state feedback.

The subject matter is clarified through exercises and examples involving practical applications. Also, relevant functionality in Matlab is introduced, which offers a powerful instrument for analysing linear dynamic models.

Knowledge and understanding: Being able to formulate linear dynamical models, state properties and define representations. Identify frequency domain properties of systems and relate them to applications in signal processing.

Applying knowledge and understanding: Being able to construct elementary mathematical models. Perform model analysis and extract model properties. Employ various model representations and choose the most appropriate one. Compute state-feedback control.

Making judgements: To recognize what are the important aspects to consider when building a mathematical model. Decide on stability of models.
Communication: Being able to convey properties of models to specialists and non-specialists.

Learning skills: Being able to independently find Matlab functionality to solve basic problems in systems theory.

Study material: Lecture notes.

Recommended literature: Richard J. Vaccaro, Digital Control: A State-Space Approach, McGraw-Hill, 1995, ISBN 0-07-066781-0. Robert L Williams , Douglas A Lawrence, Linear state-space control systems, Publisher: John Wiley & Sons 2007, ISBN: 978-0-470-11787-3, 0-470-11787-7.

Exam: Written exam and assignments and/or bonus assignments.

ECTS: 4

Multivariable Calculus (optional course) (KEN3250)

Coordinator & examiner: Dr. Stefan Maubach

Tutors: student TAs + dr. Stefan Maubach

Desired prior knowledge: Calculus, Linear Algebra

Description: Multivariable calculus develops calculus into more than one variable. Multivariable calculus can help to better understand techniques and algorithms used to analyze multidimensional data or optimization techniques used to train neural networks.   In this course, we use vector calculus combined with calculus/analysis, and we introduce path functions, curves, surfaces and volumes, and learn how to perform various types of integration: integration along paths, over surfaces and volumes, and flux. We learn how to transform into different coordinate systems (polar, cylindrical, spherical). We discuss extensions of the chain rule and product rule to higher dimensions, and the corresponding extensions of derivative and tangent lines/planes. We cover complex numbers and some of complex functions. We also cover some methods to solve optimization problems. We look at several types of differential equations.

Knowledge and understanding: Students understand various multivariable analysis and calculus problems. They know what it means to differentiate and integrate in more than one variable, and they understand specific optimization problems, and how to solve those. They are able to link their knowledge of linear algebra to multivariate calculus. They understand the theoretical background of specific problems in multivariable calculus, and are able to give (partial or full) proofs of some of the theory.

Applying knowledge and understanding: Students can model certain types of practical problems, or problems which are not exactly stated, into calculus problems. They are able to differentiate and integrate in more than one variable, and how to solve specific types of differential equations.

Making judgements: Students are able to judge what is the correct path to take to tackle a challenging problem, especially when there are several ways to solve it, not all of which are optimal or equally efficient. They can judge results of calculations and what they represent.

Communication: Students can communicate their results, and defend its validity if asked.

Learning skills: Students learn how to deal with a topic that at first glance has an unforeseeable amount of different cases, and how to extend and apply this knowledge to new problems.

Study material:
- Main source: Lecture notes provided on the course page
- Complimentary source: Hass, Heil, Weir, Bogacki, Thomas, Universty Calculus fifteenth edition in SI units, Pearson, ISBN 978-1-292-25311-4 (Many versions of this book exist, exercise numbers and paragraphs differ between versions.)

Exam:
- Closed book
- A formula sheet may be provided on exam (will be exactly as provided beforehand on the course page)
- Allowed aids: calculator from DACS allowed calculator list
- The exam is 120 minutes, without breaks

Skill classes: Tutorials.

ECTS: 4

Period 2.5

Philosophy & Artificial Intelligence (KEN2120)

Examiner: Dr. Robert Gianni

Prerequisites: none.

Description: One of the characteristics of scientific knowledge is the translation of natural phenomena into quantitative or mathematical data – the book of nature, Galileo wrote, is written in the language of mathematics. Over the course of the twentieth and twenty-first century, this desire to understand the world through the logic of mathematics has been extended beyond the natural world to include such things as human consciousness, learning, and intelligence. Indeed, the foundation of what is called 'artificial intelligence' is the pursuit of replicating human consciousness and intelligence through mathematical models and formulas. In this course we will examine these issues from a philosophical perspective, beginning with a basic overview of the notion of intelligence with an emphasis on quantification and then moving on to study philosophical issues that have developed out of the pursuit of artificial intelligence. We will analyse different understandings of intelligence, the effects of AI-based technologies like Large Language Models and continue through ethics framework aimed at addressing the role of technology in specific ways (e.g. robotics). Finally, we will operate an overview of the main regulatory frameworks from an ethical and political perspective.

Knowledge and understanding: At the conclusion of this course, students should be able to demonstrate knowledge of the following topics:
- The understanding of different meanings of intelligence
- The philosophical presuppositions of artificial intelligence
- The societal impact of artificial intelligence
- The ethics theories addressing artificial intelligence
- The international research frameworks and regulatory standards around artificial intelligence

Applying knowledge and understanding: Students will be able to draw upon both lectures and readings to write an essay that exhibits critical reflections on conventional and naïve notions of instrumentalism, technological determinism, and functionalism by persuasively arguing for a contextual approach that highlights the contingency and flexibility of design and meaning.

Making judgements: Students are asked to select relevant passages from texts that contribute to the argument that they make in the essay. This will be graded. In tutorials, students are asked to make decisions about specific problems (i.e. self-driving cars, Turing tests). This is not graded.

Communication: During tutorials, students will discuss the texts orally with their classmates and work in groups to provide experiment around alternative conceptions of AI.

Learning Skills: Students will be able to articulate and solve problems in groups. Students will also be expected to engage with a number of theories concerning computation and artificial intelligence through different texts and will be asked to reflect upon and critique these theories.

Study material: Selected texts will be made available.

Exam: Multiple answers and open questions.

ECTS: 4

Linear Programming (KEN2520)

Examiner: Dr. Steven Kelk

Desired Prior Knowledge: Linear Algebra.

Prerequisites: None

Description: A linear program is very different to, say, a Java program. It simply consists of a linear objective function (of potentially very many variables) and a set of linear inequalities. The goal is to find values of the variables, which maximize or minimize the objective function, subject to all the inequalities being satisfied. Linear programs - even very large linear programs - can be solved extremely quickly, in both theory and practice. The model is also expressive enough to capture a large number of real-world problems. These two factors explain the fundamental role of linear programming in operations research, computer science, economics, management and many other fields. The course consists of an in-depth study of the simplex algorithm (a standard algorithm for solving linear programs), duality theory, and sensitivity analysis. Examples from practice illustrate the power of the model and teach the student the skill of modelling. Practical aspects of linear programming (e.g. use of software packages for solving linear programs, and integration with languages such as Java) are also considered.

Knowledge and understanding: Students will be able to identify which real-world optimization problems can be formulated as linear programs. Students will be able to describe the mathematical foundations of the Simplex algorithm for solving linear programming, and articulate how these foundations impact upon the performance of the Simplex method in practice. Students will recognize the power and importance of duality theory for reasoning about the behaviour of linear programs (in particular with regard to sensitivity analysis). Students will be able to exhibit an awareness of non-Simplex paradigms for solving linear programs (interior-point methods) and be able to recount the importance of the linear programming model in operations research and applied mathematics.

Applying knowledge and understanding: Students will be able to 1) translate mathematical models into linear programs, 2) to apply the Simplex method by hand to solve small linear programs, 3) to show how the Simplex method behaves in normal and exceptional cases, 4) to manipulate the algebra underpinning the Simplex method, 5) to combine insights from this algebra and primal-dual relations to make rigorous statements about the (sub)optimality of solutions to linear programs, 6) to argue how small changes to linear programs impact upon their optima (sensitivity analysis), 7) to explain key differences between the Simplex method and interior-point methods, and to 8) leverage linear-to program arguments when developing simple algorithms for combinatorial optimization problems.

Making judgements: Students will be able to distinguish between mathematical models that can and cannot be cast as a linear program. Students will be able to contrast and compare the behaviour of the Simplex algorithm with interior-point methods. Students will be able to select, out of a large range of algebraic and duality-based instruments, appropriate tools for making rigorous statements about linear programs.

Communication: Students will be able to formulate linear programs and defend their correctness. Students will be able to clearly articulate and defend algebraic and duality-based arguments concerning linear programs.

Learning Skills: By the end of the course, students will be able to autonomously and critically reflect upon the appropriateness of the linear programming paradigm for tackling optimization problems arising in practice and be able to assess the correctness of mathematical arguments pertaining to linear programming. Students will be able to identify follow-up literature, which goes beyond the scope of the material presented in the course.

Study material: Hillier & Lieberman (2010 or 2015): Introduction to Operations Research (9th or 10th edition). McGraw Hill, ISBN 978-007-126767-0 or ISBN 9781259162985. Support for the 11th edition is forthcoming.

Recommended literature: students are beforehand encouraged to refresh their knowledge of: (unique) solutions of systems of linear equations, matrix inversion, and matrix rank.

Exam: Written exam and optional bonus exercises (the results of which are added to your exam score, up to 10%).

ECTS: 4

Natural Language Processing (KEN2570)

Examiners: Dr. Jerry Spanakis and Dr. Aki Härmä

Desired Prior Knowledge: Procedural Programming, Objects in Programming, Probability and Statistics, Machine Learning

Prerequisites: None

Description: ChatGPT can answer almost any question you have. Siri can tell me when I need an umbrella. But how do they work? Over the past few years, Natural Language Processing (NLP) was revolutionized by statistical, probabilistic and machine learning methods. NLP addresses fundamental questions at the intersection of human language and machine learning. How can computers acquire, understand and produce language? How can computational methods give us insight into observed human language phenomena? How to make sense of the vast amounts of information available online in free, unstructured form? In this course students will learn how computers can learn useful text/language representations and how different tasks (language modelling, text classification, information extraction, sequence labeling, etc.) can be used for solving different complex problems (spelling correction, spam detection, search engine design, opinion analysis, summarization, question-answering, etc.). Open NLP problems (such as evaluation or interactive dialogue systems) and the effect of deep learning on NLP will be discussed.

Knowledge and understanding: By the end of the course, students are able to acquire the basic text and language processing aspects. Furthermore, students are able to describe basic NLP problems, tasks and methods.

Applying knowledge and understanding: Students are able to demonstrate how to tackle a text/ language problem and to formulate, design and implement a NLP system. Students are able to suggest when a problem's complexity requires an NLP solution.

Making Judgements: Students are able to pose questions and define problems in different domains (e.g. social sciences) and contexts (e.g. business) that include language/text data. Furthermore, students are able to judge which tools are applicable for solving these problems and to decide a course of action in accordance with ethical and social consequences.

Communication: Students are able to outline an approach in real organizational problems, which require NLP and are able to demonstrate, present and communicate a solution to a NLP problem

Learning Skills: Students are able to master and choose the appropriate basic programming tools for NLP and are able to follow up on literature that will allow them to build complete NLP models.

Study material: Handouts, Jupyter and Google Collab notebooks

Recommended Literature:
1. Daniel Jurafsky and James H. Martin. "Speech and language processing an introduction to natural language processing, computational linguistics, and speech." Pearson, London, 2000
2. Chris Manning and Hinrich Schütze. "Foundations of Statistical Natural Language Processing". MIT Press. Cambridge, MA. 1999
3. Jacob Eisenstei. " Introduction to natural language processing". MIT press, 2019.
4. Yoav Goldberg. "Neural Network Methods for Natural Language Processing". Synthesis Lectures on Human Language Technologies #37, Morgan and Claypool, 2017.

Exam: Practical individual assignments (30%) + Open-Book Written Exam (70%).

ECTS: 4

Project 2-2 (KEN2600)

Coordinator: Dr. Katharina Schneider

Prerequisites: Students must have passed Project 1-2. Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms. The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 2. This project is not a prerequisite for another project / course.

Description: Students work on a project assignment in small groups of about six students. The concrete assignment is defined by the students given some umbrella topics that match the courses in the curriculum. Students indicate their umbrella topic preference at the beginning of period 2.4. The group composition stays the same for the whole project and is based on the topic preferences of the students. A least regret algorithm is used to ensure that the overall regret is minimized. Throughout the project, the groups are guided by a tutor with respect to the project management. In periods 2.4 and 2.5, the students work on the project, while also having to attend the courses of these periods. They meet their tutor approximately biweekly. In period 2.6, the students work three weeks full-time on the project and meet their tutor about twice a week.

The focus of this project lays on the project planning and communication of progress and results. During the project, the students have to hand in several deliverables such as a project plan after a few weeks from the start or the implemented code at the end of the project. Formative feedback sessions with the examiners on intermediate project plans will add to the quality of feedback the students receive. Presentations throughout the project will be used to communicate the progress to the examiners.

Applying knowledge and understanding: Students will learn to set up a project assignment and construct and maintain a planning. Additionally, they will learn formulating, selecting and validating models for a concrete problem at hand and to collect and interpret data with evaluation metrics. Lastly they will improve their ability to plan and chair meetings, create notes for minutes, work in a team such that the workload is balanced and plan teamwork by setting deadlines and distributing tasks.

Making judgement: After completing this course successfully, students will be able to compare and criticize results, position them in terms of the literature; diagnose limitations and formulate a discussion

Communication: Students will be able to write a scientific paper that: describes the project, explains the methods, summarizes the outcomes, discusses them and makes the conclusions. Furthermore, student will be able to present and defend project in English. Coordinate project progress in project meetings

Learning skills: Students will learn to reflect on the progress of the project and study relevant literature to solve problem at hand

Study material: Project Opening slides, Maastricht University

Assessment: The assessment is composed of a grade for the following deliverables:
- Project plan
- Poster
- Report

Furthermore, a grade for the project management, the groups' performance during the poster session and during a defense at the end of the project, and a peer feedback grade on the poster presentation will be included in the final grade.

As the focus of this project lays on the planning and the communication of progress and results, the grades for the project plan, the poster presentation and the defense together have the highest weight.

The examiners can deviate from the group grade if a student shows either outstanding or not enough contribution to the project. Missing mandatory project events such as project meetings and examination moments will automatically lead to a reduction of the grade or even to receiving an NG for the project.

By passing skill classes, the students can get a reward called skillClassGrade, which is 1 if the students passed all skill classes, 0.5 if the students passed all but one skill classes, and 0 if the students passed all but two skill classes. Failing more than two skill classes will lead to an NG in the project.

Skill classes: The project comes along with skill classes that enhance the students' soft and hard skills. They are closely related to the deliverables of the project. Skill classes are mandatory to pass to complete the project.

ECTS: 6

2.3 Curriculum of the Third Year of the Bachelor's Programme Data Science and Artificial Intelligence

The first semester of the third year allows you to make your own selection of subjects in the field of artificial intelligence, data science, applied mathematics and computer science, the core areas of the study of Data Science & Artificial Intelligence. In each of the periods 1 and 2, you choose 3 out of 6 optional courses. The first semester of year 3 has the same structure in the first and the second year; there are two periods of eight weeks and one period of four weeks. There is also a project in period 3.3. Alternatively, students can choose to study the first semester of the third year at a partner university abroad. Please check the Study abroad guide for more info: Going Abroad | DACS Intranet. Other options will be uploaded by the student counsellors..

Year 3		ECTS
Period 3.1*	Digital Society (KEN3111) Game Theory (KEN3130) Semantic Web (KEN3140) Recommender Systems (KEN3160) Robotics and Embedded Systems (KEN3236) Introduction to Quantum Computing (KEN3241) Project 3-1 (KEN3300)	4 4 4 4 4
Period 3.2*	Computer Security (KEN2560) Software and Systems Verification (KEN3150) Logic for Artificial Intelligence (KEN3231) Parallel Programming (KEN3235) Large Scale IT and Cloud Computing (KEN3239) Introduction to Bio-Informatics (KEN3440) Project 3-1 (KEN3300)	4 4 4 4 4 4 4
Period 3.3	Project 3-1 (KEN3300)	6
Period 3.4	Operations Research Case Studies (KEN3410) Intelligent Systems (KEN3430) Data Analysis (KEN3450)	4 4 4
Period 3.5-3.6	Bachelor's thesis (KEN3500)	18

(*) Third year students choose three electives per period out of the optional courses during period 1 and 2

*** Project 3-1 will start in period 3.1 and 3.2 with weekly meetings. The credits for the project will become available at the end of period 3.3.*

For each period, we will give a short explanation of the various courses. Before the start of each period, the students will receive detailed information about the content, the study material, the teaching form, the schedule, and the examination method.

Digital Society (KEN3111)

Lecturer & Examiner: Dr. Matthew Archer

Prerequisites: none.

Desired prior knowledge: none.

Description: Digitalization has a profound impact on our society. We can observe changes in different areas. What digital technologies do, what they look like and how they relate to each other is not identical worldwide, but dependent on local practices as well. Usually new technologies are understood as innovation and progress: and indeed, digital technologies improve a broad range of domains, such as healthcare or education. New possibilities as e.g. participation in our digital cultures arise but also new inequalities, as the access and competences needed for participation are not evenly distributed and the platforms that allow for participation also harbour new mechanisms of control and surveillance. The pace and diversity of these developments ask for continuous investigation and reflection. It requires work to shape and use technologies in ways that contribute to the public good. Moreover, digital technologies have also led to highly problematic developments such as electoral manipulation, fake news and algorithmic discrimination.

Technological developments are often conceived as predefined or given. Does a society's technology drive the development of its social structure and cultural values? Scholars in science and technology studies have shown that technology and society are deeply intertwined. Technology is inherently social. Technologies are shaped by people; they emerge and are embedded in social practices.
The aim of this course is to investigate the consequences of digitalization for our society/societies. These consequences have been differently valuated: participation vs. exploitation of users, innovation as enhancement vs. challenge, ethics and techno-moral change vs./and sustainability.
We will discuss digitalization from

- a social perspective when we read about digital participation and how technology and society are intertwined
- a political perspective when we discuss activism, digital citizenship but also problems of manipulation and verification (as in the case of fake news and deep fakes)
- a cultural perspective when we analyze imaginaries and discourses around innovation of technology and promises being made
- a legal perspective when we discuss privacy and the attempts to adapt privacy laws
- an ethical perspective when we discuss design decisions, privacy but also techno-moral change and questions of environment and sustainability.

The course is structured in the following way:
Transformations
(digital participation, digital citizenship, data-activism)

Imaginaries
(innovation and techno-moral change)

Disruptions
(fake news and deep fakes, sustainability and e-trash)

Knowledge and understanding: Students acquire knowledge on the impact of digitalization on society.

Applying knowledge and understanding: Students learn to understand the interrelation between digital technology and sociality.

Making judgements: Upon completion of the course, students can reflect on ethical challenges related to digitalization.

Communication: Students are able to communicate central topics related to digitalization to an audience of non-IT-experts (e.g. the debate will bring students from the FASoS Bachelor Digital Society and students from the bachelor's Data Science and Artificial Intelligence following this course together to train both groups to communicate topics related to digitalization from a social science and IT perspective).

Learning Skills: Students are able to reflect critically in written form on a topic related to the digital society but also to do so orally in a presentation and debate.

Study material: The literature (provided via the reference list of the library).

Exam: (group) presentation in class (1-3 students) per task (25% of the final grade), 2 short academic papers of 1500 words each (2x25% of the final grade) and participation in a final debate (25% of the final grade).
If a resit is needed for the (group) presentation, the presentation will be given via video (e.g. Zoom or Skype). If a student needs to resit the papers they can be rewritten and improved based on the comments of the tutor. If a resit is needed for the debate (in case a student does not show or participate in the debate), the student can write a 1500-word paper on the content of the debate instead.

ECTS: 4

Game Theory (KEN3130)

Examiner: Prof. Dr. Frank Thuijsman

Prerequisites: Discrete Mathematics, Linear Algebra.

Description: We introduce the field of Game Theory. Game Theory is the mathematical study of problems, called games, that involve two or more decision makers, called players, who each have their own individual preferences over the possible outcomes. In a game, each player always aims to maximize his individual payoff and chooses his actions accordingly. These actions may be probabilistic or deterministic, depending on the situation. Meanwhile he reasons logically about actions that might be taken by the other players. A basic difference exists between strategic and non-strategic models. Both types of models and their solution concepts will be discussed. Issues like value, fairness, manipulations, threats, optimality and rationality will be addressed.

Knowledge and understanding: Students can recognize and classify the main types of games, i.e. cooperative games, strategic games, bipartite matching problems, and formulate the main solution concepts value, optimal strategies, Nash- and correlated equilibrium, as well as a number of algorithms to calculate these.

Applying the use of knowledge and understanding: Students can calculate solutions for the different types of games

Making Judgements: Students can explain advantages and disadvantages of different solution concepts. They are able to judge the correctness of solutions presented Communication: Students can explain and defend the correctness of their solutions

Learning Skills: By the end of the course, students will be able to autonomously and critically reflect upon the pros and cons of different types of games for modelling competition and cooperation. This includes considerations on the computational aspects with respect to different solution concepts.

Study material: Lecture notes.

Examination: There will be a closed book written exam at the end of the course.

ECTS: 4

Semantic Web (KEN3140)

Examiner: Dr. Remzi Celebi

Desired Prior Knowledge: Logic.

Description: Most of the information available on the World Wide Web (WWW) is not directly understandable for computers. For instance, web pages are designed for human readability. Computer programs have difficulty in interpreting the information presented on web pages. The focus on human readable information introduces restrictions on what computer programs can do to support human users in tasks such as:

- finding information
- buying goods
- making travel plans

The Semantic Web should eliminate these restrictions by separating the content of what is presented on a web page from the way it is presented. In recent years, the focus has shifted to providing data, independent of webpages (for example: Linked Open Data (LOD)

Ontologies are used to provide a shared conceptualization of information. Ontologies form the basis of the Semantic Web, Knowledge Based System, Databases, etc., and they play an important role in data exchange and interoperability in many domains. Ontologies are applied in the bio-medical domains, in data mining applications, in Linked Open Data (LOD), in websites based on semantic technology, etc.

Since ontologies are intended to be shared between different systems, defining an ontology is a challenging task.

This course will focus on the standards the World Wide Web Consortium (W3C) is defining in order to realize the Semantic Web. The course also addresses the underlying knowledge representation formalisms of the current semantic web standards. Moreover, the course will address the engineering principle of crating an ontology. Note that the course does not address standards for making websites.

Knowledge and understanding: Making the student familiar with the developments and standards of the Semantic Web. The student will get insights in semantic web standard, such as RDF, RDFa, SPARQL and OWL2. Moreover, the students will get some basic insight in the semantics of RDF and the Description Logic underlying OWL. Finally, the student will be made familiar with the ontology development process, and criteria for evaluating an ontology. The student should understand the role of upper ontologies and ontology design patterns, as well as the philosophical choices they represent.

Applying knowledge and understanding: The student should be able to build applications using semantic web standards such as RDF, RDFa, SPARQL and OWL2. The student should also be able to develop an ontology for an application domain.

Making judgements: The student should be able to judge whether and how semantic web standards can be applied in applications. The student should also be able to judge the quality of an ontology.

Communication: The student should have sufficient understanding of the Semantic Web and its standards in order to explain why and how an application should be set up using semantic web standards. The student should also be able to explain and defend the choices made in the ontology engineering process.

Learning skills: The student should be able to study the literature about semantic web developments.

Study material:
- A Semantic Web Primer, Grigoris Antoniou, Paul Groth, Frank van Harmelen and Rinke Hoekstra, MIT Press, ISBN: 9780262018289 (third edition).
- Syllabi and scientific papers about ontology engineering.

Recommended Literature: The documents on the site of the World Wide Web Consortium (W3C).

Examination: Practical exercises and a written exam at the end of the course. The grade is for 70% determined by the written exam and for 30% by practical assignments. Participation in the practical is required for receiving a grade.

ECTS: 4

Recommender systems (KEN3160)

Examiner: Prof. Dr. Nava Tintarev & Dr. Francesco Barile

Desired Prior Knowledge: Natural Language Processing, Human Computer Interaction & Affective Computing

Prerequisites: Machine Learning

Description: Recommender systems play an important role in helping to mediate many of our everyday decisions and choices, including the music we listen to, the news that we read, and even the people that we date. They do this by learning from our past interactions, inferring our interests and documenting our preferences. To make the right suggestions at the right time recommender systems must not only understand our preferences but also our current needs and perhaps our immediate intent. Thus, the core focus of most recommender systems is devoted to profiling users and matching items based on these profiles and current context.

Much of the research to date on recommender systems has focussed on the engineering and evaluation of core recommendation algorithms. Researchers have developed a variety of approaches to harness different forms of preference data in the pursuit of more accurate recommendations. For example, researchers have used simple ratings for collaborative, rich meta-data for content-based methods, and even the opinions and sentiment expressed within user-generated reviews. When evaluating recommender systems, there has been a heavy emphasis on measuring the accuracy of suggestions, or the error of predictions. However, in practice it is important to consider evaluation metrics beyond accuracy, such as diversity, novelty, and serendipity. This in turn has led to increased attention being given to the nature of the interactions between users and recommender systems, and the influence that the user interface and interaction style can have on user behaviour and the overall recommendation experience. This course focuses on:

- Non-personalized and Stereotype-based Recommender Systems
- Classical recommender systems algorithms, e.g., Content-based Filtering, Collaborative-based Filtering
- Offline Evaluation e.g., protocols, criteria, metrics
- User-centered evaluation
- Interfaces and interaction in Recommender systems, e.g., explanations and conversational recommender systems
- Group Recommender Systems
- Ethics, bias, and fairness in recommender systems
- Advanced methods, e.g., Matrix Factorization, Hybrid recommender systems, Contextual Recommender systems

Knowledge and understanding: Students will be able to explain concepts from recommender systems, such as the difference between different recommendation methods and can identify advantages and limitations of these methods. Students will also be able to explain one advanced method or topic suitable for progression to a Master level program in Data Science or Artificial Intelligence.

Applying knowledge and understanding: Students will be able to apply ideas, methods, and tools for recommender systems that are suitable for a given domain. Students will be able to solve problems and design analytically, to comprehend (design) problems and abstract their essentials, to construct and develop logical arguments with clear identification of assumptions and conclusions. Students will develop the ability to transpose academic knowledge and expertise into (inter)national societal, professional and business contexts.

Making judgements: Students will gain acquaintance with the standards of academic criticism. Students will develop an awareness of, and responsibility for ethical, normative and social consequences of developments in science and technology, particularly resulting from Data Science and Artificial Intelligence.

Communication: Students will develop academically and internationally appropriate communicative skills, i.e., the ability to give effective oral presentations, both formally and informally, and understand and offer constructive criticism of the presentations of others.

Learning skills: Students will be able to reflect on their own working methods, and own readiness to take the necessary corrective action.

Study material: Course notes, required reading of scientific articles.

Recommended Literature: Jannach, Dietmar, et al. Recommender systems: an introduction. Cambridge University Press, 2010. Additional research papers and online articles

Exam: The assessment is composed by three components: Individual “Review” of Scientific Papers (15%), Practical group assignment (30%), and Written exam (55%). The three components are mandatory (e.g., A grade of at least 5.5 for each component is necessary in order to pass the exam).

ECTS: 4

Robotics and Embedded Systems (KEN3236)

Examiner: Dr. Rico Möckel.

Desired prior knowledge: Calculus, Linear Algebra, Machine Learning.

Prerequisites: Procedural Programming and Objects in Programming

Description: Nowadays, a variety of products require that algorithms from data science and artificial intelligence are adapted to and implemented in robotic and embedded systems. Applications that heavily rely on intelligent robotic and embedded systems include self-driving cars, autonomous drones, intelligent industrial robots in (semi-) autonomous factories, smart phones, intelligent medical devices, and distributed intelligent embedded devices in smart homes.

In this course, students receive an introduction to the fields of robotics, embedded systems, and real-time control. Students obtain an overview of state-of-the-art intelligent robotic and embedded systems in academia and industries. Students gain hands on experience in programming embedded robotic systems using embedded processors and a modular robotic system developed at the Department of Advanced Computing Sciences. Students learn about communication standards for embedded systems, sensors, and actuators. Student practise and strengthen their expertise in data science and knowledge engineering by applying mathematical methods for controlling robotic systems: They study control techniques including PID control, forward and inverse kinematics as well as locomotion control and learning using central pattern generators. The course concludes with a robot competition where students build and program robots using a modular robotic system.

Knowledge and understanding: Students obtain knowledge in designing, building, and programming robotic and embedded systems. Students learn how to apply mathematical concepts like dynamic systems for controlling robotic systems in real-time. Students further obtain knowledge about sensors and motor control and study the application of machine learning and mathematical methods for learning and optimizing control parameters. Students receive training in the programming language C - the most popular languages for programming microcontrollers.

Applying knowledge and understanding: After successful completion of this course, students can analyse, apply, implement, and validate control techniques in embedded and robotic systems with and without real-time constraints. Students can apply techniques from machine learning, search, and optimisation to obtain parameters for embedded control systems as required in many professional academic and industrial applications.

Making judgements: Students learn to judge where real-time systems are required and embedded systems can be beneficial. Students further learn to critically analyse the use of robotic systems in a variety of scenarios and to make design choices for robotic and embedded systems. By introducing students to a variety of state-of-the-art robotic systems, the course lays the foundation so that students can process professional literature in robotics and embedded systems.

Communication: Students will be able to 1) discuss robotic and embedded systems professionally and critically, 2) plan, discuss, implement, and validate projects in robotics and embedded, 3) present the results of project assignments in form of video, and to 4) critically analyse and explain control techniques for robotic systems to a general and professional audience.

Learning Skills: Students are able to autonomously and critically reflect upon the abilities and limitations of robotic and embedded systems in order to keep up with new developments in the field. Students can further assess the capabilities and limitations of their own solutions to a control or machine learning problem in robotics, and to identify follow-up literature, which goes beyond the scope of the material presented in the course.

Study material: Course material including video tutorials will be provided during the lectures.

Assessment: The final course grade is 80% of the final written "closed-book" exam grade plus 20% of the assignments grade.

ECTS: 4

Introduction to Quantum Computing (KEN3241)

Lecturer & Examiner: Dr. Menica Dibenedetto

Prerequisites: Linear Algebra.

Desired prior knowledge: Theoretical Computer Science, Data Structures & Algorithms.

Description: This course offers an introduction to the interdisciplinary field of quantum computation. The focus will lie on an accessible introduction to the elementary concepts of quantum mechanics, followed by introducing the mathematical formalism and a comparison between computer science and information science in the quantum domain. The theoretical capability of quantum computers will be illustrated by analysing fundamental algorithms of quantum computation and its potential applications.

Quantum technology has become one of the most prominent interdisciplinary fields of recent research. This course will focus on introducing the mathematical concepts underpinning quantum computation, and on explaining how this new computational paradigm might potentially offer possibilities beyond the scope of conventional computers. Topics that will be introduced and discussed include: (i) most common models of quantum computation (e.g., quantum circuits and measurement-based quantum computing). (ii) An exposition of the machinery borrowed from quantum mechanics, such as superposition of states, quantum entanglement, (de)coherence etc., which gives rise to the potential speed-up of quantum algorithms over their classical analogs. (iii) Some of the most common quantum algorithms (searching, factoring etc.) and protocols (quantum teleportation, EPR paradox). The course will finish with an exposition of potential applications of quantum computation and algorithms in other fields (such as security/cryptography, AI, optimization etc.)

Important: no prior knowledge in quantum mechanics is assumed or required, and all necessary concepts will be introduced and motivated from a mathematical and theoretical computer science point of view. Possible quantum architectures and/or related hardware issues will not be discussed.

Knowledge and understanding: By the end of this course, students are able to understand the differences between classical and quantum computation: Where is the computational power of quantum machines coming from? What are the limits of this new computational paradigm? What does the term "quantum supremacy" mean and why it is important? How likely is it ever to be achieved and what would it mean for our current understanding of the computational landscape?

Applying knowledge and understanding: Students are able to understand some of the most famous quantum algorithms, and to demonstrate where their power comes from. They will be able to judge how this potential computational power can be leveraged, and how it can be applied to other fields in a beneficial way.

After successful completion of this course, students are able to understand and use the mathematical framework of quantum computing to solve computational problems.

Making judgements: Students are able to judge and identify the settings where the potential quantum power might be beneficial and how they can leverage this. Students will further be able to analyse simple quantum algorithms for different computational problems.

Communication: Students are able to discuss quantum computation critically and judge not only its benefits but, equally important, its shortcomings. During lectures and practical assignments, students will be exposed to a different way of thinking about computation that will also enhance their understanding on classical computation.

Learning Skills: Students are able to critically read and understand scientific papers on quantum computing. To explain and analyse quantum algorithms described in quantum circuit or measurement-based quantum computing models. Finally, to relate quantum complexity classes to the classical ones.

Recommended Study material:
- Isaac Chuang, Michael Nielsen, "Quantum Computation and Quantum Information", 10th Anniversary Edition, Cambridge University Press, 2011.
- N. David Mermin, "Quantum Computer Science: An Introduction", 1st Edition, Cambridge University Press, 2007

Course material will be also provided during the lectures.

Exam: The final course grade is 100% of the final written "closed-book" exam grade.

ECTS: 4

Period 3.2

Computer Security (KEN2560)

Coordinator & examiner: Dr. Bastian Küppers

Desired prior knowledge: Procedural Programming, Objects in Programming, Data Structures and Algorithms, Software Engineering, Databases

Prerequisites: None.

Description: Computer security is the process of securing information systems against unauthorized access. As information systems have become mandatory in the modern world, coupled with the increased frequency of security incidents, organizations now recognize the need for a comprehensive security strategy. The course will introduce a wide range of topics in computer security and online privacy. The main objective of the course is to cultivate a security mindset by discussing various attack techniques and appropriate defenses. The topics that will be explored are information security (cryptography, cryptoanalysis), software security and network security, as well as designing secure systems. The class consists of lectures in which several computer security issues will be discussed. In parallel, there are assignments in which the students have to solve some of the most important issues discussed during the lectures.

Knowledge and understanding: Students will gain an in-depth understanding of computer security fundamentals and their applications in real world scenarios. In detail, they will understand the principles of a secure computer system and the potential attacks that could compromise it.

Applying knowledge and understanding: After completing the course, the students will be able to design and develop secure systems on their own.

Making judgements: By understanding the fundamentals of computer security and by working on the assignments, the students will be able to understand and spot mistakes in the security of computer systems.

Communication: Students will be able to explain the principles of computer security to specialists and non-specialists. They will be able to explain if the design of a system is secure or not and how the system can be improved.

Learning skills: Students will be able to read and interpret scientific literature on computer security that goes beyond the scope of the course, and independently design. They will be able to design and implement secure computer systems, to avoid common mistakes, and to work on real-world security problems.

Study material: Lecture slides, Source code examples

Assessment: Group project (20%)

Exam: Written final exam (80%)

Recommended literature: Pfleeger & Pfleeger: Security in Computing

Additional literature:
- Goodrich & Tamassia: Introduction to Computer Security
- Buchmann: Introduction to Cryptography
- Tanenbaum & Bos: Modern Operating Systems

ECTS: 4

Software and Systems Verification (KEN3150)

Coordinator & examiner: Dr. Pieter Collins

Desired prior knowledge: Reasoning Techniques, Theoretical Computer Science

Prerequisites: None.

Description: Have you ever written a program with a bug in it? Then this course is for you! Software verification tools can check whether your program works by showing that it correctly satisfies its specification, or finds a case in which it can go wrong. Unlike unit testing and other software validation methods, verification tools use formal methods to rigorously prove correctness. Similar techniques can be used to show that (mathematical models of) cyber-physical robotic systems work as designed.

In this course, we will start by and introducing the main notions of object-oriented program verification, including pre- and post-conditions for methods, and class invariants. We shall use Hoare logic to convert programs and their specifications into logical statements to be proved. We shall apply these techniques to the verification of simple programs written in Java.

In the second part of the course, we consider formal models of software and systems as labelled transition systems (automata), using temporal logics for specification, and consider the fundamental algorithms for verification. We shall apply these algorithms to simple discrete verification problems, such as vending machines and communications systems, modelled using a formal system specification language. Finally, we will look at simple continuous systems, such as robots and electronic systems, and show how to verify these using rigorous numerical methods based on interval arithmetic.

Knowledge and understanding: Students are able to recognise the difference between formal verification and validation, and distinguish rigorous numerical methods, notably how interval arithmetic differs from floating-point. They can explain the various kinds of annotations used in program specification, state the deduction and precondition rules of Hoare logic, and interpret linear temporal logic formulae.

Applying knowledge and understanding: Students are able to write formal specifications for simple programs. Furthermore, students can use Hoare logic to reduce program specification to first-order logic statements, and justify these. Students are able to construct Büchi automata accepting temporal logic formulae and can apply interval and affine arithmetic for verifying properties of continuous systems. Moreover, students are able to write annotations for object-oriented software, use software for model-checking discrete systems and use software for rigorous numerics to verify safety of simple continuous systems.

Making judgements: Students are able to determine the most appropriate modeling framework and verification tools for a given problem.

Communication: Students can write and read formal specifications and can discuss informal design goals and their translation into formal specifications.

Learning skills: Students will critically reflect on their own human reasoning and the potential of digital computers to perform provably-correct automated reasoning.

Study material: Course notes.

Recommended literature:
- J.B. Almeida, M.J. Frade, J.S. Pinto & S. Melo de Sousa, "Rigorous Software Development: an Introduction to Program Verification", Springer, 2011.
- C. Baier & J.P. Katoen, "Principles of Model Checking", MIT Press, 2008.
- L. Jaulin, M. Kieffer, O. Didrit & E. Walter, "Applied Interval Analysis", Springer, 2001.

Exam: Written exam, closed book (100%).

ECTS: 4

Logic for Artificial Intelligence (KEN3231)

Examiner: Prof. Dr. ir. Nico Roos.

Desired prior knowledge: Knowledge of propositional and predicate logic.

Prerequisites: The first year bachelor course: Logic (KEN1530).

Description: Logics form the formal foundation of knowledge representation and reasoning, which is a fundamental topic in Artificial Intelligence. Logics play a role as an analysis aid and as a knowledge-representation formalism. Moreover, the semantics of logics enables us to evaluate the intended meanings of knowledge representation formalisms, and the correctness and completeness of reasoning processes.
Humans make assumptions in their day-to-day reasoning. Examples of reasoning with assumptions

are: common sense reasoning, model-based diagnosis, legal argumentation, agent communication and negotiation, and so on and so forth. The assumptions humans use in their reasoning may be incorrect in the light of new information. This implies that conclusions may have to be withdrawn in the light of new information. Therefore this form of reasoning is called non-monotonic reasoning and the underlying logics are called non-monotonic logics.

The course will cover model-based diagnosis as an application of reasoning with assumption, standard logics extended with defeasible rules, argumentation systems, the semantics of reasoning with assumptions and defeasible rules, and closure properties of the reasoning systems.

Knowledge and understanding:
- The student should be able to describe non-monotonic logics and argumentation systems.
- The student should be able to identify the logic underlying specific forms of knowledge representation.
- The student should be able to describe and discuss the semantic of non-monotonic logics.

Applying knowledge and understanding:
- The student should be able analyze important properties of practical formalisms for knowledge representation and reasoning.
- The student be able to apply non-monotonic logics and argumentation systems to practical problems

Making judgements:
- The student should be able to judge whether specific knowledge representation formalisms are able to represent the intended meaning of the knowledge to be represented.
- The student should be able to analyze whether conclusions derived from a knowledge representation are correct and complete.

Communication:
- The student should be able to explain how logic can be used as a tool for analyzing a knowledge representation problem.
- The student should be able to explain issues involved in the handling assumptions in a knowledge-representation.

Learning skills:
- The student should be able to study autonomously the literature describing the applications of logics for knowledge representation and reasoning.

Study material: Syllabi.

Recommended Literature: A syllabus and scientific literature.

Examination: Written exam at the end of the course. A bonus of 1.0 point can be earned by a series of bonus assignments.

ECTS: 4

Parallel Programming (KEN3235)

Examiner: Dr. Christian Terboven

Prerequisites: Procedural Programming, Objects in Programming, Data Structures and Algorithms.

Description: Parallel programming introduces the students to the paradigm of parallel computing on a computer. Nowadays almost all computer systems include so-called multi-core chips. Hence, in order to exploit the full performance of such systems one needs to employ parallel programming.

This course covers shared-memory parallelization with OpenMP and java-Threads as well as parallelization with message passing on distributed-memory architectures with MPI. The course starts with a recap of the programming language C followed by a brief theoretical introduction to parallel computing. Next, the course treats theoretical aspects like MPI communication, race conditions, deadlocks, efficiency as well as the problem of serialization. This course is accompanied by practical labs in which the students have the opportunity to apply the newly acquired concepts. After completing this course students will be able to write parallel programs with MPI and OpenMP on a basic level, and deal with any difficulties they may encounter

Knowledge and understanding: Students recall the basic concepts for parallel programming and recognize important parallelization patterns.

Applying knowledge and understanding: Students are able to write parallel software code using MPI, OpenMP, and Java Threads.

Communication: Students are able to explain why a specific pattern is adequate for a given problem.

Learning Skills: Students are able to study autonomously the literature describing parallel programming in order to comprehend important details and problems of the field.

Study material: Course notes and several codes will be provided online.
Recommended literature: Parallel programming with MPI; Peter Pacheco; Morgan Kaufmann (1996); (a very early revision is available online)

Exam: Written exam.

ECTS: 4

Large Scale IT and Cloud Computing (KEN3239)

Examiner: Dr. Marius Politze & Dr. Thomas Eifert

Desired Prior Knowledge: Procedural Programming, , Databases

Prerequisites: none

Description: The course offers a comprehensive introduction to the field of scalable IT systems, so-called “Big IT”, and cloud computing. After a technical introduction to the available methodologies of setting up and running scalable systems, use cases are presented. These use cases emphasize the correlation of the processes and requirements of large institutions and possible technical solutions. A special focus is put upon the question which technological platform is best used for which use case as well as process aspects of scaling. Security aspects specific to cloud computing are discussed along the use cases. Cloud computing, as a special case of scalable IT, is discussed in detail. Different cloud providers are presented and evaluated in the context of university requirements, i.e. requirements posed by research and teaching processes.

Knowledge and understanding: students acquire an overview of existing technologies for scalable systems, and specific security requirements for the different use cases.

Applying knowledge and understanding: Students are able to understand scalability and are able to set up and use a scalable IT system. In addition, students are able to evaluate high scalable IT solutions in terms of benefits and security risks.

Making judgements: Students are able to analyze the requirements of a specific use case and can decide which technology is best used for that case of application.

Communication: students are able to communicate about scalable IT systems and specific security requirements.

Learning skills: Additionally, students are able to analyse the interdependencies between large organizations, processes and IT solutions - taking into account security-related aspects - and to design suitable solutions using cloud offerings.

Study material: Lecture notes

Exam: Assignments and Project

ECTS: 4

Introduction to Bio-Informatics (KEN3440)

Examiner: Dr. Rachel Cavill.

Desired Prior Knowledge: Procedural Programming, MatLab.

Prerequisites: None.

Description: This course presents a general introduction to the fundamental methods and techniques of bioinformatics in biomedical and biological research. The objective is that the students will acquire a general understanding of bioinformatics methods at the algorithmic level and will therefore be able to read and understand publications in this field, and – to some extent – apply their knowledge to concrete biological problems. This relates to the major areas of bioinformatics like sequence alignment, phylogenetic analysis, gene finding, and omics data analysis. This course consists of a series of closely related lectures and computer classes, based on relevant case-studies using real data. In the lectures the main theoretical aspects are presented. In the computer practicals, the students work to analyse real data using the techniques they have encountered. By extensively exploring the case study, the students acquire a thorough understanding of the subject.

Knowledge and understanding: Students should be able to perform common analyses on both sequence data and numeric data from omics experiments. This includes sequence alignment, building phylogenetic trees, applying hidden Markov models, detecting differentially expression and performing pathway analysis.

Applying knowledge and understanding: For all the above topics students should be able to demonstrate the algorithms on paper with simple examples and apply the algorithms appropriately on realistic datasets using a computer.

Making judgements: After successful completion of the course, students will be able to judge the use, quality, and correctness of different bioinformatics algorithms and results.

Communication: After successful completion of the course, students will be able to judge the use, quality, and correctness of different bioinformatics algorithms and results.

Learning Skills: After successful completion of the course students will be able to independently read bioinformatics literature to further their knowledge.

Study material: Introduction to Computational Genomics, A Case Studies Approach, Nello Cristianini, Matthew W. Hahn, Cambridge University Press, 2006, Hardback and Paperback (ISBN-13: 9780521856034 | ISBN-10: 0521856035).

Exam: Written exam (50%) + assignments (50%).

ECTS: 4

Project 3-1 (KEN3300)

Examiners: Dr. Katharina Schneider and Dr. Rico Möckel

Prerequisites: Project 2-1.

Description: Project 3-1 consists of two distinct paths: projects at the Department of Advanced Computing Sciences with focus on university research and DSAI/BSSC/BISS projects with focus on applied research proposed by companies that are affiliated with BSSC (Brightlands Smart Service Campus). The DSAI/BSSC/BISS projects are facilitated in cooperation with BISS (Brightlands Institute for Smart Society). In the first week of period 1, students indicate their preference by ranking these projects. Groups are created by means of an algorithm that minimizes regret and allocates students to their most preferred options.

About the projects at the Department of Advanced Computing Sciences: Students work in small groups, guided by teachers of the subjects concerned and by the tutors. During the project, students apply their knowledge in data science, and artificial intelligence to robotic and other intelligent and autonomous systems. Depending on their chosen specialization within their project group, students study and search for solutions in at least one, typically in multiple of the following fields: control, machine learning, computer vision, signal processing, human-computer/robot interaction, multi-agent and distributed systems, optimization, data visualization as well as modelling and simulation.

About the DSAI/BSSC/BISS projects: Students participate in small groups and receive guidance from a tutor, a teacher with knowledge of the subjects concerned, and a content expert from the company. Furthermore, the students receive business related skills such as creating business presentations from a teacher at BISS. Students learn how to apply their knowledge in data science, and artificial intelligence to solve real-world problem that arise in a professional environment, and how to interact with a client from the industry.

Knowledge and understanding: Students gain the opportunity to specialise in a variety of methods in artificial intelligence and data science. Students obtain knowledge in designing, building, and programming complex system solutions. Students learn how to conduct professional research by using appropriate research methodologies.

Applying knowledge and understanding: Students learn to apply complex methods from artificial intelligence and data science to real-world applications. Students furthermore learn how to apply domain-specific tools to solve real-world challenges proposed by companies or driven by societal needs.

Making judgements: Students learn how to analyse complex challenges and to judge whether these can be targeted with methods from artificial intelligence and data science. Students further learn how to select promising solutions and approaches in a methodological way.

Communication: Students learn and practice how to communicate with stakeholders to understand their requirements and specific needs regarding complex real-world challenges. Students further learn how to communicate their plans in targeting challenges, how to communicate intermediate and final results and how to obtain feedback in a structured way to further enhance their problem-solving approaches.

Learning skills: After competing the project course, students will be able to autonomously and critically reflect on their knowledge and skills for solving complex real-world challenges, to identify gaps in their knowledge and skills and to identify professional literature and other resources to fill these gaps.

Study material: Project book and project descriptions. Additional study material will be provided for the individual project topics.

Assessment: The project will be assessed based on the (scientific) insights, developments and professional documentation generated by the student groups.

Skill classes:
- Group CV Check: during this class, you will receive tips and feedback on how to write a professional résumé (i.e. Curriculum Vitae).
- Networking skills: during this class, you will learn hands-on tips to build an interesting network to support you in your search for a job or internship.

Both of the above classes are provided by instructors of the UM Career Services.

ECTS: 6

Period 3.4

Operations Research (KEN3410)

Examiner: Dr. Steven Kelk & Dr. Barbara Franci

Desired Prior Knowledge: Linear Programming.

Prerequisites: None.

Description: Operations Research (OR) is concerned with the best way to assign scarce resources to competing activities. It is for this reason an important branch of mathematics that is widely used in industry to support economically efficient decision making, but also in other application areas where discrete or stochastic optimization has a central role. In this course we will explore a number of themes both within deterministic OR (where all the problem data is known at the beginning) and stochastic OR (decision problems involving uncertainty and randomness). Themes within deterministic OR include the network simplex method (used for solving minimum-cost flow problems), integer linear programming and non-linear programming. Stochastic themes include queuing systems, Markov chains and Markov decision problems. As background students will be introduced to the methodological similarities and differences between OR and data science.

Knowledge and understanding: Students can recognize, classify and distinguish some of the major types of OR models, i.e. transportation and network optimization models, integer and non-linear programming, Markov chains and Markov decision problems, queueing models.

Applying knowledge and understanding: Students can apply a wide variety of algorithms to calculate solutions for problems of the types mentioned above. Students will be able to translate simple real-world/industrial optimization problems into a format suitable for (variously) the transportation simplex, network simplex and integer linear programming.

Making judgements: Students can explain advantages and disadvantages of different models and algorithms. They are able to judge the correctness of solutions presented.

Communication: Students can explain and defend their solution methods.

Learning skills: Students will be able to critically reflect upon the scope and limitations of the learned models, and be able to identify follow-up literature describing paradigms, models and algorithms that go beyond the scope of the course.

Study material: Hillier & Lieberman (2010 or 2015): Introduction to Operations Research (9th or 10th edition). McGraw Hill, ISBN 978-007-126767-0 or ISBN 9781259162985. Support for the 11th edition is forthcoming.

Recommended literature: None.

Exam: Written exam, worth 100% of the credit.

ECTS: 4

Intelligent Systems (KEN3430)

Coordinator & examiner: Dr. ir. Kurt Driessens

Tutor(s): t.b.d.

Prerequisites: None.

Description: The course offers an introduction to intelligent systems, their components, design issues and possible development paths. Based on the metaphor of a computational agent (that is, a software program or a robot which acts and interacts flexibly and autonomously in order to achieve some goal), basic concepts and methods from agent technology are discussed. Topics covered are the concept of artificial intelligence, expert systems, characteristics of an agent and agent architectures, agent cooperation and competition among agents, behaviour-generation and -learning with the added complexity of a multi-agent environment, agent oriented world views and possible future paths to general artificial intelligence. An emphasis is made on the complexity of interacting systems, both between different agents, but also between the subsystems of a single agent. In the practical part of the course, the students build up their experience with the implementation of a number of different types of agents.

Knowledge and understanding: Students are able to compare and discuss benefits and drawbacks of a number of different agent technologies. They can also explain the complexities arising from interactions between multiple techniques within a single agent, and the interactions between agents and systems.

Applying knowledge and understanding: Students will be able to implement of a number of different types of agents architectures and agent-subsystems and agent behavior generation techniques.

Making judgements: The student will be able to judge whether it is beneficial to use intelligent systems technology over other approaches for handling a given problem, and which agent architectures might fit best.

Communication: The student will gain a working knowledge of intelligent system terminology and will learn to motivate his/her choices concerning the application of intelligent technology.

Learning Skills: Students have to reflect upon their knowledge and recognize the need for continued learning as they are confronted with the complexities involved with applying the knowledge gained in their bachelor studies and linking individual techniques into a working system.

Study material: Course slides are shared as a support for the lectures; supplementary material consisting of research papers and book chapters are provided through the student portal.

Assessment: Assessment is based on a 20% grade for daily work and 80% for a written exam at the end of the course.

Exam: The exam consists of a combination of multiple choice questions and open-ended questions.

Additional literature: Artificial Intelligence: A Modern Approach, Russel and Norvig.

ECTS: 4

Project Skills period 3.1 & 3.2:
Group CV Check (online or on-site): during this class, you will receive tips and feedback on how to write a professional résumé (i.e. Curriculum Vitae).

Networking skills (online or on-site): during this class, you will learn hands-on tips to build an interesting network to support you in your search for a job or internship.
Both of the above classes are provided by instructors of the UM Career Services.

Study material: Period book 3.1-3.3. Maastricht University.

Exam: The project will be assessed based on report, product, presentation and project management.

ECTS: 6

Data Analysis (KEN3450)

Examiners: Dr. Jerry Spanakis

Desired Prior Knowledge: Calculus, Linear Algebra, Mathematical Modelling & Simulation, Machine Learning, Introduction to Computer Science 1 and 2.

Prerequisites: None

Description: This course aims at preparing students on how to be a successful "data scientist". The crucial processes of inspecting, cleaning, transforming, restoring and preparing data for modelling are tackled. Different types of data are going to be explored through case studies ("clinics") that a modern "data scientist" has to deal with. Furthermore, several techniques from machine learning and mathematical modelling (multiple regression, classification, tree-based models, dimensionality reduction, etc.) are presented from the data analysis perspective and students learn how to apply these techniques to different types of data. Finally, the cornerstone of data analysis is presented: correct communication of the analysis outcome (storytelling, visualization, etc.).

Knowledge and understanding: Students are able to illustrate and explain data analysis and machine learning techniques with emphasis on modelling, and to give examples of different domains where data analysis can be applied

Applying Knowledge and understanding: Students are able to examine datasets using techniques learned in course, and to experiment with different techniques for data modelling

Making Judgements: After successful completion of the course, students are able to 1) judge the quality of data (of any kind), 2) to justify and rank which techniques should be applied in each problem and 3) to assess results of data analysis process

Communication: Students are able to present the results of different stages of data analysis to specialists and non-specialists and are able to decide on the correct communication medium (scientific, verbal and visual) of the analysis outcome

Learning Skills: After successful completion of the course, students are able to suggest options for tackling different datasets combining verbal, numerical/scientific and visual descriptions, also taking into account the context cases (e.g. business, academic) or the domain of application. Furthermore, students are able to formulate data descriptions based on their characteristics and can suggest options for modelling data and perform basic temporal analysis and dimensionality reduction

Study material: Jupyter notebooks (and limited slides)

Recommended literature: Selected chapters from the following textbooks:
- A. Downey, Think Stats: Exploratory Data Analysis
- James, G., Witten, D., Hastie, T., Tibshirani: An Introduction to Statistical Learning (with Applications in R)
- J. Vanderplans, Data Science Handbook
- S. Skiena, The Data Science Design Manual
- J W. McKinney, Python for Data Analysis
- Chris Albron, Machine Learning with Python Cookbook

Exam: Open-book Digital Exam 70%, Practical assignment 30%

ECTS: 4

Bachelor's Thesis (KEN3500)

ECTS: 18

Bachelor's thesis Data Science and Artificial Intelligence
At the end of the Bachelor's study in Data Science and Artificial Intelligence each individual student has to write a thesis manuscript. This thesis manuscript must be designed as a scientific article of 8 pages using a standard (LaTeX) design. Students are expected to conduct a pro-active and independent research on their topics. This includes the search and reading of related work. The topics must be discussed with the potential thesis supervisor(s) and a research plan must be submitted to and approved by the Board of Examiners as an initial step. The thesis has to be accompanied by relevant attachments and software. Students will present the thesis in a public on-site conference.

This means that a strict submission form will be used. In order to start working on the thesis, a student is required to have obtained at least 140 ECTS (among which are 60 credits of the first year, and 40 ECTS of the second year).

General procedure
Below is an indication for these phases. A bachelor's thesis coordinator will supervise the entire procedure and schedule. *Please note there is also an option to start the trajectory in September. For more information ask the thesis coordinator.*

November
Phase 0: Thesis Topic meeting
Potential topics and research fields will be presented.

January
Phase 1: Topic selection
During the skills class, each student selects a topic (and problem statement) and finds two appropriate prospective thesis examiners. Every student hands in a signed bachelor's project plan to the bachelor's thesis coordinator. If the Board of Examiners approves the thesis plan, the examiners are appointed.
Periods 3.4-5: February - May

Phase 2: Research
In this period every student conducts his/her own research. This will preferably be guided in groups by the thesis supervisor. Further two seminars will be organised during which the students present their progress.

Phase 3: Writing
Parallel to the research, a scientific article is written.

Period 3.6: June
In Period 6, the research is finished and the first versions of the thesis manuscript is expected (first submission). The first thesis examiner will evaluate the thesis manuscript and gives a first reaction within around a week. The second examiner will also evaluate the paper during this week. The second and final submission will take place at the end of the second week of period 3.6 (concrete dates will be announced).

Phase 4: Preparation for presentation
In the second week of period 3.6, the preparation for the final presentation will start for every student individually. The presentations will be created with PowerPoint and have a maximum length of 10 minutes

Phase 5: Presentation
The bachelor's theses will be presented in the third week of period 3.6 in a scientific conference setting at the university. The presentations have a maximum length of 15 minutes per student (including questions). The conference is open to all students and teaching staff from Data Science and Artificial Intelligence and anyone else who might be interested. The final decision on the grade for the bachelor's thesis will be made shortly after the presentations. A special bachelor's thesis coordinator will supervise the phase.

Re-sit: In case the student fails to present his/her work at the Bachelor conference, the student gets one opportunity to defend his/her work at the next bachelor conference. If the student does pass at any of those two conferences, the student has to select a new topic and submit a new thesis plan. For students not finishing at the June Conference there is one re-sit possibility in a Conference at the end of August.

Requirements for the bachelor's thesis project
For the bachelor's thesis, every student has to conduct a short scientific research project. This can be an empirical as well as a theoretical research. The topic for the research project is open, as long as it fits into the Data Science and Artificial Intelligence program.
The department will offer a list of potential research topics. The topic and the research questions have to be approved by the examiners and the Board of Examiners. To this end, the student will create a bachelor's project plan using the form provided by the Board of Examiners. This plan will be signed by the student, the prospective thesis examiners and then handed in to the bachelor's thesis coordinator. It is possible to execute the bachelor thesis project as an external training period. This should be well defined in the bachelor's thesis plan. In this case, the plan should also include the name of the external institute or company, the name of the external supervisor, the size of the project and any agreements about compensation. The plan should also be signed by the external supervisor. In principle, there should be no confidentiality agreements for a thesis, and staff members cannot be expected to commit to these. The external research cannot start before period 3.5 due to courses in period 3.4. When not selecting a topic offered by the department, or when wanting to do a thesis with the involvement of an external party, it is advisable to start the preparations and requesting permission three months in advance. The research needs to be original

in such a way that the thesis supervisor is convinced that this research has not been done before. The research also needs enough depth and still it must be possible to finish it in the set amount of time. Every thesis is an individual work.

Requirements for the bachelor's thesis manuscript

Content aspects
The thesis manuscript describes the cause, research question, approach and results of the research. This has to be done in a clear, structured and scientific manner which includes:

- a clear introduction in which the context and research questions are presented;
- a clear conclusion, based solely on the already used thought out principles and derived results;
- a clear line is shown between problem statement, methods, and the derived results;
- a motivation of the methods followed;
- an adequate description of the methods followed;
- an honest, clear, and concise description of the derived results, if necessary using tables;
- a discussion of the results;
- the usage of relevant and recent literature;
- the correct usage of references;
- the adequate usage of the literature for the reasoning in the thesis manuscript.

Design aspects
The number of pages of the thesis is 8, in the designated LaTeX format, including images and references. This thesis should at least contain:

- title;
- author;
- abstract;
- one or two keywords;
- list of references;
- page numbers.

It goes without saying that the correct scientific references are used for used resources (by using the designated BiBTeX reference style). Images and tables are accompanied by an index and caption. Mathematical formulas, definitions, etc. have to be properly designed and numbered. The start and end of mathematical formulas have to be properly defined.

Language aspects
The thesis manuscript has to be written in Dutch or English, considering correct spelling, syntactical structure of sentences and structure of content in paragraphs. The target audience is fellow Data Science and Artificial Intelligence students. Any jargon and/or abbreviations have to be explained unless they are common knowledge for this audience (e.g. CPU).

Citations
It is allowed to use several short citations with a maximum length of two sentences.
These citations have to be clearly referenced and have to be typographically distinguishable (that is, citations are placed in quotes). Non-allowed citations or missing references will result in an unsuccessful result.

Assessment
The assessment will take place based on the contents and design of the thesis, the presentation of this thesis and the process. The weighing of the various aspects is up to the examiners.

2.4 Curriculum of the First Year of the Bachelor Programme Computer Science

Course year 1:		ECTS
Period 1.1	Introduction to Computer Science (BCS1110) Procedural Programming (BCS1120) Discrete Mathematics (BCS1130) Project 1-1 (BCS1300)	4 4 4
Period 1.2	Objects in Programming (BCS1220) Calculus (BCS1440) Logic (BCS1530) Project 1-1 (BCS1300)	4 4 4
Period 1.3	Project 1-1 (BCS1300)	6
Period 1.4	Linear Algebra (BCS1410) Data Structures and Algorithms (BCS1420) Object-Oriented Modelling (BCS1430) Project 1-2 (BCS1600)	4 4 4
Period 1.5	Databases (BCS1510) Statistics (BCS1520) Algorithmic Design (BCS1540) Project 1-2 (BCS1600)	4 4 4
Period 1.6	Project 1-2 (BCS1600)	6

(*) Project 1-1 will start in period 1.1 and will run until period 1.3; Project 1-2 will start in period 1.4 and will run until period 1.6. The credits for the projects will become available at the end of period 1.3 and period 1.6, respectively. Please see the course description section Project 1-1 and Project 1-2 for more details on the project curriculum. For each period, we will give a short explanation of the various parts. Before the start of each period, the students will receive detailed information about the content, the study material, the teaching form, the schedule, and the examination method.

Period 1.1

Introduction to Computer Science (BCS1110)

Examiner: Dr. Ashish Sai & Dr. Thomas Bitterman

Desired Prior Knowledge: None

Prerequisites: None

Description: The primary goal of Introduction to Computer Science is to introduce fundamental concepts and foster critical skills found throughout the field of computer science. Fundamental concepts include algorithms, computer architecture and hardware, models of computation, computer networks, and operating systems. Critical skills include abstraction, decomposition, pattern recognition, and algorithmic thinking. All concepts and skills are introduced in a lecture setting and explored further in the lab through the development of a wirelessly controlled microcontroller device. At the end of this course, students will appreciate the depth of the field and be prepared for subsequent research and educational activities.

Knowledge and insight: Students will be able to explain the fundamental concepts and how they relate to the broader field of Computer Science. Students will demonstrate awareness of the current trends and developments in Computer Science.

Applying knowledge and insight: Students will analyze and decompose a given computational problem and identify appropriate methods and tools to design a solution. They will apply the fundamental concepts and critical skills to a practical problem requiring a solution that touches on various subfields of Computer Science.
Judgement: The students will be able to recognize and compare different approaches and

techniques available within computer science to solve a given problem.

Communication: Students will communicate about different aspects of computer science using appropriate terminology and format such as pseudo-code and UML diagrams. They will describe solutions to abstract computational problems verbally and in writing. They will also be able to justify their design choices and document them in a traceable manner.

Learning skills: Students will work independently and collaboratively to understand and decompose a computing problem. They will identify and apply techniques and tools in order to solve the problem.

Study material: "An Invitation to Computer Science" by G. Michael Schneider, 8th Edition

Additional literature:
"Computational Thinking for the Modern Problem Solver" by David Riley, Kenny A. Hunt
"Computer Science Illuminated" by Nell B. Dale

Exam: Written exam (75%) + group project (25%)

ECTS: 4

Procedural Programming (BCS1120)

Coordinator: Dr. Enrique Hortal

Examiners: Dr. Enrique Hortal& Charis Kouzinopoulis

Desired Prior Knowledge: None.

Prerequisites: None. It appears as part of the pre-requisites of the second semester project in year 1, both projects of year 2, the year 2 course Databases and the year 3 courses, Parallel Programming and Robotics and Embedded Systems.

Description: The course provides the basics of computer science and computer programming. After a short introduction to computer organization, the principles of programming are presented. The main topics of the course are: data types, variables, methods, parameters, decision structures, iteration, arrays, recursion and a branching application (related to the semester project). Programming skills will be acquired during practical sessions using the object-oriented programming language Java.

Knowledge and understanding: The course offers preliminary methodological and theoretical bases for studying and applying computers and computer programming on which the rest of the curriculum builds.

Applying knowledge and understanding: Whenever a computer system or a programming system has to be designed and implemented the knowledge and insights acquired during the course can be used and applied.

Making judgements: After successful completion of the course, students will be able to judge the quality and correctness of simple non-object-oriented programs.
Communication: The skills acquired during the course will enable students to communicate about

standard programming constructs and algorithmic basics.

Learning Skills: After successful completion of the course, students will be able to formalize, analyse and program solutions to simple software problems.

Study material: Lecture slides, example code and multimedia material that are made available before and after each lecture.

Recommended literature: H. Schildt, Java: A Beginner's Guide, Eighth Edition, ISBN: 1260440214, McGraw-Hill Education

Additional literature: C. Horstmann (2016). Java Concepts (8th Edition). John Wiley & Sons, New York, ISBN: 978-1-1190-5645-4 or C.Horstmann (2012). Big Java Late Objects. John Wiley & Sons, New York, ISBN 978-1-1180-8788-6

Assessment: Closed-book written exam (90%) + Assignments (10%)

ECTS: 4

Discrete Mathematics (BCS1130)

Examiner: Dr. Marieke Musegaas, dr. Otti D'Huys and dr. Stefan Maubach

Desired Prior Knowledge: None.

Prerequisites: None.

Description: In this course, we build a mathematical framework that is based on logic and reason. The main objective of the course is to make students familiar with the language of mathematics. Students will learn how to make sound arguments and to detect where and why certain arguments go wrong. For this purpose, we will discuss the basic principles of logic and, closely related, the basic types of mathematical proofs. In doing so, we will encounter numbers such as integers, natural numbers and real numbers and we shall examine what makes these numbers special. After that, we will use basic logic to discuss, among other things, the following mathematical concepts: infinity, sets, relations, functions, permutations and combinations. Our fundamental tool in all of this is plain common sense. You really do not need your toolbox of mathematical formulas learned in previous studies and neither do you need a calculator. Pen and paper are the basic instruments needed. After completing each topic, exercises will be provided to be completed in class or at home, since mathematics is mainly learned by practising repeatedly.

Knowledge and understanding: Students will be able to read, interpret and manipulate basic mathematical terminology (propositional logic, quantifiers, set theory, relations, functions, and combinatorics). Students will also be able to read and interpret several different types of mathematical proofs and identify whether a purported proof is mathematically sound.

Applying knowledge and understanding: Upon completion of the course students will know how to read, interpret, write and manipulate rigorous mathematical statements using propositional logic, quantifiers, set theory, relations, functions and combinatorics. Students will be able to select, from a range of mathematical tools, which is appropriate to prove or disprove a given mathematical statement, and apply the chosen tools, rigorously and clearly in order to achieve the desired goal.
Making judgements: Students will be able to distinguish between mathematically sound and

unsound statements and defend the rigour of their own mathematical arguments.

Communication: Students will be able to write clear, rigorous and explicit mathematical arguments using standardized mathematical terminology and such that each step in the argument is a logical consequence of earlier steps.

Learning skills: By the end of the course, students will be able to autonomously and critically reflect upon the mathematical correctness of their own arguments.

Study material: A. Chetwynd & P. Diggle: Discrete Mathematics. Butterworth- Heinemann, Oxford, ISBN 9780340610473. Lecture notes will also be provided.

Recommended literature: None

Exam: Closed book written exam

ECTS: 4

Period 1.2

Objects in Programming (BCS1220)

Coordinator: Dr. Thomas Bitterman

Examiners: Dr. Thomas Bitterman, Dr. Evgueni Smirnov

Tutors: Prianikov, Nikola; Barta, Lázár; Doss, Heinz; Bams, Guillaume; Balan, Alexandra; Goldie, Samuel; Buiter Sanchez, Arantxa; Gójska, Maja; Timmermans, Derrick; Straka, Filip; Goffinet, Arthur

Desired prior knowledge: Procedural Programming

Prerequisites: None.

Description: This course is a follow-up to the course Introduction to Computer Science 1. It teaches object-oriented programming in Java. The main topics covered in the course are objects and classes, interfaces and polymorphism, event handling, inheritance, graphic user interfaces, exception handling, and streams.

Knowledge and understanding: After successful completion of the course, students will be able to explain the methodological and theoretical principles of object-oriented programming.

Applying knowledge and understanding: Students will be able to implement basic object-oriented computer programs. They will be able to design and describe simple object-oriented computer systems.

Making judgements: Students will be able to judge the quality and correctness of simple object-oriented programs.

Communication: Students will be able to communicate about object-oriented programming constructs and algorithmic basics.
Learning skills: Students will be able to recognize their own lack of knowledge and understanding

and take appropriate action such as consulting additional material or other sources of help.

Study material: Course notes, slides, and other information made available.

Recommended literature: C. Horstmann (2016). Java Concepts (8th Edition). John Wiley & Sons, New York, ISBN: 978-1-1190-5645-4
C.Horstmann (2012). Big Java Late Objects. John Wiley & Sons, New York, ISBN 978-1-1180-8788-6

Assessment: Written exam (80%) + practical assignments (20%).

ECTS: 4

Calculus (BCS1440)

Examiner: Dr. Otti D'Huys & Dr. Gijs Schoenmakers

Prerequisites: None.

Description: The following subjects will be discussed in Calculus: limits and continuity, differential calculus, integral calculus, sequences and series, introduction to differential equations, introduction tomultivariable calculus. In addition to the main facts and concepts, problem-solving strategies will be discussed. Both the intuition behind the concepts and their rigorous definitions will be presented along with simple examples of formal mathematical proofs.

Knowledge and understanding: Students can define, write and explain key facts and concepts involving limits and continuity, can interpret and solve differential calculus, integral calculus, sequences and series, first-order linear differential equations problems, and understand the basics of multivariable calculus.

Applying knowledge and understanding: Students are able to solve problems applying the concepts learned in the course, using standard problem-solving strategies.

Making judgements: Students are able to analyse a simple problem within the course content and justify the solution methodology they choose. They can summarize this methodology mathematically.

Communication: Students are able to explain their solution strategy in written form and defend their solution strategy in discussion with others

Learning Skills: After successful completion of the course the students will be able both to solve standard problems (constructing graphs of functions, finding extrema of functions, computing limits, summing infinite series etc.) and to apply their knowledge in solving and analysing more complex problems (e.g. in analysis of numerical algorithms).

Study material: Calculus, a complete course, any edition, by R.A. Adams, Addison Wesley Longman and materials provided during the lectures.

Exam: Intermediate bonus assignments and a final written exam.

ECTS: 4

Logic (BCS1530)

Coordinators: Dr. Tjitze Rienstra & Dr. Stefan Maubach

Examiners: Dr. Tjitze Rienstra, Dr. Stefan Maubach & Dr. Nico Roos

Description: This course deals with three logical systems, namely propositional logic, first-order predicate logic and dynamic logic. The course covers notation systems, syntax and semantics, valid consequences, deduction, semantic tableaux, and proof systems.

Knowledge and understanding: Students need to get accustomed to the fundamental concepts of mathematical logical systems (propositional logic and predicate logic) to able to describe information in a logical framework and to reason and prove correctly. Students will get accustomed to the basic concepts of some advanced logical systems (dynamic logic and Hoare logic).

Applying knowledge and understanding: Student will apply the reasoning and proof methods learned to small-scale problems and some more complex situations.

Making judgements: Students will learn to judge how to reason correctly using mathematical proofs and how to judge which logical system is suitable to solve the problem at hand.

Communication: The chosen syntax of the logical language used must be easily understandable by peers and other experts the logical proofs given must be correct, concise and easily understandable.

Learning skills: Having learned basic logical concepts and reasoning techniques the students are able to apply them to larger-scale problems.

Study material: Johan van Benthem, Hans van Ditmarsch, Jan van Eijck, Jan Jaspars, Logic in Action. Edition of February 2014 or later. This is a freely available e-book. Check your Canvas for the link.

Exam: Written exam.

ECTS: 4

Project 1-1 (BCS1300)

Coordinator: Dr. Martijn Boussé

Description: Students work on a project assignment in small groups of six to seven students. The group composition stays the same for the whole project and is announced shortly before the project opening in period 1.1. The students are guided through the project by tutors (for project management) and mediators (for team dynamics). The project assignment is divided into three subtasks (one per period) and is strongly related to the course content from period 1.1 and 1.2. In period 1.1, after receiving the assignment for the whole project at the end of week 4, the students can start working on the project, and work full-time on the project in week 8 after the exams. In period 1.2, the students continue working on the projects in parallel to the other courses of that period. In period 1.3, the students work two weeks full-time on the project. The students meet their tutor about 3 times per period. A plenary Q&A with the examiners will be organized in period 1.1 and 1.2. The students will engage in several feedback and graded moments with the examiners during the project.

Knowledge and understanding: Interpret constraint-satisfaction problems arising in practice and translate this to discrete-mathematical algorithmic models capable of solving the problem. Gain insight into practical use of basic software design and development principles. Recognise and relate user-computer interactions to concepts from graphics and user-interface frameworks. Strengthen knowledge of basic algorithms and methods for efficiently solving constraint-satisfaction problems arising in applied mathematics (especially: discrete mathematics) and artificial intelligence.

Applying knowledge and understanding: Design an answer strategy for scientific questions using analytical thinking and logical reasoning. Translate discrete-mathematical algorithmic models to software code. Implement software to efficiently solve constraint-satisfaction problems arising in applied mathematics (especially: discrete mathematics) and artificial intelligence by finding, designing and applying appropriate algorithms. Formulate computational experiments, and analyse and interpret the results. Apply basic design and development principles in the construction of software systems. Use existing software application frameworks for graphics and user interfaces. Use tools for software project management such as version control systems and issue trackers. Identify project goals, deliverables, and constraints. Plan and chair meetings. Create minutes for meetings. Work in a team such that the workload is balanced. Plan teamwork by setting deadlines and distributing tasks.

Making judgements: Evaluate different mathematical and computational models with respect to their suitability, efficiency and correctness for a specific task. Elicit and evaluate relevant scientific background information. Evaluate the group's progress during the project.

Communication: Give a clear and well-constructed presentation, including a demonstration of the product, and with appropriate use of illustrations and/or videos. Offer and respond to questions on and constructive criticism of presentations. Write a project report according to the structure of an academic article. Submit arguments in exact sciences, with appropriate use of formulae and figures. Cite published sources in the project report according to the academic guidelines. Structurally inform stakeholders on project progress. Effectively communicate with project group members about task division, planning and project deadlines. Effectively communicate with group members by listening to others' ideas; be contactable and include others in the discussion. Cooperate in a group to reach a consensus view. Give constructive feedback to team members. Communicate in the English language.

Learning skills: Reflect on one's own academic abilities and functioning in a team.

Study material: Project manual project 1-1, Maastricht University

Assessment:
The final grade will be composed of a project grade and a skill class grade. The project grades consists of several components such as project management, deliverables, presentation, and peer feedback. The skill class grade will depend on the total number of passed skill classes. (NG for the project is given if a student failed more than 2 skill classes).
Students not complying with attendance and participation requirements during the project meetings or examination moments may not be allowed to attend examination moments or may receive an NG.

Skill classes:
Students will engage with a series of skill classes that prepare and support them for project work such as project management, (pair) programming, team dynamics, and communication.

ECTS: 6

Period 1.4

Linear Algebra (BCS1410)

Examiner: Dr. Marieke Musegaas, dr. ir. Philippe Dreesen & dr. Steve Chaplick

Desired Prior Knowledge: None.

Prerequisites: None.

Description: This course introduces the fundamental concepts of linear algebra, and examines them from both an algebraic and a geometric point of view. First, we address what can be recognized without doubt as the most frequently occurring mathematical problem in practical applications: how to solve a system of linear equations. Then we discuss linear functions and mappings, which can be studied naturally from a geometric point of view. Vectors spaces are then introduced as a common framework that brings all themes together. Next, we shift from the geometric point of view to the dynamic perspective, where the focus is on the effects of iterations (i.e., the repeated application of a linear mapping). This involves a basic theory of eigenvalues and eigenvectors, which have many applications in various branches of science as for instance in problems involving dynamics and stability, in control theory, and in optimization problems found in data science. Key concepts in the course are vectors, matrices, systems of linear equations, eigenvalues, eigenvectors, linear transformations, and orthogonality. The software package Matlab is introduced in the accompanying computer classes, where emphasis is put on the application of linear algebra to solve real world problems.

Knowledge and understanding: Students are able to recognize and explain the fundamental concepts of Linear Algebra: systems of linear equations, vectors and vector spaces, basis and coordinates, matrices and matrix-vector computations, linearity and orthogonality, linear independence, rank, fundamental spaces (row space, column space, and null space), determinants and invertibility, eigenvalues and Eigen spaces, diagonalization.

Applying knowledge and understanding: Students are able to analyse a linear algebra problem from both an algebraic and a geometrical point of view. Students can solve systems of linear equations, compute determinants and rank, compute eigenvalues and Eigen spaces, make use of complex numbers, diagonalizable matrices, and perform change of coordinates.

Making judgements: Students are able to look at the same problem from different angles and to switch their point of view (from geometric to algebraic and vice versa).

Communication: Students are able to motivate both from an algebraic and a geometric point of view the solution set of a system of linear equations, the linear independence and orthogonality of a set of vectors, the linear transformation between two coordinate systems, the fundamental spaces associated with a matrix, the invertibility of a matrix, and the diagonalization of a matrix in terms of the properties of its eigenvalues and eigenvectors.

Learning skills: Students have acquired the skills to autonomously recognize elements of practical problems, which can be addressed and solved with linear algebra, and use Matlab to solve larger scale problems.

Study material: David C. Lay, Linear algebra and its applications, 6th ed., Pearson, ISBN: 978-1-292-35121-6.

Recommended literature: None

Exam: Closed book written exam

ECTS: 4

Data Structures and Algorithms (BCS1420)

Coordinator: Tom Pepels, M.Sc.

Examiners: Dr. Francesco Barile, Tom Pepels, M.Sc., & Dr. Bastian Küppers

Tutor(s): TBA

Desired prior knowledge: Programming in Java, Procedural Programming (BCS1120), Objects in Programming (BCS1220)

Prerequisites: None.

Description: The Data Structures and Algorithms course introduces the students to the design and application of data structures and algorithms. Abstract datatypes will be used as a central topic in this course. Together with the basic abstract data types such as trees, lists, and graphs, the associated algorithms and their complexity are discussed. The differences between the best, expected, and worst behaviour of an algorithm is explained. Supported by the concepts of complexity bounds and big O notation complexity is illustrated on several algorithms such as search, and string & graph algorithms. After completing this course, students will be able to determine the appropriate data structures and algorithms for simple problems.

Knowledge and understanding: Students will acquire a thorough understanding of both fundamental and complex data structures—ranging from arrays and linked lists to trees and graphs—alongside the principles of algorithm design, such as recursion, sorting, and graph algorithms. The curriculum emphasizes the importance of complexity analysis, teaching students to evaluate algorithm performance using Big O notation and other measures. Students will explore various algorithmic strategies, including dynamic programming and greedy algorithms, to solve computational problems efficiently.

Applying knowledge and understanding: Students will directly apply theoretical concepts through hands-on coding tutorials, designing and implementing algorithms to address specific problems. The primary learning goals include mastering the selection and application of appropriate data structures for optimizing software performance, conducting complexity analysis to evaluate and improve algorithm efficiency, and developing solutions for software development challenges. This approach aims to enhance students' problem-solving skills and prepare them for advanced computational tasks in their academic and professional futures. By the end of this course section, students will have gained experience in applying theoretical knowledge to practical scenarios, demonstrating their ability to navigate complex problems and develop efficient, effective solutions.

Making judgements: Students are tasked with developing the ability to critically assess the efficiency and effectiveness of different data structures and algorithms in solving computing problems. This involves comparing various algorithmic approaches based on their time and space complexities, understanding the trade-offs involved in algorithm selection, and justifying the choice of specific data structures for given scenarios.

Communication: Students will be able to explain how data structures and algorithms are to be included in program designs.

Learning skills: Students are encouraged to develop autonomous learning habits and critical thinking abilities. The focus is on fostering the capacity to independently acquire new computational techniques, adapt to evolving programming paradigms, and apply problem-solving strategies in unfamiliar contexts.

Study material: The course follows the Algorithms Fourth Edition book. Next to the book, weekly lecture videos and short introduction videos to key topics are provided.

Exam: 'Closed Book' written exam

Recommended literature: Sedgewick and Wayne (2011) Algorithms Fourth Edition. Addison Wesley. ISBN: 978-0321573513

Additional literature: A Y Bhargava (2016). Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People. Manning. ISBN: 978-1617292231

ECTS: 4

Object-Oriented Modelling (BCS1430)

Examiner: Dr. Ashish Sai, Dr. Yuquan Wang

Desired Prior Knowledge: Procedural Programming, Objects in Programming.

Prerequisites: None.

Description: This course introduces students to the design and analysis aspects of object-oriented programming. Software construction for real world applications has inherent complexities both in terms of designing and maintaining it. In this course, the students will learn how to model a real-world problems in an object-oriented programming context using tools like Unified Modelling Language (UML). Students will also learn techniques such as structural, behavioral and creational design patterns, GRASP principles to create modular, flexible and reusable software. After completing the course, the students would have gained practical experience in problem formulation, decomposition (analysis) and solution building (design) using object-oriented modelling techniques.

Knowledge and insight: Students understand the principles and practices of object-oriented programming with a focus on design and analysis. Students can explain the use of tools such as UML for modelling software systems. Students also understand the use of design patterns and GRASP principles in improving software quality.

Applying knowledge and insight: After the course, students can critically analyze software models, designs and implementations. Students will be able to use UML diagrams to represent software requirements, system architecture, class structure, system behavior and interactions. Students will also be able to implement software solutions using appropriate design patterns and GRASP principles in an object-oriented language such as Java.

Making Judgements: The students can judge the suitability of different object-oriented modelling techniques, design patterns and GRASP principles for different software development problems. Students can also evaluate the trade-offs between various design alternatives in terms of complexity, modularity, flexibility and reusability. Students can critically assess the quality of their own and others' software designs and implementations using appropriate criteria and metrics.

Communication: Through the course, students become able to communicate effectively with different stakeholders involved in software development using appropriate techniques. Students can present their software designs and implementations using UML diagrams and documentation.

Learning skills: Students develop independent learning skills to keep up with the evolving trends in object-oriented design and analysis. Students also practice how to reflect on software design and implementation in a professional manner.

Study material: "Applying UML and Patterns" by Craig Larman

Additional literature:

- "Clean Code: A Handbook of Agile Software Craftsmanship" by Robert Cecil Martin
- "Design Patterns: Elements of Reusable Object-Oriented Software" by Erich Gamma et al.

Exam: Written exam (70%) + practical assignments (30%)

ECTS: 4

Period 1.5

Databases (BCS1510)

Examiner: Dr. Tony Garnock-Jones

Desired prior knowledge: Procedural Programming, Objects in Programming, Data Structures and Algorithms, Software Engineering.

Prerequisites: None.

Description: This course covers the use of (relational) databases and data modelling with the goal of writing (distributed) data-intensive software applications. Specifically, students will learn to use the Structured Query Language (SQL) to manipulate data to develop data-models that are Atomic, Consistent, Isolated and Durable. Moreover, the course covers alternative (distributed) data-storage methods and object persistence techniques such as NoSQL. During the course, students will learn to use different database management systems and how to use them to build software.

Knowledge and understanding: Students will be able to describe the basic concepts of databases, explain the fundamental concepts of database management systems, query languages, data modelling and database programming.

Applying knowledge and understanding: Students will be able to explain the proper database design based on system requirements, indicate possibilities and limitations of database types. In addition, students will be able to combine software architectures to design and construct a database application.

Making judgements: Students will be able to analyze and justify a practical database problem, examine different approaches, and refine database models based on use cases. Moreover, they can make improvements to existing database designs, reflect on certain solutions of the databases design and implementation, and assess the correctness of the database model.

Communication: Students will be able to summarize the basic entities and relationships involved in persistent data, and communicate with developers, database managers and users on proper database design and interfacing.
Learning skills: Students will be able to identify and understand follow-up literature, beyond the teaching material of the course.

Study material: Alan Beaulieu, 2020. Learning SQL, (3rd ed.). O'Reilly Media, Inc.

Recommended literature: Martin Kleppmann, 2017. Designing Data-Intensive Applications. O'Reilly Media, Inc.

Exam: Written exam (75%) + practical assignment (25%)

ECTS: 4

Statistics (BCS1520)

Examiner: Dr. Anirudh Wodeyar

Prior Knowledge: Calculus and Discrete Mathematics.

Prerequisite: None.

Description: Statistics introduces the student to the main concepts of both probability theory and statistics. With respect to probability theory, students learn how to make use of random variables to extract the probability distribution of an experiment. Additionally, topics such as expectation, standard deviation, and independence will be discussed. The statistics part of the course discusses basic statistical topics such as the central limit theorem, verification of hypotheses, and confidence intervals.

After completing this course students will have obtained an overview of commonly seen probability distributions, as well as several statistical procedures. Additionally, the student will be able to deal with problems that involve probabilities and determine an outcome for such problems (e.g., the expected outcome).

Knowledge and understanding: Students will obtain an overview of the most relevant and frequently used probability distributions as well as several statistical procedures

Applying Knowledge and understanding: Students can calculate probabilities, expectations, variances and related quantities in a wide variety of probabilistic experiments; estimate statistical quantities; perform several statistical tests to extract information.

Making judgements: Students can analyse probabilistic experiments, critically analyse statistical inferences and decide whether to accept or reject statistical hypotheses. Choose, motivate, and contrast methods for statistical analysis.

Communication: Students can explain how to solve problems involving probabilities and/or statistical procedures.

Learning Skills: Students can reflect on the use of probability theory and statistics in other domains in order to increase one's knowledge.

Study material: Probability and Statistics for Engineers and Scientists (ninth edition) by Walpole, Myers, Myers, and Ye, ISBN 9781292161365. Additional material from lectures, tutorials and other resources.

Recommended literature: None

Assessment: 20% homework assignments and 80% written final exam).

ECTS: 4

Algorithmic Design (BCS1540)

Coordinator: Dr. Thomas Bitterman

Examiners: Dr. Thomas Bitterman & Dr. David Mestel

Tutors: Arslan, Ayse; Schmitz, Britt; Verşebeniuc, Dumitru; Vroom, Thomas; Spišák, Matej; Weindorfer, Eric; Straka, Filip; Anerdi, Giacomo

Desired prior knowledge: Data Structures and Algorithms

Prerequisites: None

Description: Algorithmic Design formalizes the main algorithmic paradigms and techniques including greedy and divide-and-conquer strategies, dynamic programming, multi-dimensional searching, computational geometry, linear programming, randomization, and approximation algorithms. It familiarizes students with amortization and NP-completeness. After completing the course, students will be expected to show good design principles and adequate skills at reasoning about the correctness and complexity of algorithms.

Knowledge and understanding: Students can give examples of run-time complexity classes for well-known algorithms. Students can differentiate between different algorithm designs from examples. Students can describe some advanced algorithms and highlight their properties. Students know the complexity classes P and NP.

Applying knowledge and understanding: Students will be able to derive the run-time complexity of select algorithms. Amortization can be applied when deriving algorithm complexity.

Making judgements: Students will be able to judge the quality and correctness of simple object-oriented programs.

Communication: Students will be able to communicate about object-oriented programming constructs and algorithmic basics.

Learning Skills: Students will be able to recognize their own lack of knowledge and understanding and take appropriate action such as consulting additional material or other sources of help.

Study material: Course notes, slides, and other information made available.

Assessment: Written exam (80%) + practical assignments (20%).

Recommended literature: C. Horstmann (2016). Java Concepts (8th Edition). John Wiley & Sons, New York, ISBN: 978-1-1190-5645-4 or C. Horstmann (2012). Big Java Late Objects. John Wiley & Sons, New York, ISBN 978-1-1180-8788-6

Additional literature: H. Schildt, Java: A Beginner's Guide, Eighth Edition, ISBN: 1260440214, McGraw-Hill Education.

ECTS: 4

Project 1-2 (BCS1600)

Coordinator: Dr. Otti D'Huys

Prerequisites: In order to participate in this project the student has to have passed two out of four courses from the set: Discrete Mathematics, Calculus, Procedural Programming and Objects in Programming.

Description: Students work on a project assignment in small groups of about seven students. The group composition stays the same for the whole project and is announced before the project opening in period 1.4. The students are guided through the project by a fixed tutor. The project assignment is related to the content of the courses from year 1. In period 1.4, after receiving the assignment for the whole project at the end of week 5, the students start working on the project in parallel to their courses. They meet their tutor approximately once every week. In period 1.6, the students work three weeks full-time on the project and meet their tutor twice a week.

At the beginning of period 1.5, the students hand in a planning, along with a short summary about the work completed so far, and receive feedback from the examiners. By the end of period 1.5 the students have a midway evaluation as formative assessment, and hand in a draft report. In period 1.6, they submit a final report on their project and attend a final examination.

Knowledge and understanding: Interpret the meaning of mathematical models of real-world processes. Gain insight into practical use of software design and development principles. Recognise and relate user-computer interactions to concepts from graphics and user-interface frameworks. Strengthen knowledge of basic algorithms and methods for specific problems in computer science.

Applying knowledge and understanding: Students will be able to design an answer strategy for scientific questions using analytical thinking and logical reasoning and to translate mathematical models to software code. Furthermore, students will be able to implement software to solve problems in applied mathematics by applying numerical methods and artificial intelligence algorithms, formulate computational experiments, and analyse and interpret the results, apply design and development principles in the construction of software systems and use existing software application frameworks for graphics and user interfaces. Even more so, students will learn to use tools for software project management such as version control systems and issue trackers, identify project goals, deliverables, and constraints. Lastly they will learn how to plan and chair meetings, create notes for minutes, work in a team such that the workload is balanced and plan teamwork by setting deadlines and distributing tasks.

Making judgements: Students will learn to evaluate different mathematical and computational models with respect to their suitability, efficiency and correctness for a specific task.

Communication: Students will be able to give a clear and well-constructed presentation, including a demonstration of the product, and with appropriate use of illustrations and/or videos, to offer and respond to questions on and constructive criticism of presentations. Furthermore, they will learn to write a project report according to the structure of an academic article, submit arguments in exact sciences, with appropriate use of formulae and figures. They learn to cite published sources in the project report according to the academic guidelines. Additionally, students will learn to structurally inform stakeholders on project progress and effectively communicate with project group members about task division, planning and project deadlines, effectively communicate with group members by listening to others' ideas; be contactable include others in the discussion. It will be important to cooperate in a group to reach a consensus view, communicate in the English language, elicit and evaluate relevant scientific background information.

Learning skills: Reflect on one's own academic abilities and functioning in a team.

Study material: Project manual project 1-2, Maastricht University.

Assessment:

$$FinalGrade=0.9 \cdot (ProjectGrade \vee IndividualGrade)+skillClassGrade$$

The *individualGrade* is given due to either outstanding or not enough contribution of a student to the project. By passing skill classes, the students can get a reward called *skillClassGrade*, which is 1 if the students passes sufficiently many components of all skill classes, 0.5 if the students passed most components, and 0 if the students fails for a critical amount of the skill class tasks. Failing all components of more than 2 skill classes will lead to an NG in the project. Missing mandatory project events such as project meetings and examination moments will lead to a reduction of the grade or even to receiving an NG for the project.

Skill classes: There will be skill classes on the following topics (each with components spread over periods 1.4-1.5)

Information Research: Systematic Literature Search

These skill classes will give the students an introduction to which databases, search strings and settings can be used to systematically search for literature, and are guided in drafting a search plan for the relevant literature in the project.

Team Dynamics: The team dynamics workshops aim to provide the students with a deeper awareness, insight and practice in effective team collaboration & co-creation. In a later stage, students evaluate the team collaboration and communication by means of interactive exercises.

Academic Writing: In the project skills components you will explore the key structure of your report, as well as key points of Academic Writing at Maastricht University. Areas of focus include: structure of paper; linguistic aspects of writing in English, presenting information logically and citation and reference procedures.

ECTS: 6

2.5 Curriculum of the second year of the Bachelor Programme Computer Science

Year 2		ECTS
Period 2.1	Computer Networks (BCS2110) Introduction to Artificial Intelligence (BCS2120) Intelligent User Interfaces (BCS2130) Elective Module Project 2-1 (*): • **Either:** M2-1: Intelligent Interaction (BCS2710) - Project 2-1: Human-Computer Interaction • **Or:** M2-1: Artificial Intelligence and Machine Learning (BCS2720) - Project 2-1: Adaptive Systems	4 4 4 10 10
Period 2.2	Software Engineering and Architectures (BCS2210) Principles of Programming Languages (BCS2220) M2-1: Intelligent Interaction (BCS2710) (*): • Image and Video Processing • Elective Module Project 2-1 Human-Computer Interaction M2-1: Artificial Intelligence and Machine Learning (BCS2720) (*): • Machine Learning • Elective Module Project 2-1 Adaptive Systems	4 4
Period 2.3	Elective Module Project 2-1: • Project 2-1 Human-Computer Interaction (BCS2710) • Project 2-1 Adaptive Systems (BCS2720)	 10 10
Period 2.4	Embedded Programming (BCS2410) Computer Security (BCS2420) Parallel Programming (BCS2430) Elective Module Project 2-2 (*): • **Either:** M2-2: High Performance Computing (BCS2730) • **Or:** M2-2: Cybersecurity & IoT – Information Security (BCS2740) • **Or:** M2-2: Cybersecurity & IoT – Ubiquitous Computing & IoT (BCS2750)	4 4 4 10 10 10
Period 2.5	IT Management and Privacy (BCS2510) Numerical Methods (BCS2540) M2-2: High Performance Computing (BCS2730) (*): • High Performance Computing • Elective Module Project 2-2 High Performance Computing M2-2: Cybersecurity & IoT – Information Security (BCS2740) (*): • Information Security • Elective Module Project 2-2 Cybersecurity & IoT M2-2: Cybersecurity & IoT – Ubiquitous Computing & IoT (BCS2750) (*): • Ubiquitous Computing & IoT • Elective Module Project 2-2 Cybersecurity & IoT	4 4
Period 2.6	Elective Module Project 2-2: - Project 2-2 High Performance Computing (BCS2730) - Project 2-2 Cybersecurity & IoT – Information Security (BCS2740) - Project 2-2 Cybersecurity & IoT – Ubiquitous Computing & IoT (BCS2750)	 10 10 10

(*) Elective Modules: second year students choose 1 out of 2 modules each semester. Each module comprises an elective course, an elective project and related skill classes.

Period 2.1

Computer Networks (BCS2110)

Coordinator: Adriana Iamnitchi

Desired prior knowledge: Students should be comfortable with basic computer science concepts such as algorithms, data structures, and programming in languages like C, C++ or Java.

Prerequisites: None.

Description: This course introduces the fundamental concepts in computer networking. It covers the principles and structures of network architectures, protocols, and interfaces. Topics include the OSI and TCP/IP models, network devices, routing algorithms, wireless communication, network security, and emerging technologies.

Knowledge and understanding: By the end of the course, students will understand the architecture, protocols, and components that constitute computer networks. They will have a clear grasp of how these elements interact to provide reliable, secure communication across interconnected devices.

Applying knowledge and understanding: Students will apply their knowledge through hands-on labs and projects, configuring network setups, implementing protocols, analyzing network traffic, and engaging in simulation activities to understand network behaviors and troubleshooting.

Making judgements: Students will be required to critically evaluate network designs, protocol efficiency, and security measures. They will make informed decisions to optimize network performance and resolve complex networking issues.

Communication: Students will develop their communication skills through group projects and presentations, where they will articulate complex networking concepts and solutions.

Learning skills: Students will learn through a combination of theoretical knowledge and hands-on experience with packet analyzers and network simulations.

Study material: James F. Kurose & Keith Ross, Computer Networking: A Top-Down Approach, 8th edition (2022). ISBN: 9781292405469. Additional material provided electronically.

Assessment: Written exam (70%) and group projects (30%).

Recommended literature: None.

ECTS: 4

Introduction to Artificial Intelligence (BCS2120)

The course starts with an analysis of the question “Can machines think?”, and the preconceptions usually encountered in discussions about that idea. Next, the metaphor of an “intelligent agent” is introduced, that is, of an entity that pursues goals by perceiving and acting flexibly and autonomously in a possibly very complex environment. Several state-of-the-art concepts, algorithms, and methods that enable computers (i.e., software and robots) to solve problems in a way that deserves to be called intelligent are discussed. Besides the technical background, societal and ethical issues around artificial intelligence such as the current quest for human-centered and explainable AI and how computational techniques might lead to biases and misconceptions through incautious data collection are discussed.

Intelligent User Interfaces (BCS2130)

Intelligent user interaction is a relatively new field in Computer Science, involving the areas of Human-Computer Interaction and Artificial Intelligence. It can often be seen as the intersection of computer science, behavioural sciences, and other field of study. This course will start with an overview of how to develop user-centric interfaces, what is the role of data analysis and research design methods in prototyping, and how different phases, from ideation to high fidelity prototyping relate to each other. Moreover, intelligent user interactions, involving technologies able to automatically recognize human intentions and behaviours, will be discussed. These can come, for example, from the fields of computer vision, speech processing, text mining. A thriving area in the field is that of interface personalization and affective computing, that is, computing driven by human emotions and behaviours.

Period 2.2

Software Engineering and Architectures (BCS2210)

In this course, the student is introduced to the software engineering process. The course addresses the way in which large and complex software projects are conceived and managed. Topics in this course include, among others, requirement analysis, design methodologies, implementation strategies and test and maintenance procedures. In addition, the course discusses the software architectural design process. Several guidelines and several popular example software architectures are presented, as are different software delivery platforms and the current state of the art app development. After completing this course, the student will be able to judge the viability of a selected software development methodology and architectures.

Principles of Programming Languages (BCS2220)

Coordinator: Dr. Tony Garnock-Jones

Desired prior knowledge: Experience of writing programs (in any programming language); acquaintance with the idea of “dynamic dispatch” from object-oriented programming or “pattern matching” as seen in many functional programming languages; acquaintance with the basics of formal logic: propositions, connectives, judgements, inference rules. Discrete Mathematics; Procedural Programming; Object-Oriented Modelling

Prerequisites: None.

Description: Programming Languages are ubiquitous. Every program with an “if” statement is, in an important sense, an interpreter for a programming language, however simple! This course will equip students with an analytical toolkit for understanding contemporary programming languages, relating them to a “standard model” of programming languages. The flip-side of analysis is construction: students will develop the basic understanding and knowledge necessary to begin creating programming languages of their own and writing tools which interpret, compile, or otherwise process other programs as their input data. On the way, students will examine the

algebraic, functional "heart" of many programming languages, using equational reasoning to investigate time, change, concurrency and communication in programs.
Knowledge and understanding: After successful completion of the course, students will be able to relate specific programming languages to a "standard model" of programming languages.
Applying knowledge and understanding: Students will learn to solve programming problems in "pure functional" style, including the use of higher-order functions, and to translate between high-level data types and various low-level data representations as seen in real machines.

Making judgements: Students will be able to differentiate between syntactic, semantic and pragmatic aspects of programming and programming languages, to compare and contrast static types and run-time predicates over data, and to explain the benefits and limitations of "pure" or "effect-free" styles of programming.

Communication: Students will be able to discuss the ubiquity of interpretation in computer programming and to explain program execution in terms of the interaction between data and control.

Learning skills: Finally, after completion of the course, students will be able to identify the various languages they come across in day-to-day programming and to identify opportunities for applying linguistic techniques to programming problems.

Study material: Course notes, slides, and other information will be made available.

Assessment: [TBD]

Recommended literature: Krishnamurthi, Shriram. Programming Languages: Application and Interpretation. Version 3.2.2., 2023. https://www.plai.org/.

Additional literature:
- Felleisen, Matthias, Robert Bruce Findler, Matthew Flatt, and Shriram Krishnamurthi. How to Design Programs. Second edition. MIT Press, 2014. https://htdp.org/.
- Fisler, Kathi, Shriram Krishnamurthi, Benjamin S. Lerner, and Joe Gibbs Politz. A Data-Centric Introduction to Computing. Version 2023-02-21., 2023. https://dcic-world.org/.
- Van Roy, Peter, and Seif Haridi. Concepts, Techniques and Models of Computer Programming. Cambridge, Massachusetts: MIT Press, 2004. https://www.info.ucl.ac.be/~pvr/book.html.
- Abelson, Harold, Gerald Jay Sussman, and Julie Sussman. Structure and Interpretation of Computer Programs. 2nd ed. Cambridge, Massachusetts: The MIT Press, 1996. https://mitpress.mit.edu/sites/default/files/sicp/index.html.
- Felleisen, Matthias, Robert Bruce Findler, and Matthew Flatt. Semantics Engineering with PLT Redex. Cambridge, Massachusetts: MIT Press, 2009.

ECTS: 4

Image and Video Processing
In this course, students will have a brief introduction to basic 2D signals and systems, sampling, image filtering. Colour domain processing in different spaces (RGB, CiE, Lab) and its relevance to our visual perception system will be presented. Students learn about linear and non-linear filtering in the spatial domain, for segmentation, noise reduction, smoothing, among others. Frequency domain transforms will be presented (Fourier, DCT), along with their use in filtering for image enhancement, de-noising, restoration, and the understanding of standards like JPEG. Video analysis will be introduced, with a focus on motion estimation and its relevance to compression standards like MPEG.

This course is part of M2-1 Intelligent Interaction (BCS2710)

Machine Learning

Examiners: Dr. Evgueni Smirnov & Dr. Enrique Hortal Quesada

Desired prior knowledge: Procedural Programming, Calculus, Linear Algebra, Logic
Prerequisites: None.

Description: Machine learning is a major frontier field of artificial intelligence. It deals with developing computer systems that autonomously analyse data and automatically improve their performance with experience. This course presents basic and state-of-the-art techniques of machine learning. Presented techniques for automatic data classification, data clustering, data prediction, and learning include Decision Trees, Bayesian Learning, Linear and Logistic Regression, Recommender Systems, Artificial Neural Networks, Support Vector Machines, Instance-based Learning, Rule Induction, Clustering, and Reinforcement Learning. Lectures and practical assignments emphasize the practical use of the presented techniques and prepare students for developing real-world machine-learning applications.

Knowledge and understanding: After successful completion of the course, students will be able to describe and explain the basic machine learning algorithms. Students will understand the mathematical foundation of machine learning algorithms and how mathematical methods are successfully combined to obtain the variety of machine learning algorithms that are currently available.

Applying knowledge and understanding: Students will acquire the knowledge to apply, formulate, and validate techniques from machine learning and to apply basic machine learning algorithms to real-life problems. Students will be able to implement machine-learning algorithms in software and apply existing machine learning software implementation to datasets. Students will have the necessary knowledge to design, implement, and apply data processing systems that autonomously extract information from data, interpret results, and make decisions.

Making judgements: Students learn how to critically analyse real-world problems, select appropriate machine learning techniques according to the specific problem, and predict the consequences of their choices. After successful completion of the course, students gain the ability to judge which problems can be solved better and to which extend through the application of machine learning techniques. Students obtain an awareness of and responsibility for ethical and social consequences of developments in and application of machine learning.

Communication: The skills acquired during the course will allow students to present the results of different stages of the application of machine-learning techniques to specialists or non-specialists.

Learning skills: After successful completion of the course, students can analyse, adapt, design, implement, and critically reflect on machine-learning algorithms and tools. Students also obtain the critical fundamental skills and knowledge to study further advanced machine learning techniques in the professional literature.

Study material: Lecture material provided during the lecture.

Recommended literature:
- T. Mitchell (1997). Machine Learning, McGraw-Hill, ISBN-13: 978-0071154673.
- H. Blockeel, Machine Learning and Inductive Inference (course text), Uitgeverij ACCO, 2012.
- I.H. Witten and E. Frank (2011). Data Mining: Practical Machine Learning Tools and Techniques (Third Edition), Morgan Kaufmann, January 2011, ISBN-13: 978-0123748560.

Exam: Written “open-book” exam at the end of the course. During the course, students receive several graded assignments that can earn them a maximum bonus grade of 1.0.

This course is part of M2-1 Artificial Intelligence and Machine Learning (BCS2720)

Project 2-1 (BCS2710 & BCS2720)

Coordinator: Dr. Katharina Schneider

Prerequisites: Students must have passed Project 1-1. Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms. The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 1. This project is a prerequisite for Project 3-1.

Description: Students work on a project assignment in small groups. The group composition stays the same for the whole project and is announced at the beginning of period 2.1. Throughout the project, the groups are guided by a tutor with respect to the project management. The project assignment is related to the content of the courses from period 2.1 and 2.2. In periods 2.1 and 2.2, the students work on the project, while also having to attend the courses of these periods. They meet their tutor approximately once every two weeks. In period 2.3, the students work three weeks full-time on the project and meet their tutor about once to twice a week.
The focus of this project lays on the software implementation/design and the product functionality. During the project, the students have to hand in several deliverables such as a project plan after a few weeks from the start or the implemented code at the end of the project. Peer feedback on implementations will add to the quality of feedback the students receive. Presentations throughout the project will be used to communicate the progress to the examiners.

Applying knowledge and understanding: Students will learn to concretize project assignment and construct and maintain a planning Furthermore, they will learn formulating, selecting and validating models for the problem chosen and collect and interpret experimental data with evaluation metrics. Lastly they will improve their ability to plan and chair meetings, create notes for minutes, work in a team such that the workload is balanced and plan teamwork by setting deadlines and distributing tasks.

Making judgements: After completing this project, students will be able to compare and criticize results, position them in terms of the literature diagnose limitations and formulate a discussion.

Communication: Students will be able to write a scientific paper that: describes the project, explains the methods, summarizes the outcomes, discusses them and makes the conclusions. Students will be able to present and defend project in English and coordinate project progress in project meetings

Learning skills: Students will be able to reflect on the progress of the project and study relevant literature to solve problem at hand.

Study material: Project manual project 2-1, Maastricht University.

Assessment: The assessment is composed of a grade for the following deliverables:
- Project plan
- Product
- Report

Furthermore, a grade for the project management and a peer feedback grade on the product functionality will be included in the final grade.

As the focus of this project lays on the software implementation/design and product functionality, the grade for the product has the highest weight.

The examiners can deviate from the group grade if a student shows either outstanding or not enough contribution to the project. Missing mandatory project events such as project meetings and examination moments will automatically lead to a reduction of the grade or even to receiving an NG for the project.

By passing skill classes, the students can get a reward called skillClassGrade, which is 1 if the students passed all skill classes, 0.5 if the students passed all but one skill classes, and 0 if the students passed all but two skill classes. Failing more than two skill classes will lead to an NG in the project.

Skill Classes: The project comes along with skill classes that enhance the students' soft and hard skills. They are closely related to the deliverables of the project. Skill classes are mandatory to pass to complete the project.

Elective Module Project 2-1 Intelligent Human-Computer Interaction (BCS2710)

During this project, students will employ advanced video processing techniques to enhance the standard HCI experience. To allow students to quickly move on to the exploitation of contextual information, they will use open-source libraries and code for body gesture tracking (e.g. openpose) or facial expression recognition. This will draw away the emphasis from the technical details regarding the underlying AI models but will teach the students how to handle software development when interfacing with existing code and pre-trained models. The aim will be to build intelligent interfaces instead of accurate AI models. Specific topics that might be interleaved between years include (i) gesture-driven interfaces, i.e., interfaces that can be driven by hand/finger gestures instead of mouse clicks or keyboard commands; (ii) analysis of user interfaces based on eye-gaze tracking; (iii) automatic emotion and/or personality recognition during interaction and personalization, in replacement of personality questionnaires.

Elective Module Project 2-1 Adaptive Systems (BCS2720)

During this project, students will develop game-playing agents of different levels for simple video games. To allow students to quickly venture into the software development issues encountered when adding techniques from artificial intelligence or machine learning into video games, they will rely on existing frameworks for the core AI and ML techniques, so they do not have to know all the technical details that go into these advanced methods. Instead, they will focus on working within the computational and memory constraints as defined by the platform (e.g. Android) and the real-time operation of video games.

M2-1 (project + course) ECTS: 10

Period 2.4

Embedded Programming (BCS2410)

This course starts by introducing the students to CPU architectures (circuits, adders, multipliers, floating-point units, RAM) and then introduces the hardware description language VHDL. Students will get to practice design, implementation and debugging while they create a circuit that, for example, can add two numbers. Near the end of the course, when the C programming language has been introduced in parallel skill classes, the step to microcontroller programming will be made, discussing the advantages and limitations of micro-controllers and micro-processors, and allowing students to get some initial experience with limited general-purpose circuits.

Computer Security (BCS2420)

Computer security is the process of detecting and preventing unauthorized and illicit access to a computer. As information systems have become mandatory in the commercial world, coupled with the increased frequency of security incidents, organizations now recognize the need for a comprehensive security strategy. This course will introduce a wide range of topics in such as security concepts and services, physical, operational, and organizational security, the role of people in systems security, an introduction to cryptography and the public key infrastructure, computing systems hardening, secure code, and secure applications development.

Parallel Programming (BCS2430)

Coordinator & examiner: Dr. Bastian Küppers

Desired prior knowledge: Introduction to Computer Science, Prodecural Programming, Objects in Programming, Data Structures and Algorithms

Prerequisites: None

Description: Parallel programming is the paradigm of doing computations on a computer in parallel. This is possible, since nowadays almost all computer systems include so-called multi-core chips. To exploit the full performance of such systems, parallel programming needs to be emploed.

This course covers shared-memory parallelization with OpenMP as well as parallelization with message passing on distributed-memory architectures with MPI. The course starts with a recap of the programming language C followed by a brief theoretical introduction to parallel computing. Next, the course treats theoretical aspects like MPI communication, race conditions, deadlocks, efficiency as well as the problem of serialization. The course is accompanied by practical labs in which the students have the opportunity to apply the newly acquired concepts.

Knowledge and understanding: Students recall the basic concepts of parallel programming and recognize important parallelization patterns.

Applying knowledge and understanding: Students are able to write parallel programing code using OpenMP and MPI.

Making judgements: **Students are able to understand parallel source code and can decide whether a** taken approach is appropriate for a given problem.

Communication: Students are able to explain why a specific approach is adequate for a given problem.

Learning skills: Students are able to study literature on parallel programming autonomously in order to comprehend important details and problems in the field.

Study material: Lecture slides, Source code examples

Exam: Written final exam

Recommended literature: Eijkhout: The Art of HPC Vol. 1 & 2 (https://theartofhpc.com/)

Additional literature: Pacheco: An Introduction to Parallel Programming

ECTS: 4

Period 2.5

IT Management and Privacy (BCS2510)

This course will provide the student with a general introduction to enterprise information systems and the functionality of specific enterprise IS types (e.g., Enterprise Resource Planning, Customer Relationship Management, Supply Chain Management, Executive IS, Product Lifecycle Management systems). The students will become familiar with System Development Life Cycle (SDLC), different adaptation and transition approaches, project management approaches, as well as issues related to post-implementation, such as maintenance. Additionally, the course will provide the student with an introduction to the legal and technological aspects of EU and global data protection and privacy issues, and a business understanding of data usage practices, including the GDPR. Students will learn to grasp the key actors, concepts, and obligations of the GDPR, and develop awareness and understanding of the legal requirements they will encounter in their professional careers. The general introduction of the aspects of the GDPR is concretized by several examples. Students will learn how to become ambassadors for a global digital economy and society by learning to approach the processing of personal data competently and ethically. The concepts of digital ethics and accountability along with their limitations are illustrated through different real-life cases and scenarios.

Numerical Methods (BCS2540)

Coordinator: Dr. Pieter Collins

Examiners: Dr. Pieter Collins & Dr. Ir. Martijn Boussé

Desired prior knowledge: Calculus, Linear Algebra

Prerequisites: None.

Description: Numerical methods are techniques for solving problems from continuous mathematics (calculus and linear algebra) with the aid of a digital computer. In this course, we will cover the fundamental concepts of numerical mathematics, including the floating-point representation of real numbers, truncation and round-off errors, iterative methods and convergence. We will study the simplest and most important methods for core problems of continuous mathematics, namely the solution of algebraic equations and differential equations, interpolating data by polynomials, numerically estimating derivatives and integrals, approximating functions by polynomials and trigonometric series, solving systems of linear algebraic equations and computing eigenvalues. There will be a strong practical component, with students being expected to write their own numerical code and test the performance and suitability of different methods on various problems.

Knowledge and understanding: By the end of this course, students will have knowledge of the fundamental problems of numerical mathematics and basic methods for their solution. Students will understand issues of efficiency and numerical accuracy, will be able to analyse which numerical methods are likely to perform best on different types of problem, and evaluate whether the results of a given computation are trustworthy. Students will be able to write their own code (in MATLAB) implementing basic numerical algorithms. Advanced students will have the skills necessary to adapt existing numerical algorithms and develop new algorithms.

Applying knowledge and understanding: Students will be expected to implement the methods covered in the lectures themselves, apply them to practical problems, and explain the performance of different algorithms in terms of theoretical analyses.

Making judgements: Students will learn how to analyse which numerical methods are likely to perform best on different types of problem, and to evaluate whether the results of a given computation are trustworthy.

Communication: Students will learn the terminology required to discuss numerical algorithms and the results of numerical computations with mathematicians, (social) scientists and engineers.

Learning skills: Students will learn to design, analyse, implement and apply numerical methods.

Study material: Slides, pre-recorded lectures, exercise sheets.

Assessment: Written examination with formula sheet (100%).

Recommended literature: J.D. Faires & R. Burden, "Numerical Methods", International 4th Edition, Cengage, 2012; ISBN: 978-0-495-38569-1.

Additional literature: C.F. Gerald & P.O. Wheatley, "Applied Numerical Analysis", Seventh Edition, Pearson, 2003; ISBN: 0-321-13304-8. T. Siauw & A.M. Bayen, "An Introduction to Matlab Programming and Numerical Methods for Engineers", Academic Press, 2015; ISBN 978-0-12-520228-3.

ECTS: 4

High Performance Computing (BCS2530)

This course focuses on exploring different parallelism models and methods to obtain high performance in applications. Using the C++ language as a driving language, aspects of parallelism, GPU operations and memory performance are discussed. Programming for heterogeneous architectures programming will be introduced with the CUDA language. Performance modelling, guided performance tuning and the roofline model are also explored. After the course, students will be able to write efficient parallel implementations of various problems and will become familiarized with some performance tuning techniques.

This course is part of M2-2: High Performance Computing

Information Security (BCS2540)

Coordinator & examiner: Dr. Bastian Küppers

Desired prior knowledge: Procedural Programming, Objects in Programming, Data Structures and Algorithms

Prerequisites: None.

Description: Information security is a subfield of Computer security dealing with protecting information, which means to ensure confidentiality, integrity, and availability of information. Topics such as Cryptography, Access Control, Identification and Authentication play a key role in this field to ensure the technical foundation for information security. However, nowadays, where information is in constant flow across the globe, network security and (global) privacy laws are topics that also need to be considered when ensuring information security.

Knowledge and understanding: Students will gain an in-depth understanding of information security fundamentals and their applications in real-world scenarios. In detail, they will understand the basic principles of confidentiality, integrity, and availability and their importance for information security. Furthermore, the students will understand the technologies available for securing information and their proper application.

Applying knowledge and understanding: After completing the course, the students will be able to develop computer systems that are in-line with the state-of-the-art of information security.

Making judgements: By understanding the fundamentals of information security, the students will be able to understand and spot mistakes in the proper application of technologies for ensuring information security.

Communication: Students will be able to explain the principles of information security to specialists and non-specialists. They will be able to explain if the design of a computer system is appropriately taking into account the principles of information security or not and how the system could be improved.

Learning skills: Students will be able to read and understand literature on information security autonomously. The will be able to design and implement computer systems that obey the basic principles of information security.

Study material: Lecture slides, Source code examples

Assessment: Group project (20%)

Exam: Written final exam (80%)

Recommended literature:
- Buchmann: Introduction to Cryptography
- Pfleeger & Pfleeger: Security in Computing

Additional literature:
- Goodrich & Tamassia: Introduction to Computer Security
- Tanenbaum & Bos: Modern Operating Systems

This course is part of M2-2: Cybersecurity & IoT – Information Security

Ubiquitous Computing & Internet of Things (BCS2750)

This course will look at the technical specifics of dealing with low power devices, the architecture description language and their network communication protocols including Zigbee, WLAN and Bluetooth and the information flow this can entail. Besides the technical challenges of low power devices, the course will also look at the business implications of ubiquitous computing and provide some answers to the question of why to introduce smart sensors and place them with customers. By considering models such as outcome economy, students will learn how the Internet of Things approach can be integrated into a business plan and generate business value.

This course is part of M2-2: Cybersecurity & IoT – Ubiquitous Computing & IoT

Project 2-2 (BCS2730, BCS2740 & BCS2750)

Coordinator: Dr. Katharina Schneider

Prerequisites: Students must have passed Project 1-2. Furthermore, the student has to have passed at least two out of the following three courses: Procedural Programming, Objects in Programming, and Data Structures and Algorithms. The student furthermore needs to be registered for or has already completed at least three courses of the programme in year 2, semester 2. This project is not a prerequisite for another project / course.

Description: Students work on a project assignment in small groups of about six students. The concrete assignment is defined by the students given some umbrella topics that match the courses in the curriculum. Students indicate their umbrella topic preference at the beginning of period 2.4. The group composition stays the same for the whole project and is based on the topic preferences of the students. A least regret algorithm is used to ensure that the overall regret is minimized. Throughout the project, the groups are guided by a tutor with respect to the project management. In periods 2.4 and 2.5, the students work on the project, while also having to attend the courses of these periods. They meet their tutor approximately biweekly. In period 2.6, the students work three weeks full-time on the project and meet their tutor about twice a week.

The focus of this project lays on the project planning and communication of progress and results. During the project, the students have to hand in several deliverables such as a project plan after a few weeks from the start or the implemented code at the end of the project. Formative feedback sessions with the examiners on intermediate project plans will add to the quality of feedback the students receive. Presentations throughout the project will be used to communicate the progress to the examiners.

Applying knowledge and understanding: Students will learn to set up a project assignment and construct and maintain a planning. Additionally, they will learn formulating, selecting and validating models for a concrete problem at hand and to collect and interpret data with evaluation metrics. Lastly they will improve their ability to plan and chair meetings, create notes for minutes, work in a team such that the workload is balanced and plan teamwork by setting deadlines and distributing tasks.

Making judgement: After completing this course successfully, students will be able to compare and criticize results, position them in terms of the literature; diagnose limitations and formulate a discussion

Communication: Students will be able to write a scientific paper that: describes the project, explains the methods, summarizes the outcomes, discusses them and makes the conclusions. Furthermore, student will be able to present and defend project in English. Coordinate project progress in project meetings

Learning skills: Students will learn to reflect on the progress of the project and study relevant literature to solve problem at hand

Study material: Project Opening slides, Maastricht University

Assessment: The assessment is composed of a grade for the following deliverables:
- Project plan
- Poster
- Report

Furthermore, a grade for the project management, the groups' performance during the poster session and during a defense at the end of the project, and a peer feedback grade on the poster presentation will be included in the final grade.

As the focus of this project lays on the planning and the communication of progress and results, the grades for the project plan, the poster presentation and the defense together have the highest weight.

The examiners can deviate from the group grade if a student shows either outstanding or not enough contribution to the project. Missing mandatory project events such as project meetings and examination moments will automatically lead to a reduction of the grade or even to receiving an NG for the project.

By passing skill classes, the students can get a reward called skillClassGrade, which is 1 if the students passed all skill classes, 0.5 if the students passed all but one skill classes, and 0 if the students passed all but two skill classes. Failing more than two skill classes will lead to an NG in the project.

Skill classes: The project comes along with skill classes that enhance the students' soft and hard skills. They are closely related to the deliverables of the project. Skill classes are mandatory to pass to complete the project.

Elective Module Project 2-2 High Performance Computing (BCS2730)

In this project, students will tackle the particle physics problem of particle track reconstruction, in which they will attempt to reconstruct the trajectories (tracks) of particles as they leave energy deposits in a Large Hadron Collider detector. They will develop a pattern recognition model and minimize the error on found tracks by applying filtering techniques. To efficiently solve track reconstruction, they will design and implement data reductions and a data-parallel approach to tracking on GPU architectures. Students will learn a systematic approach to performance engineering by iteratively designing, developing, and testing a high throughput solution to tracking.

Elective Module Project 2-2 Cybersecurity & IoT (BCS2740 & BCS2750)

During this project, students will build a demonstrator of a secure distributed computational process using limited power hardware devices and low power data collection. They will develop a distributed approach to analytics by avoiding the need to centralize data, but instead working around the size, heterogeneity, and ownership of data by dynamically deploying micro-services in a federated learning setup from the service repository to the IoT edge on low power computational models. Attention will be paid to data security and privacy.

M2-2 (project + course) ECTS: 10 gr

3 Master

3.1 Master Artificial Intelligence

The Master in Artificial Intelligence (AI) is a two-year advanced programme organized by the Department of Advanced Computing Sciences. The focus of this programme is on the understanding, design and creation of intelligent systems, such as those used in robotic systems, games or digital personal assistants. Artificial Intelligence has become a very active domain in both academia and industry. It has given rise to computer programmes and robots that learn from experience, recognize and adapt to patterns in their environment, and reason strategically in complex decision-making situations.

The impact of the field of Artificial Intelligence is pertinent due to the key role it plays in technological applications that have become indispensable in society, such as simple personal assistants that adapt the settings of your smart phone to automatically recognised activities (e.g. driving or attending a meeting, automated trading software used in real markets to respond to rapid price changes, interactive computer games that include human like opponents, robotic assistance in the exploration of dangerous environments, etc.). In this Master's programme, you are trained to become an expert and capable of dealing with todays and future challenges in the field of Artificial Intelligence and its applications.

The master's programme Artificial Intelligence covers a range of subjects emphasizing the following research topics as its core:

1. Intelligent techniques for playing and solving (board) games and controlling virtual characters in video games;
2. Situated agents to study the control and coordination of embodied agents, i.e. robots (e.g. autonomous flying robot swarms);
3. Multi-agent systems of collaborating or competing autonomous intelligent systems;
4. Machine learning to extract useful patterns and knowledge from experience and make predictions about the future;

The members of the teaching staff are actively involved in one or more of these research topics. As a result, the educational contents of the courses relate directly to the research performed..

3.2 Curriculum of the first year of the Master Programme Artificial Intelligence

Year 1		ECTS
Period 1 (*)	Intelligent Search & Games (KEN4123)	6
	1 elective of the following courses:	6
	Data Mining (KEN4113)	6
	Stochastic Decision-Making (KEN4221)	6
	Signal and Image Processing (KEN4222)	
	Research Project 1 (**)	
Period 2 (*)	Advanced Concepts in Machine Learning (KEN4154)	6
	1 elective course from the following courses:	6
	Deep Learning for Image and Video Processing (KEN4244)	6
	Advanced Natural Language Processing (KEN4259)	
	Research Project 1 (**)	
Period 3	Research Project 1-1 (KEN4130)	6
Period 4 (*)	Agents and Multi-Agent Systems (KEN4111)	6
	1 elective course from the following courses:	6
	Explainable AI (***) (KEN4246)	6
	Dynamic Game Theory (KEN4251)	6
	Planning and Scheduling (KEN4253)	6
	Building and Mining Knowledge Graphs (KEN4256)	6
	Research Project 2 (**)	
Period 5 (*)	Autonomous Robotic Systems (KEN4114)	6
	1 elective course from the following courses:	6
	Information Retrieval and Text Mining (KEN4153)	6
	Introduction to Quantum Computing for AI and DS (KEN4155) (****)	6
	Reinforcement Learning (KEN4157)	6
	Computer Vision (KEN4255)	
	Research Project 2 (**)	
Period 6	Research Project 2 (KEN4131)	6

(*)	*ECTS credits obtained in year 1 of the programme cannot be used for exemptions in year 2 of the programme. 90 unique ECTS (course) credits (+ 30 ECTS for the Master's thesis) need to be obtained throughout the Master's programme.*
(**)	*The Research Project 1 will start in period 1.1 and 1.2 with weekly meetings. The credits for the project will become available at the end of period 1.3. The Research Project 2 will start in period 1.4 and 1.5 with weekly meetings. The credits for the project will become available at the end of period 1.6.*
(***)	*The course has a capacity of 60 students.*
(****)	*This course is a prerequisite for the elective courses Quantum Algorithms, Quantum AI, and Quantum Information and Security. These four courses, together with a dedicated research project on quantum computing, forms the specialization in Quantum Computing for AI and Data Science.*

3.3 Curriculum of the second year of the Master Programme Artificial Intelligence

Period 1, 2 and 3 of year two of the master's program consist of electives to be chosen by the student. This optional program can be assembled at your own choice from the options provided, but within academic significance, level and relevance to your master's track. The choice of electives is subject to approval by the Board of Examiners. The electives consist of the following options to choose from: master courses to be followed at the Department of Advanced Computing Sciences, at other UM master programmes, at another research university, a research project, an internship, a semester abroad at a foreign university. Note that you must have obtained at least 40 ECTS of course year 1 in order to enter the second year of the programme.

Electives at Maastricht University outside the Department of Advanced Computing Sciences

It is possible to take electives at other relevant master's programmes at Maastricht University for at most 13 ECTS in the second year of the programme. The following courses below will be automatically approved by the Board of Examiners of the master's programmes AI and DSDM. You should apply through the Special Course Approval procedure via the My UM Portal. Note that they may have limited capacity.

School of Business and Economics	ECTS
Collective Decision Making (EBC4005)	6.5
Supply Chain Operations Management (EBC4016)	6.5
Negotiation & Allocation (EBC4193)	6.5
Ethics, Privacy and Security in a Digital Society (EBC4026)	6.5
Big Data Econometrics (EBC4218	6.5
Digital Business and Economics (EBC4083)	6.5

Faculty of Psychology and Neuroscience

Besides complying that you have passed 40 ECTS, for taking these electives at FPN you should have passed "Advanced Concepts in Machine Learning" and "Autonomous Robotic Systems" at the Master AI or Master Data Science for Decision Making.

Faculty of Psychology and Neuroscience	ECTS
Auditory and Higher Order Language Processing (PSY4051)	4
Perception and Attention (PSY4052)	4
Sensorimotor Processing (PSY4054)	4

Exam: Depends on content of the elective program.

ECTS: 30

Internships:
Another option for the elective semester in the Master Programme is to conduct a business or research internship. The students can choose the company or research organisation themselves. Together with a supervisor from the Department of Advanced Computing Sciences and a representative of the host organisation, the student fills out an internship proposal (which can be found on Canvas) and this requires approval of the Board of Examiners prior to its start. For this reason, it is important to start this process early. The university uses a standard internship agreement that students must use.

Year 2		ECTS
Semester 1 *	• Internship (research or business) • Study abroad • Elective courses at other UM Master's programmes (at most 13 ECTS)	30
	AND/OR:	
Fall:	*At most 2 electives from the following courses:*	
Period 1	Data Mining (KEN4113) Mathematical Optimization (KEN4211) Stochastic Decision-Making (KEN4221) Signal and Image Processing (KEN4222) Quantum Algorithms (KEN4235)	6 6 6 6 6
Period 2	*At most 2 electives from the following courses:*	
	Quantum AI (KEN4236) Quantum Information and Security (KEN4237) Model Identification and Data Fitting (KEN4242) Advanced Natural Language Processing (KEN4259) Network Science (KEN4275)	6 6 6 6 6
Period 1-3	Research Project 2-1	6
Spring:		
Period 4	*At most 2 electives from the following courses:*	
	Data Fusion (KEN4223) Explainable AI (**) (KEN4246) Dynamic Game Theory (KEN4251) Planning and Scheduling (KEN4253) Building and Mining Knowledge Graphs (KEN4256) Computational Statistics (KEN4258)	6 6 6 6 6 6
Period 5	*At most 2 electives from the following courses:*	
	Information Retrieval and Text Mining (KEN4153) Introduction to Quantum Computing for AI and Data Science (***) (KEN4155) Reinforcement Learning (KEN4157) Symbolic Computation and Control (KEN4252) Algorithms for Big Data (KEN4254) Computer Vision (KEN4255)	6 6 6 6 6 6
Period 4-6	Research Project 2-2	6
Semester 2	Master's thesis AI (KEN4160)	30

(*) Note: during the elective semester (first semester of year 2) of the master's programme it is possible to take electives from our other master's programme or relevant master's programmes at Maastricht University (maximum of 13 ECTS outside the Department of Advanced Computing Sciences) or to participate in a research project, a business internship or a study abroad semester at one of our partner universities. Please contact exchange officer and/or the Student Counsellor for more information.

(**) The course has a capacity of 60 students

(***) This course is a prerequisite for the elective courses Quantum Algorithms, Quantum AI and Quantum Information and Security. These four courses, together with a dedicated research project on quantum computing, form the specialization in Quantum Computing for AI and Data Sciences.

3.4 Core courses Master Programme Artificial Intelligence

Period 1.1

Intelligent Search and Games (KEN4123)

Examiners: Prof. dr. Mark Winands & dr. Dennis Soemers

Desired prior knowledge: Data Structures & Algorithms

Course description: In this course, the students learn how to apply advanced techniques in the framework of game-playing programs. The following subjects will be discussed:
1. Basic search techniques. Alpha-beta; A*.
2. Advanced search techniques. IDA*; B*, transposition tables; retrograde analysis and endgame databases; proof-number search and variants; multi-player search methods; Expectimax and *-minimax variants.
3. Heuristics. killer moves; history heuristic, PVS; windowing techniques; null-moves; forward-pruning techniques; selective search.
4. Monte Carlo methods. Monte Carlo Tree Search (MCTS) techniques, enhancements, and applications; AlphaGo and AlphaZero approaches.
5. Video game AI techniques: World representations, GOAP, hierarchical task networks, behaviour trees.

Knowledge and understanding: The student can explain basic and advanced search techniques and can identify which of them to use either in a game context, or in problems with a similar structure.

Applying knowledge and understanding: Students have obtained the knowledge to develop, program, analyse, and apply advanced techniques autonomously to a wide variety of problems. They will also learn that adapting known techniques to fit a given problem can achieve a better performance.

Making judgements: Students will be able to judge the quality of approaches (systems or scientific publications) based on the techniques taught.

Communication: Students will be able to present the results of their game programs and search algorithms to specialists or non-specialists.

Learning skills: Students will be able to familiarize themselves with Game AI techniques beyond the scope of the course in order to solve a problem.

Study material: Course notes and other information made available.

Recommended Literature:
- Millington, I. (2019). Artificial Intelligence for Games, 3rd Edition, CRC Press, ISBN: 978-1138483972
- Russell, S.J. and Norvig, P. (2020). Artificial Intelligence: A Modern Approach, 4th edition. Pearson. ISBN 0-13-461099-7.
- Yannakakis, G.N. and Togelius, J. (2018) Artificial Intelligence and Games, Springer, Berlin. ISBN 978-3-319-63519-4 (eBook) 978-3-319-63518-7 (hardcover)

Assessment: Written exam (50%) + a large practical task (50%).

ECTS: 6

Period 1.2

Advanced Concepts in Machine Learning (KEN4154)

Coordinator & examiner: Dr. Enrique Hortal

Desired prior knowledge: Machine Learning

Prerequisites: None.

Description: This course will introduce a number of advanced concepts in the field of machine learning such as Support Vector Machines, Gaussian Processes, Deep Neural Networks, Neuromorphic Learning, etc. All of these are approached from the view that the right data representation is imperative for machine learning solutions. Additionally, different knowledge representation formats used in machine learning are introduced. This course counts on the fact that the basics of machine learning were introduced in other courses so that it can focus on more recent developments and state-of-the-art machine learning research. Labs and assignments will give the students the opportunity to implement or work with some of these techniques and will require them to read and understand published scientific papers from recent Machine Learning conferences.

Knowledge and understanding: Students can explain, construct and adapt powerful machine learning techniques, most with a statistical background. Students recognise the need for non-standard techniques and representations that can be used for complex/structured data. They can explain the strengths and weaknesses of different machine learning approaches.

Applying knowledge and understanding: Students will be able to select, adapt and apply a number of advanced machine learning approaches. They will be able to select the correct representation for a machine learning problem and translate a machine learning problem into a suited representational format.

Making judgements: Students will be able to judge which machine learning approach and data representation is best suited. They will also be able to comprehend and judge machine learning research.

Communication: Students will be able to relate different machine learning techniques to each other and explain their working, benefits, and disadvantages to non-experts. They will also be able to discuss the need and use of structured representation with both experts and non-experts.

Learning skills: Students will be able to relate information from different sources, and read, process and evaluate recent research developments in the field of machine learning.

Study material: Slides that support the lectures and collected notes and chapters from freely available books and course notes.

Assessment: Closed-book written exam (80%) + Assignments (20%)

Recommended literature:
- Pattern Recognition and Machine Learning - C.M. Bishop
- Bayesian Reasoning and Machine Learning - D. Barber
- Gaussian Processes for Machine Learning - C.E. Rasmussen & C. Williams
- The Elements of Statistical Learning - T. Hastie et al.

ECTS: 6

Project 1-1 (KEN4130)

Coordinator: Dr. Linda Rieswijk

Examiner(s): To be announced

Tutors: Dr. Gijs Schoenmakers, Dr. Menica Dibenedetto & Dr. Linda Rieswijk

Desired prior knowledge: None.

Prerequisites: None.

Description: The research project takes place during the three periods of the semester. Project topics are presented at the start of the semester and assigned to students based on their preferences and availability. The emphasis in the first phase is on initial study of the context set out for the project and the development of a project plan. In the second period, the goal is to start modelling, prototyping and developing. In period 3, the implementation, model and/or experiments set out in the project plan has to be finished and reported on. The project results in a project presentation, a project report and possibly a public website and/or product.

Knowledge and understanding: Students get to know and possibly contribute to state of the art methods within the fields of Artificial Intelligence and/or Data Science for Decision Making to answer an open question.

Applying knowledge and understanding: Student write their own research plan in coordination with a staff member (plus possibly outsiders) who act as clients with an open question. Students with different backgrounds and from both masters work together in teams to build and evaluate an answer to an open question. Students find, judge the suitability, apply, and evaluate state of the art techniques to answer questions and construct applications in the field of Artificial Intelligence and Data Science. Students apply the accumulated knowledge from other educational activities in application specific areas

Making judgements: Students judge feasibility of tasks, attainability of goals, and the amount of work involved. Students think about the possible consequences of their work. Students evaluate state of the art and the applicability and scope of research results.

Communication: Students will learn to:
1. orally communicate and cooperate with peers
2. orally report on progress and intermediate results to superiors
3. orally negotiate and communicate with clients
4. communicate their ideas in written form, both for an academic and a general audience
5. give effective presentations

Learning skills: Students increase their own level of knowledge in a specialized sub-discipline of the field of Artificial Intelligence and/or Data Science. Students perform research into recent state of the art techniques. Students learn that the field of Artificial Intelligence and Data Science are constantly evolving beyond what is taught in class

Study material: Slides provided at the end of joint information sessions. Literature provided by the project supervisors.

Assessment: Phase 1: project plan (15%); Phase 2: social media post+ presentation (15%); Phase 3: Project report + presentation (70%)

Skill classes: A number of skill classes will be offered

ECTS: 6

Period 1.4

Agents and Multi-Agent Systems (KEN4111)

Coordinator & examiner: Prof. dr. Gerhard Weiss

Desired prior knowledge: Basic knowledge and skills in programming.

Description: The notion of an (intelligent) agent is fundamental to the field of artificial intelligence. Thereby an agent is viewed as a computational entity such as a software program or a robot that is situated in some environment and that to some extent is able to act autonomously in order to achieve its design objectives. The course covers important conceptual, theoretical and practical foundations of single-agent systems (where the focus is on agent-environment interaction) and multi-agent systems (where the focus is on agent-agent interaction). Both types of agent-based systems have found their way to real-world applications in a variety of domains such as e-commerce, logistics, supply chain management, telecommunication, health care, and manufacturing. Examples of topics treated in the course are agent architectures, computational autonomy, game-theoretic principles of agent-based systems, coordination mechanisms (including auctions and voting), and automated negotiation and argumentation. Other topics such as ethical or legal aspects raised by computational agency may also be covered. In the exercises and in the practical part of the course students have the opportunity to apply the covered concepts and methods.

Formal models that will be investigated: Coordination and interaction models from game theory and social choice theory.

Knowledge and understanding: The student is able to describe and explain single- and multi-agent concepts and methods, and to analyse their strengths and shortcomings.

Applying knowledge and understanding: The student is be able to apply the gained knowledge in concrete application scenarios and practical applications.

Making judgements: The student is be able to judge for a given problem whether and in how far it is beneficial to use an agent-based approach for its solution.

Communication: The student is able to motivate and explain benefits and shortcomings of their usage in a given application, and thereby showing sufficient understanding of single- and multi-agent concepts.

Learning skills: The student is able to study independently and critically literature on single- and multi-agent technology, including, in particular, literature describing new developments in the methods and techniques covered in this course.

Study material: Course slides; supplementary material to be announced.

Assessment: Practical assignments (30%) and written exam (70%)

Recommended literature:
- Stuart Russell and Peter Norvig (2010). Artificial Intelligence. A Modern Approach. 3rd edition. Prentice Hall.
- Gerhard Weiss (Ed.) (2013, 2nd edition): Multi-agent Systems. MIT Press.
- Mike Wooldridge (2009, 2nd edition): An Introduction to Multi Agent Systems, John Wiley & Sons Ltd.
- Yoav Shoham and Kevin Leyton-Brown (2009): Multi-agent Systems. Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press.

ECTS: 6

Period 1.5

Autonomous Robotics Systems (KEN4114)

Examiner: Dr. Rico Möckel

Desired prior knowledge: Discrete Mathematics, Linear Algebra, Probabilities and Statistics, Data Structures and Algorithms, Machine Learning, Search Techniques.

Prerequisites: None.

Description: Operating autonomously in unknown and dynamically changing environments is a core challenge that all robotic systems must solve to work successfully in industrial, public and private areas. Currently popular robotic systems that must demonstrate such capabilities include self-driving cars, autonomously operating drones, and personal robotic assistants. In this course, students obtain deep knowledge in creating autonomous robotic systems that can operate in unknown and dynamically changing environments by autonomously modelling and navigating in such environments. Students learn to approach these challenging tasks through three main techniques: swarm intelligence, model-based probabilistic frameworks, and (mostly) model-free techniques from artificial evolution and machine learning.

Knowledge and understanding: Students gain a deep understanding of the challenges in autonomous robotic systems and how these challenges are addressed in state-of-the-art systems. Students learn about and practice techniques for autonomous mapping, localization, navigation, sensing, modelling robot motion, planning, and decision-making. Through the course, students obtain in-depth knowledge and hands-on experience in a variety of algorithms and techniques including Bayesian filters (like Kalman Filters, Extended Kalman Filters, Histogram Filters, and Particle Filters), artificial neural networks, evolutionary algorithms, and swarm intelligence.

Applying knowledge and understanding: After successful completion of the course, students will have obtained in-depth knowledge to understand, adapt, apply, and autonomous robotics systems. Students obtain the ability to select from a variety of available tools feasible solutions for the complex and rather ill defined problem domains of autonomous robotic systems and to predict the resulting consequences of their choices. Furthermore, students learn how to choose, apply, formulate, and validate models of autonomous robotic systems and of appropriate control techniques from artificial intelligence for these systems.

Making judgements: Students will be able to comprehend and to critically judge scientific publications on autonomous systems, artificial evolution, and swarm intelligence. From this literature, students are able to search for and to critically process information to solve given ill-defined but in practice highly relevant problems in autonomous systems. Students are able to critically discuss social, economic, and ethical consequences of artificial intelligence and autonomous decision-making.

Communication: Students learn to critically discuss challenges and professional solutions in autonomous robotic applications with both experts and non-experts.

Learning skills: The course prepares students to work on robotic applications in professional research and business environments. Students will be able to autonomously acquire new skills and knowledge to develop, program, analyse and apply advanced techniques to a wide variety of problems.

Study material: Thrun et al. (2005), Probabilistic Robotics, The MIT press, ISBN-13: 978-0262201629. Lecture material and publications provided during the lecture.

Assessment: The final course grade is 80% of the final written “closed-book” exam grade plus 20% of the practical group assignments grades.

Recommended literature: Floreano and Nolfi (2000), Evolutionary Robotics, The MIT press. ISBN-13: 978-0262640565.
Dario Floreano und Claudio Mattiussi (2008), Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies, ISBN-13: 978-0262062718

ECTS: 6

Project 1-2 (KEN4131)

Coordinator: Dr. Linda Rieswijk

Examiner(s): T.B.A.

Tutors: Dr. Gijs Schoenmakers, Dr. Menica Dibenedetto & Dr. Linda Rieswijk

Desired prior knowledge: None.

Prerequisites: None.

Description: The research project takes place during the three periods of the semester. Project topics are presented at the start of the semester and assigned to students based on their preferences and availability. The emphasis in the first phase is on initial study of the context set out for the project and the development of a project plan. In the second period, the goal is to start modelling, prototyping and developing. In period 3, the implementation, model and/or experiments set out in the project plan has to be finished and reported on. The project results in a project presentation, a project report and possibly a public website and/or product.

Knowledge and understanding: Students get to know and possibly contribute to state of the art methods within the fields of Artificial Intelligence and/or Data Science for Decision Making to answer an open question.

Applying knowledge and understanding: Student write their own research plan in coordination with a staff member (plus possibly outsiders) who act as clients with an open question. Students with different backgrounds and from both masters work together in teams to build and evaluate an answer to an open question. Students find, judge the suitability, apply, and evaluate state of the art techniques to answer questions and construct applications in the field of Artificial Intelligence and Data Science. Students apply the accumulated knowledge from other educational activities in application specific areas

Making judgements: Students judge feasibility of tasks, attainability of goals, and the amount of work involved. Students think about the possible consequences of their work. Students evaluate state of the art and the applicability and scope of research results.

Communication: Students will learn to:
1. orally communicate and cooperate with peers
2. orally report on progress and intermediate results to superiors

3. orally negotiate and communicate with clients
4. communicate their ideas in written form, both for an academic and a general audience
5. give effective presentations

Learning skills: Students increase their own level of knowledge in a specialized sub-discipline of the field of Artificial Intelligence and/or Data Science. Students perform research into recent state of the art techniques. Students learn that the field of Artificial Intelligence and Data Science are constantly evolving beyond what is taught in class

Study material: Slides provided at the end of joint information sessions. Literature provided by the project supervisors.

Assessment: Phase 1: project plan (15%); Phase 2: social media post+ presentation (15%); Phase 3: Project report + presentation (70%)

Skill classes: A number of skill classes will be offered

ECTS: 6

3.5 Electives courses Master Programme Artificial Intelligence

Period 1.1 & 2.1

Data Mining (KEN4113)

Examiner: Dr. E.N. Smirnov

Desired prior knowledge: Statistics and Basic Machine Learning

Prerequisites: None.

Description: Data mining is a major frontier field of machine learning. It allows extracting useful and interesting patterns and knowledge from large data repositories such as databases and the Web. Data mining integrates techniques from the fields of databases, machine learning, statistics, and artificial intelligence. This course will present basic and state-of-the-art techniques of data mining. The lectures and labs will emphasize the practical use of the presented techniques and the problems of developing real data-mining applications. A step-by-step introduction to data-mining and python-based environments will enable the students to achieve specific skills, autonomy, and hands-on experience. A number of real data sets will be analyzed and discussed.
Formal models that will be investigated: Parametric and non-parametric models for supervised and unsupervised learning.

Knowledge and understanding: Students will acquire knowledge on data preparation, data preprocessing, feature selection/generation, data mining, and model validation.

Applying knowledge and understanding: When confronted with real-life problems, students will be able to identify data-analysis tasks. Then, they will be able to apply data-mining techniques for supervised and unsupervised data-analysis. If necessary, students will be able to design data-mining algorithms specific for the tasks they have.

Making judgements: Students will be able to assess the quality of data-mining models, processes, results, and tools.
Communication: Students will be able to present the results of different stages of data-mining processes to specialists or non-specialists.

Learning skills: Students will be able to recognize their own lack of knowledge and understanding and take appropriate action such as consulting additional material or other sources of help.

Study material: Course notes, slides, and other information made available.

Assessment: Written exam + practical assignments.

Recommended Literature: Han, J., Pei, J., and Tong, H. (2022). Data Mining Concepts and Techniques, 4th Edition, ISBN-10: 9780128117606, ISBN-13: 978-0128117606

Additional literature: Pang-Ning, T., Steinbach, M., Karpatne, A., and Kumar, V. (2018). Introduction to Data Mining, 2nd Edition, Pearson, ISBN-10: 0133128903, ISBN-13: 978-0133128901

ECTS: 6

Stochastic Decision-Making (KEN4221)

Coordinator: Dr. Gijs Schoenmakers

Examiners: Dr. Gijs Schoenmakers & Dr. Dennis Soemers

Desired prior knowledge: Probability & Statistics.

Description: Any realistic model of a real-world phenomenon must take into account the possibility of randomness. That is, more often than not, the quantities we are interested in will not be predictable in advance but, rather, will exhibit an inherent variation that should be taken into account by the model. Mathematically, this is usually accomplished by allowing the model to be probabilistic in nature. In this course, the following topics will be discussed:

1. Basic concepts of probability theory: Probabilities, conditional probabilities, random variables, probability distribution functions, density functions, expectations and variances.
2. Finding probabilities, expectations and variances of random variables in complex probabilistic experiments.
3. Discrete and continuous time Markov chains and related stochastic processes like random walks, branching processes, Poisson processes, birth and death processes, queueing theory.
4. Markov decision problems.
5. Multi-armed bandit problems, bandit algorithms, contextual bandits, cumulative regret, and simple regret

Knowledge and understanding: In this course, the students acquire tools for modelling complex processes involving randomness, providing a basis for originality in developing and/or applying ideas in a research context.

Applying knowledge and understanding: When confronted with complex problems that involve probabilistic experiments, students have the tools to create and analyse appropriate models.

Making judgements: The students are able to analyse complex problems as stochastic processes and solve them. Furthermore, students can find optimal solutions in decision problems that are based on these stochastic processes.

Communication: The students will be able to communicate their conclusions and the underlying rationale to expert and non-expert audiences.

Learning skills: The students have obtained the skills to study related material in a largely autonomous manner.

Study material: Introduction to Probability Models by Sheldon M. Ross (9 th or 10th ed.) + Lecture notes that are provided via Student Portal.

Exam: Written exam.

Recommended Literature: Probability: A Lively Introduction by Henk Tijms; Reinforcement Learning by Richard S. Sutton and Andrew G. Barto (2nd ed.) (chapter 2); Bandit Algorithms by Tor Lattimore and Csaba Szepesvári

ECTS: 6

Signal and Image Processing (KEN4222)

Examiners: Dr. Joel Karel & dr. Pietro Bonizzi

Desired prior knowledge: Linear algebra, Calculus, basic knowledge of Matlab. Some familiarity with linear systems theory and transforms (such as Fourier and Laplace) is helpful.

Prerequisites: None.

Description: This course offers the student a hands-on introduction into the area of digital signal and image processing. We start with the fundamental concepts and mathematical foundation. This includes a brief review of Fourier analysis, z-transforms and digital filters. Classical filtering from a linear systems perspective is discussed. Next wavelet transforms and principal component analysis are introduced. Wavelets are used to deal with morphological structures in signals. Principal component analysis is used to extract information from high-dimensional datasets. We then discuss Hilbert-Huang Transform to perform detailed time-frequency analysis of signals. Attention is given to a variety of objectives, such as detection, noise removal, compression, prediction, reconstruction and feature extraction. We discuss a few cases from biomedical engineering, for instance involving ECG and EEG signals. The techniques are explained for both 1D and 2D (images) signal processing. The subject matter is clarified through exercises and examples involving various applications. In the practical classes, students will apply the techniques discussed in the lectures using the software package Matlab.

Knowledge and understanding: Students are able to explain fundamental concepts of signal and image processing and their mathematical foundation. They are able to 1) describe various types of filters and their properties, 2) explain orthogonal wavelet filter banks and describe their properties, 3) explain a construction scheme and elicit a wavelet-based noise-filtering scheme, 4) explain principal component analysis and empirical signal processing techniques and how they complement the other techniques discussed.

Applying knowledge and understanding: Students are able to use the various techniques discussed during the lectures to solve real-world problems, such as being able to apply wavelet filtering and principal component analysis on various signals. They are also able to analyse a signal by using Matlab, and independently interpret the outcome of an analysis.

Making judgements: Students are able to assess what technique is suited for a signal processing problem at hand, and to independently and critically look at a signal or image, and understand if and what type of pre-processing is required.

Communication: Students are able to communicate signal and image processing techniques and strategies, and the results of their analyses to experts and non-experts.

Learning skills: Students are able to independently master signal and image processing techniques, from classical signal processing techniques to more empirical techniques, and they are able to stay up to date with the state of the art in the field.

Study material: Discrete Wavelet Transformations: An Elementary Approach with Applications, Patrick J. Van Fleet, Wiley, ISBN: 978-0-470-18311-3.
Additional material provided electronically on Student Portal.

Recommended literature: Principal Component Analysis, Ian T. Jolliffe, Springer, ISBN13: 978-0387954424.

Exam: Written exam/Computer exam.

ECTS: 6

Period 1.2 & 2.2

Advanced Natural Language Processing (KEN4259)

Examiners: Prof. dr. ir. J.C. Scholtes & Dr. Aki Härmä

Desired prior knowledge: None.

Prerequisites: None.

Description: How do I say, "Where is the next Italian restaurant" in Dutch? Can I actually use speech recognition instead of typing my question? Can I get a summary of today's lecture? Can your chatbot assist me in finding the right information, answer my question or solve my problem? How do I know for sure that the chatbot does not hallucinate? How can I integrate multi-modal information in my language task?

Computers able to answer these questions are a long-time dream of humankind. For many years, computers underperformed using linguistic skills compared to humans. However, the development of Large Language Models (LLM) allowed us to make huge progress and perform at the human level for tasks such as machine translation, Q&A, abstracting, speech recognition, summarization and having a conversation with a computer program.

This course will provide the skills and knowledge to develop state-of-the-art (SOTA) solutions for these natural language processing (NLP) tasks.

After a short introduction to traditional grammatical and statistical approaches to NLP, the course will focus on deep learning techniques to solve these problems. In the first part of the course, we will investigate methods to model basic sequence labeling tasks like Part-of-Speech techniques. The second part of the lecture will focus on deep-learning models to solve many NLP tasks like machine translation, summarization and question answering.

In this course, major challenges when building the systems will be address: representing words in neural networks, neural network architectures to model language, methods to train complex models and algorithms to find the most probable output. Most of the lectures will focus on transformer-based models, both encoder, decoder and encoder-decoder models as well as multi-modal approaches. In addition, we discuss important aspects of Large Language Models (LLM) such as quantitative measuring of quality, fine-tuning LLM's, limitations to prompt engineering, ethics, energy consumption and eXplainable AI (XAI).

The theory discussed in the course is supported by various (Python) tutorials where the students can experience the inner-workings of the algorithms themselves.

Linear Algebra, Statistics, Deep Learning and Natural Language Processing play an important role in this course.

This course is complementary to the course Information Retrieval and Text-Mining. Overlap is reduced to the necessary minimum. Both courses can be followed in any particular order. In the Information Retrieval and Text Mining course we focus more on creating an optimal search experience, in the Advanced Natural Language Processing course, we do a deep dive into the algorithms and models used for different language-related problems such as machine translation, abstracting, and dialogs with chatbots. Tutorials are shared between the two courses.

Knowledge and understanding: Student will be taught traditional approaches in NLP, as well as statistical models. Finally, state-of-the-art (SOTA) deep learning techniques for natural language processing (including multi-modal information), including understanding methods to evaluate the performance of such models. They will learn techniques to address the major challenges when building a natural language processing tool, including explainability and more efficient energy usage of such models.

Appling knowledge and understanding: The achievements in deep learning have significantly improved the quality of state-of-the-art methods for natural language processing. With the knowledge acquired in the course, students will be able to build SOTA solutions. Students will also understand why deep learning models are outperforming traditional grammatical and statistical models and what the limitations and risks are of deep learning models in terms of applying explainable AI and more energy-efficient models.

Making judgements: Students will be able to analyze the specific challenges of a task in NLP. Based on the gather knowledge on different ways to model tasks they are able to select and implement a fitting model to solve the task.

Communication: Through reporting on tutorials, students will be enabled to communicate their findings and explain the rationale behind their choices in deep learning techniques for natural language processing.

Learning skills: After successful completion of the course, students will be able to develop natural language processing tools and perform research on new ideas in the field.

Study material: Mostly based on the lecture notes and the provided material including recent papers published in this field. We will also provide references to a number of good books that are on-line available for more background information.

Recommended literature: Papers published in top international conferences and journals in machine learning field.

Assessment: Participation in the Tutorials (30%), final exam (70%). The exam is open book.

ECTS: 6

Network Science (KEN4275)

Coordinator: Adriana Iamnitchi

Desired prior knowledge: Introductory knowledge of programming for data analysis, particularly in Python; algorithms; algorithmic complexity. Introductory courses in algorithms, data structures, and data analytics.

Prerequisites: None.

Description: Many aspects of everyday life and science can be represented as networks: social networks represent relationship (links) between people (nodes); brain activity can be represented via synapses (links) between neurons (nodes); the street map is formed of roads (links) that connect at intersections (nodes); authors of scientific papers connect to each others in a citation network, with directed links from the paper cited to the paper citing it; communication networks connect routers via physical or logical links; etc. Network analysis plays a significant role in the "big data" analytics because of size, data velocity, or computational complexity.

This course focuses on the study of network structures and dynamic processes on networks using real data from various disciplines, including socio-technological platforms, biology, social science, and economics. Topics cover the analysis and modeling of complex networks, network dynamics, community detection, network resilience and contagion, as well as processing of network structures for machine learning tasks.

Formal models that will be investigated: random graphs, scale-free networks, preferential attachment model, Watts and Strogatz model, epidemics models.

Knowledge and understanding: Students will acquire a solid understanding of the key concepts and terminology in network science, will comprehend the theoretical underpinnings of various network models, and recognize relevant network characteristics across different contexts and applications.

Applying knowledge and understanding: Students will employ computational tools to model, analyze, and visualize networks from various real-world sources, and implement simulations to study network dynamics and evolution.

Making judgements: Students will critically assess different network models and their applicability to real-world problems. They will evaluate the implications of network structure on system dynamics. They will evaluate the benefits and limitations of various network embedding techniques for machine learning tasks.

Communication: Students will be able to present complex network concepts clearly to both specialist and non-specialist audiences and collaborate effectively in teams on network analysis projects.

Learning skills: In addition to the guiding material formally provided in the course, students will research independently from various sources.

Study material: Will be provided throughout the lecture.

Assessment: Assignments and group project.

Recommended literature: “Networks, Crowds, and Markets: Reasoning About a Highly Connected World” by David Easley and Jon Kleinberg.

Additional literature: Graph Theory and Complex Networks: An Introduction by Maarten van Steen

ECTS: 6

Period 1.4 & 2.4

Explainable AI (KEN4246)

Coordinators: Prof. Dr. Nava Tintarev & Dr. Tjitze Rienstra

Examiners: Prof. Dr. Nava Tintarev & Dr. Tjitze Rienstra

Tutor: Aashutosh Ganesh

Desired prior knowledge: Data Analysis and Data Mining or ACML

Prerequisites: None.

Description: A key component of an artificially intelligent system is the ability to explain to a human agent the decisions, recommendations, predictions, or actions made by it and the process through which they are made. Such explainable artificial intelligence (XAI) can be required in a wide range of applications. For example, a regulator of waterways may use a decision support system to decide which boats to check for legal infringements, a concerned citizen might use a system to find reliable information about a new disease, or an employer might use an artificial advice-giver to choose between potential candidates fairly. For explanations from intelligent systems to be useful, they need to be able to justify the advice they give in a human-understandable way. This creates a necessity for techniques for automatic generation of satisfactory explanations that are intelligible for users interacting with the system. This interpretation goes beyond a literal explanation. Further, understanding is rarely an end-goal. Pragmatically, it is more useful to operationalize the effectiveness of explanations in terms of a specific notion of usefulness or explanatory goals such as improved decision support or user trust. One aspect of intelligibility of an explainable system (often cited for domains such as health) is the ability for users to accurately identify, or correct, an error made by the system. In that case it may be preferable to generate explanations that induce appropriate levels of reliance (in contrast to over- or under-reliance), supporting the user in discarding advice when the system is incorrect, but also accepting correct advice.

The following subjects will be discussed:
1. Intrinsically interpretable models, e.g., decision trees, decision rules, linear regression.
2. Identification of violations of assumptions, such as distribution of features, feature interaction, non-linear relationships between features; and what to do about them.
3. Model agnostic explanations, e.g., LIME, scoped Rules (Anchors), SHAP (and Shapley values)
4. Ethics for explanations, e.g., fairness and bias in data, models, and outputs.
5. Symbolic approaches to AI
6. (Adaptive) User Interfaces for explainable AI
7. Evaluation of explanation understandability

Knowledge and understanding: Students can explain the difference between different explanation approaches (e.g., global versus local models) and can identify which are suitable to use based on underlying assumptions and relative advantages and limitations.

Applying knowledge and understanding: Students can critically choose and apply XAI methods. Students can formulate evaluation protocols to validate the understandability of explanations, demonstrating awareness of the ethical, normative, and social consequences of their applications.

Making judgements: Students will be able to critically evaluate the quality (rigor of methodology), and ethical consequences, of approaches (systems or scientific publications) based on the XAI techniques taught.

Communication: Students will be able to communicate their ideas effectively in written form. They will be able to actively contribute to group-wise communication, and in both oral and written form present their models and outputs to specialists.

Learning skills: Students will be able to familiarize themselves, and critically assess XAI techniques beyond the scope of the course in order to solve a problem.

Study material: Course notes, required reading of scientific articles.

Assessment: Group project and individual written assignment

Recommended literature:
- Molnar, Christoph. Interpretable Machine Learning (second edition). Lulu.com, 2022 (available free online)
- Rothman, Denis. Hands-On Explainable AI (XAI) with Python: Interpret, visualize, explain, and integrate reliable AI for fair, secure, and trustworthy AI apps, Packt, 2020.

ECTS: 6

Dynamic Game Theory (KEN4251)

Examiners: Prof. dr. Frank Thuijsman & dr. Monica Salvioli

Desired prior knowledge: Students are expected to be familiar with basic concepts from linear algebra, calculus, Markov chains and differential equations.

Prerequisites: None.

Description: The course will focus on non-cooperative games and on dynamic games in the following order: matrix and bimatrix games, repeated games, differential games, specific models of stochastic games, Stackelberg games, games in extensive form and evolutionary games. These are games in which the players are acting as strategic decision makers, who cannot make binding agreements to achieve their goals. Instead, threats may be applied to establish stable outcomes. Besides, relations with population dynamics and with “learning” will be examined. Several examples will be taken from biological settings.

Knowledge and understanding: Students are able to recognize and classify the main types of dynamic games, i.e. repeated games, stochastic games, Stackelberg games, differential games, and evolutionary games and formulate the main solution concepts value, optimal strategies, Nash- and Stackelberg equilibrium

Applying knowledge and understanding: Students are able calculate solutions of the different types of dynamic games.

Making judgements: Students are able to explain advantages and disadvantages of different solution concepts. They are able to judge correctness of solutions presented.

Communication: Students are able to explain and defend correctness of their solutions.

Learning skills: By the end of the course, students will be able to autonomously and critically reflect upon the pros and cons of different types of games for modelling competition and cooperation. This includes considerations on the computational aspects with respect to different solution concepts.

Study material: Handouts will be provided.

Exam: There will be a closed book written exam at the end of the course.

ECTS: 6

Planning and Scheduling (KEN4253)

Examiner: Dr. Steven Kelk

Desired prior knowledge: Data Structures & Algorithms. Discrete Mathematics. Graph Theory

Prerequisites: None.

Description: In many real-world processes, particularly in industrial processes and logistics, decisions need to be taken about the time of the completion of (sub)tasks, and the decision about what production machines complete the tasks. There are often constraints on the order in which tasks, or 'jobs' can be performed, and there are usually capacity constraints of the machines. This leads to natural, industrially critical optimization problems. For example, a company might choose to buy many machines to process jobs, but then there is a risk that the machines will be underused, which is economically inefficient. On the other hand, too few machines, or an inappropriate ordering of tasks, may lead to machines spending a significant amount of time standing idle, waiting for the output of other machines, which are overcrowded with tasks. In this course, we look at various mathematical models and techniques for optimizing planning and scheduling problems, subject to different optimality criteria. We will discuss, among others, single-machine models, parallel-machine models, job-shop models, and algorithms for planning and scheduling (exact, approximate, heuristic) and we also touch upon the computational complexity (distinguishing between 'easy' and 'difficult' problems) of the underlying problems. Last but not least, we will also introduce integer linear programming as a uniform and generic tool to model and solve planning and scheduling problems.

Knowledge and understanding: Students will possess the mathematical and algorithmic tools to model and solve planning/scheduling problems. Students will be able to recognize real-world problems in the unified theory and established language of planning and scheduling.

Applying knowledge and understanding: Students will be able to apply the new techniques to various problems arising in real-world applications. Students will be able to deploy the standard algorithmic techniques, and be able to design new algorithmic solutions, and to argue about their performance properties.

Making judgements: Students will understand under which circumstances different planning/scheduling problems are computationally tractable, and will judge algorithmic technique can be used to exactly or approximately solve these problems.

Communication: Students will be able to analytically argue about correctness of the used algorithmic approaches. Students will be able to explain modelling approaches to planning and scheduling problems in the language of the theory of planning and scheduling.

Learning skills: Students will enhance their study skills such as time management, effective reading, critical thinking and reading, exact and unambiguous writing and formulating of ideas and statements, and reflection on marked work. Along the way, students will improve general learning skills such as self-motivation, careful listening and giving instructions, and openness to new knowledge. Students will also be exposed to autonomous self-study.

Study material: Appropriate study material will be provided throughout the course.

Assessment: Written exam (75%) at the end of the course, and graded exercises (25%) throughout the course.

ECTS: 6

Building and Mining Knowledge Graphs (KEN4256)

Examiner: Prof. dr. Michel Dumontier

Desired prior knowledge: Introduction to Computer Science

Prerequisites: None.

Description: Knowledge graphs are, seen through a semantic-web lens, large-scale, machine-processable representations of entities, their attributes, and their relationships. Knowledge graphs enable both people and machines to explore, understand, and reuse information in a wide variety of applications such as answering questions, finding relevant content, understanding social structures, and making scientific discoveries. However, the sheer size and complexity of these graphs present a formidable challenge particularly when mining across different topic areas.

In this course, we will examine approaches to construct and use knowledge graphs across a diverse set of applications using cutting-edge technologies such as machine learning and deep learning, graph databases, ontologies and automated reasoning, and other relevant techniques in the area of data mining and knowledge representation.

Knowledge and understanding: Students will be able to:
- Define and describe the nature and attributes of a Knowledge Graph
- Identify and describe the components of a Knowledge Graph
- Distinguish between different representations for Knowledge Graphs
- Describe applications of Knowledge Graphs
- Identify advantages and disadvantages of Knowledge Graphs as compared to other formalisms
- Describe and execute approaches to construct and maintain Knowledge Graphs from structured and unstructured sources, across different domains
- Construct and query Knowledge Graphs to answer questions about their content using open standards such as RDF and SPARQL
- Use Large Language Models to construct Knowledge Graphs, and to retrieve their contents
- Execute link prediction and associated graph mining techniques to enrich information in Knowledge Graphs
- Describe the FAIR principles and construct Knowledge Graph metadata using available standards
- Describe Knowledge Graph quality metrics and evaluate the quality of a Knowledge Graph
- Develop own Knowledge Graph solution for a problem of interest

Applying knowledge and understanding: Students will be able to identify requirements and steps to convert knowledge in traditional data formats to Knowledge Graph formats. Students will also be able to implement such strategies. Students will be able to query Knowledge Graphs (for instance using SPARQL query language) to answer basic to intermediately advanced questions. Students will be able to implement basic reasoning strategies on Knowledge Graphs to answer intermediately advanced questions, which cannot be answered by SPARQL queries alone. Students will be able to implement popular methods to integrate different data sources by transferring them into a Knowledge Graph. Students will be able to enrich existing Knowledge Graphs with missing information using basic predictive algorithms. Students will be able to perform basic data quality assessment on Knowledge Graphs. Students will be able to assess the degree of compliance that Knowledge Graphs have with FAIR principles.

Making judgements: Students will be able to select which tools are most suitable for constructing, querying, visualising & reasoning with Knowledge Graphs. Students will be able to differentiate between different types of Knowledge Graphs, according to their representation, coverage and content. Students will be able to select which Knowledge Graph is appropriate for answering a particular question. Students will be able to diagnose incompleteness in a Knowledge Graph with respect to answering a particular question. Students will be able to evaluate the data quality and FAIRness of a Knowledge Graph.

Communication: Students will be able to explain the advantages of representing information on the web in Knowledge Graphs. Students will be able to communicate the steps required to convert information to a Knowledge Graph format. Students will be able to communicate to non-experts the main content and representational components of a Knowledge Graph. Students will be able to outline to non-experts the steps required to answer a question by querying a Knowledge Graph.

Learning skills: Students will be able to reflect critically on the challenges and open problems remaining in Knowledge Graphs research. Students will be able to formulate and propose strategies to answer complex questions using Knowledge Graphs. Students will be able to assess the feasibility of different combinations of methods for answering questions using Knowledge Graphs.

Study material: Slides for the labs and lectures will be released on Canvas just before the respective session in PDF format.

Assessment: Individual project for application of knowledge and three group assignments to demonstrate understanding of core concepts. Assessments will be released in PDF format on Canvas according to the dates indicated in the previous slide for evaluation.

Recommended literature: Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., ... & Zimmermann, A. (2021). Knowledge graphs. ACM Computing Surveys (Csur), 54(4), 1-37.
Additional literature: A Semantic Web Primer. 3rd Edition. Grigoris Antoniou, Paul Groth, Frank van Harmelen and Rinke Hoekstra. 2012. MIT Press, ISBN: 9780262018289.
Semantic Web for the Working Ontologist. 3rd Edition. James Hendler, Fabien Gandon, Dean Allemang. 2020. Morgan Kaufmann. ISBN-13: 978-1450376174; ISBN-10: 1450376177.
Practical RDF. Shelley Powers. 2003. O'Reilly Media, Inc. ISBN: 9780596002633.
Learning SPARQL. Bob DuCharme. 2011. O'Reilly media, Inc. ISBN: 9781449306595.
Programming the Semantic Web. Toby Segaran, Colin Evans, Jamie Taylor. 2009. O'Reilly Media, Inc. ISBN: 9780596153816.

ECTS: 6

Period 1.5 & 2.5

Information Retrieval and Text Mining (KEN4153)

Examiner: Prof dr. ir. J.C. Scholtes

Desired prior knowledge: None.

Prerequisites: None.

Description: Normal search is about "Finding the needle in the haystack". This course focusses on a more complex problem: "How does the needle look like and where is the haystack? Also explain me why!"

Building a full-text search engine may look trivial, but it is not! How do you search hundreds of billions of documents that can be located anywhere, with sub-second responds times? How do you find exactly what you are looking for without missing relevant information or having to plough through hundreds of irrelevant documents? How can you find if you do not know exactly what you are looking for? How can you find information which is deliberately hidden? How do you know that your search engine has given you the right information? Where does it come from? Is the answer factually correct?

In this course, we will teach you how to address these questions in three steps: (1) how is a search engine is constructed, optimized and used effectively, (2) How can techniques from the word of text-mining, information extraction, text classification, clustering, topic modeling and data visualization add to a better search experience, and (3) What is the best way to integrate chatbots with search engines. How to best guarantee factuality, avoid hallucinations and provide provenance and explainability of the chatbots' recommendations.

Linear Algebra, Statistics, Deep Learning and Natural Language Processing play an important role in this course.

This course is complementary to the course Advanced Natural Language Processing (ANLP). Overlap is reduced to the necessary minimum. Both courses can be followed in any particular order. In the Information Retrieval and Text Mining course we focus more on creating a optimal search experience, in the Advanced Natural Language Processing course, we do a deep dive into the algorithms and models used for different language-related problems such as machine translation, abstracting, and dialogs with chatbots. Tutorials are shared between the two courses.

Knowledge and understanding: The student will be able to select, understand and apply different phases and methods used to create applications that exhibit an optimal search experience or provide excellent analytical insights for natural language. In addition, the student learns to evaluate the quality of such methods according to best-practice standards as used in the field.

Applying knowledge and understanding: Students will be able to recognize applications of text mining, information retrieval and conversational AI in different domains such as consumer search, legal services, medical research, regulatory oversight, compliance, digital humanities, and customer services. After the course, the student can formulate an opinion or course of action when dealing with text-based AI-problems based on incomplete, limited and in part unreliable information. After the course, students can apply their knowledge and understanding in a manner that shows a scientific approach to their work or vocation. They are able to handle complex and ill-defined text-based problems for which it is not a priori known if there is an appropriate solution, they know how to acquire the necessary information to solve the sub-problems involved, and they know how to proceed with problems for which there is no standard or reliable route to the solution.

Making judgements: Upon completion of the course, students are able to recommend the most appropriate methods from the fields of text mining, information retrieval and conversational AI when confronted with problems involving search and analysis of textual unstructured data.

Communication: Students are able to communicate the (dis)advantages of several methods from the field of text mining and information retrieval to both an audience of non-experts.

Learning skills: After the course, the student has developed those learning skills that are necessary for a successful further career in text mining or information retrieval at the highest professional level. The student will be able to continue to develop their text-mining and information retrieval skills. The student is able to detect missing knowledge and abilities and to deal with them appropriately by finding and consulting resources that can help them to fill the gaps and new developments.

Study material: A syllabus and copies of the course slides will be used.

Recommended literature (not mandatory): Introduction to Information Retrieval. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. Cambridge University Press, 2008. In bookstore and online: http://informationretrieval.org and Feldman, R., and Sanger, J. (2006). The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press.

Assessment: The result of various Colab tutorials and a project contributes 30% to the final examination of the course. The other 70% is determined by the theoretical exam. The theoretical exam is open book. For the project, students can select a research topic and a text corpus from the provided list (or another relevant open source collection) and implement a number of relevant search, text-mining or conversational AI operations by using the methods discussed in the course. The delivery of the project are the results of the experiments (presented at the end of the course) and a report discussing the methods used and the quantitative quality of the efforts undertaken.

ECTS: 6

Introduction to Quantum Computing for AI and Data Science (KEN4155)

Examiners: Dr. Menica Dibenedetto & Dr. Georgios Stamoulis

Desired prior knowledge: Probability theory, linear algebra, design and analysis of algorithms

Prerequisites: None.

Description: In this course, we lay down the foundations and basic concepts of quantum computing. We will use the mathematical formalism borrowed from quantum mechanics to describe quantum systems and their interactions. We introduce the concept of a quantum bit and discuss different physical realizations of it. We then introduce the basic building blocks of quantum computing: quantum measurements and quantum circuits, single and multi-qubit gates, the difference between correlated (entangled) and uncorrelated states and their representation, quantum communication, and basic quantum protocols and quantum algorithms. Finally, we discuss the different types of noise involved in real quantum computers (coherent and incoherent errors, state preparation, projection and measurement) and their effect on performance, and outline current efforts for mitigating the issues.

Knowledge and understanding: Students will learn the fundamental principles and concepts behind quantum computing, protocols, and algorithms. Students will understand the differences between classical and quantum computation, and where the (theoretical) computational power of quantum machines comes from. Students will also get to understand the current challenges in building and using quantum computers.

Applying knowledge and understanding: Students will be able to apply existing quantum algorithms as black-box to various simple computational problem. Students will further be able to analyse simple quantum algorithms for different computational problems.

Making judgements: Students will be able to judge how the potential computational power of quantum computing can be leveraged, and how it can be applied to other fields in a beneficial way.

Communication: Students will be able to discuss quantum computation critically and judge not only its benefits but, equally important, its shortcomings. Students will especially be able to communicate potential benefits of quantum computation to the fields of artificial intelligence and data science.

Learning skills: Students will practice learning entirely new computational concepts, and how to relate existing concepts (classical computation) to new concepts (quantum computation). Students will learn to critically reflect on both the scientific literature and the societal expectations. Students will learn to self-study from state-of-the-art research articles, when classical text-books are not available.

Study material: To be announced.

Exam: Written exam (100%)

ECTS: 6

Reinforcement Learning (KEN4157)

Coordinators: Dr. ir. Kurt Driessens & Dr. Dennis Soemers

Examiners: Dr. ir. Kurt Driessens & Dr. Dennis Soemers

Desired prior knowledge: Machine Learning

Prerequisites: None.

Description: Reinforcement learning is a type of machine learning problem in which the learner gets a (delayed) numerical feedback signal about its demonstrated performance. It is the toughest type of machine learning problem to solve, but also the one that best encompasses the idea of artificial intelligence as a whole. In this course we will define the components that make up a reinforcement learning problem and will see what the important concepts are when trying to solve such a problem, such as state and action values, policies and performance feedback. We will look at the different properties a reinforcement learning problem can have and what the consequences of these properties are with respect to solvability. We will discuss value based techniques as well as direct policy learning and learn how to implement these techniques. We will study the influence of generalisation on learning performance and see how supervised learning (and specifically deep learning) can be used to help reinforcement learning techniques tackle larger problems. We will also look at the evaluation of learned policies and the development of performance over time.
Formal models that will be investigated: Markov Decision Processes

Knowledge and understanding: Students will be able to explain the setup of a reinforcement learning problem and list its formal components, explain the difficulties faced when adding function approximation to reinforcement learning, explain the origins of the learning signal for policy gradient methods for reinforcement learning.

Applying knowledge and understanding: Students can implement and apply online and offline tabular techniques of value based reinforcement learning algorithms, apply the use of function approximation in value based reinforcement learning algorithms, implement and apply policy gradient methods for discrete and continuous action tasks and deep learning methods to reinforcement problems.

Making judgements: Students will be able to judge the suitability of reinforcement learning techniques as a solution for an AI problem, choose/select between exploration and exploitation tradeoff methods suited to the problem faced, interpret and judge the results of a reinforcement learning agent.

Communication: Students will gain a working knowledge of reinforcement learning as a problem, and of the state of the art in solution techniques and will be able to motivate his/her choices concerning the application of these techniques.

Learning skills: Students will learn that the state of the art in reinforcement learning continues to develop at a rapid pace and that becoming and staying an expert in the domain will require continued learning.

Study material: Course slides to support the lectures; supplementary material consisting of research papers and book chapters.

Assessment: Assessment for this course is based on the construction of a portfolio with which students prove that they attained all learning goals, at which point they will pass the course. The level of a passing grade is determined by the quality of a large final research and implementation assignment.

Recommended literature: Reinforcement Learning: An Introduction by R. Sutton and A. Barton

Additional literature: Algorithms for Reinforcement Learning by C. Szepesvári; Reinforcement Learning and Optimal Control by D. Bertsekas

ECTS: 6

Computer Vision (KEN4255)

Coordinator & examiner: Dr. Mirela Popa

Tutors: Subilal Vattimunda Purayil, Aashutosh Ganesh

Desired prior knowledge: Basic knowledge of Python, linear algebra and machine learning. This course offers the basics on image processing although prior knowledge is also a plus.

Prerequisites: None.

Description: Can we make machines look, understand and interpret the world around them? Can we make cars that can autonomously navigate in the world, robots that can recognize and grasp objects and, ultimately, recognize humans and communicate with them? How do search engines index and retrieve billions of images? This course will provide the knowledge and skills that are fundamental to core vision tasks of one of the fastest growing fields in academia and industry: visual computing. Topics include introduction to fundamental problems of computer vision, mathematical models and computational methodologies for their solution, implementation of real life applications and experimentation with various techniques in the field of scene analysis and understanding. In particular, after a recap of basic image analysis tools (enhancement, restoration, color spaces, edge detection), students will learn about feature detectors and trackers, fitting, image geometric transformation and mosaicing techniques, texture analysis and classification using unsupervised techniques, face analysis, deep learning based object classification, detection and tracking, camera models, epipolar geometry and 3D reconstruction from 2D views.

Knowledge and understanding: Students will be able to apply the most suitable techniques for image pre-processing (e.g. enhancement, restoration), feature extraction, texture analysis, perspective geometry, camera models and topics on object recognition. In addition, they will be able to identify the most suitable techniques in a series of visual computing problems.

Applying knowledge and understanding: Students will be able to choose and/or construct solutions in a variety of professional/vocational contexts requiring image processing and computer vision (robotics, manufacturing, AI, web applications, surveillance). They will be able to build and assess methodologies for handling real-world complex problems in computer vision, making use of pre-existing data for training their models.

Making judgements: Students will be able to choose and combine methods to tackle real-world computer vision problems, captured in real-life settings and having no obvious solutions. They will be able to propose and build techniques combining computer vision methods with machine learning instruments for scene understanding and object recognition.

Communication: Through small research projects, students will be able to communicate their findings and explain the rationale behind their choices in computer vision techniques for image/video analysis.

Learning skills: After successful completion of the course, students will be able to analyze images and videos and retrieve or process content in order to derive useful information, applicable in a variety of domains (e.g. satellite imagery, surveillance, robotics, medical imaging, ambient assisted living).

Study material: A syllabus and copies of the course slides will be used along with the recommended literature.

Assessment: Written exam (50%) and two assignments (50%)

Recommended literature:
- “Computer vision: algorithms and applications”. Szeliski, Richard. Springer Science & Business Media, 2010 (available online)
- “Computer Vision: A Modern Approach, 2nd Edition”. David A. Forsyth, University of Illinois at Urbana-Champaign .Jean Ponce, Ecole Normale Superieure, Paris
- “Computer Vision: Models, Learning and Inference”, Simon J.D. Prince 2012.

Additional literature:
- “Digital Image Processing”, Rafael C. Gonzalez & Richard E. Woods, Pearson, 3rd Edition, 2016.
- “Machine Vision: Automated Visual Inspection and Robot Vision”, David Vernon, Prentice Hall, (available online at: https://homepages.inf.ed.ac.uk/rbf/BOOKS/VERNON/)
- OpenCV/Pytorch/Tensor Flow tutorials (https://docs.python.org/3/tutorial/,https://pytorch.org/tutorials/,https://colab.research.google.com, https://docs.opencv.org/4.x/d9/df8/tutorial_root.html)

ECTS: 6

Mathematical Optimization (KEN4211)

Coordinator & examiner: Dr. Pieter Collins

Desired prior knowledge: Calculus, Linear Algebra, Linear Programming

Prerequisites: None.

Description: Optimization is the subject of finding the best or optimal solution to a problem from a set of potential or feasible solutions. Optimization problems are fundamental in all forms of decision-making, since one wishes to make the best decision in any context, and in the analysis of data, where one wishes to find the best model describing experimental data. This course treats two different areas of optimization: nonlinear optimization and combinatorial optimization, building on knowledge of linear programming using the simplex algorithm. Together, nonlinear and combinatorial optimization cover a wide range of real life optimization problems. Nonlinear optimization deals with the situation that there is a continuum of available solutions. A best solution is then usually approximated with one of several available general purpose algorithms, such as Brent's method for one-dimensional problems, (quasi-)Newton and conjugate gradient methods for unconstrained problems, and Lagrangian methods, including active-set methods, sequential quadratic programming and interior-point methods for general constrained problems. Combinatorial optimization deals with situations that a best solution from a discrete set of available choices must be found. A variety of techniques, such as linear programming, branch and cut, Lagrange relaxation and approximation algorithms are employed to tackle this type of problem. Throughout the course, we aim to provide a coherent framework for the subject, with a focus on optimality conditions (notably the Karush-Kuhn-Tucker conditions), Lagrange multipliers and duality, relaxation and approximate problems, and on convergence rates and computational complexity. The methods will be illustrated by in-class computer demonstrations, exercises illustrating the main concepts and algorithms, modelling and computational work on case studies of practical interest, and a discussion of advanced stochastic and batch optimization methods for machine-learning.

Formal models that will be investigated: Unconstrained and constrained nonlinear programming problems; integer-linear programming problems.

Knowledge and understanding: By the end of this course, students will have a strong foundation in nonlinear and combinatorial optimization. You will be able to formulate real-life problems as optimization problems. You will understand optimality conditions, including the Karush-Kuhn-Tucker conditions and be able to test for optimality. You will know how to solve a variety of general optimization problems, including constrained nonlinear problems, and (mixed-)integer linear problems. You will understand notions of duality and Lagrange multipliers, and be able to apply techniques based on relaxation and approximation.

Applying knowledge and understanding: Students will know the advantages and disadvantages of different methods, and be able to choose an appropriate method for a given problem. You will be able to implement and test optimization algorithms on a computer. You will be able to apply your knowledge to the solution of practical problems and in developing new efficient algorithms.

Making judgements: Students will be able to select an appropriate solution method for a given optimization problem, and judge the quality of the solution obtained.

Communication: Students will be able to discuss the development and use of optimization algorithms.

Learning skills: Students will learn how to develop and implement mathematical methods for optimization, select and evaluate algorithms, and formulate mathematical model of real-world problems.

Study material: Lecture notes, handouts.

Exam: Written examination, closed book (100%).

Recommended literature: J. Nocedal and S.J. Wright, "Numerical Optimization", Springer, 2006; ISBN: 978-0-387-30303-1. C.H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", Dover Publications, 1998; ISBN: 978-0-13152-462-0. W.J. Cook, W.H. Cunningham, W.R. Pulleyblank and A. Schrijver, "Combinatorial Optimization", Wiley-Interscience, 1998; ISBN: 978-0-47155-894-1.

ECTS: 6

Quantum Algorithms (KEN4235)

Examiner: Dr. Georgios Stamoulis

Desired prior knowledge: Fundamentals of Quantum Computation, Very Good command of Linear Algebra, Algorithms and Complexity

Prerequisites: Introduction to Quantum Computing for AI and Data Science

Description: This course will provide a thorough examination of the most important Quantum Algorithms. We will see how the quantum mechanical formalism gives rise to a new algorithmic design paradigm with the potential of performing certain computational tasks faster than we could do using a classical computer. The course will start with some basic algorithms like Bernstein-Vazirani and Simon's algorithm, then we will move on to Quantum Fourier Transform and Phase Estimation. Then, a thorough discussion of Shor's celebrated algorithm for factoring will follow, together with a detailed coverage of Grover's unstructured search algorithm, its optimality, adaptations, and applications. Further, we will move on to the HHL algorithm for solving systems of linear equations, a crucial component of many quantum algorithms, including Machine Learning quantum algorithms. In the last part of the course, we will present algorithms for quantum simulation, discuss quantum walks, and basics of quantum complexity theory by introducing and discussing the BQP and QMA complexity classes.

Knowledge and understanding: Students will learn how and why certain computational tasks can be performed faster in a quantum computer, what are the major techniques used in the design of such algorithms and, equally important, what are the limitations of the quantum algorithm design.

Applying knowledge and understanding: Students will be able to apply these theoretical techniques to design and analyze algorithms for many problems that could benefit from a quantum computation point of view.

Making judgements: Students will be able to judge whether proposed quantum algorithms indeed offer speedups over classical ones and how they may be able to achieve that.

Communication: Students will practice technical communication of research work in this area, describing and critically evaluating the work's contributions.

Learning skills: Students will learn from lectures/textbook/notes and then use this knowledge to read relevant research papers.

Study material: Lectures, textbook.

Assessment: 75% final exam, 25% in-class presentation summarizing a topic of relevant interest in class.

Recommended literature: Quantum Computation and Quantum Information: 10th Anniversary Edition Anniversary Edition, Michael A. Nielsen, and Isaac L. Chuang

Additional literature: Papers, notes and other relevant material will be distributed in class.

ECTS: 6

Period 2.2

Quantum AI (KEN4236)

Coordinator & examiner: Dr. Menica Dibenedetto

Tutor: Vincenzo Lipardi

Desired prior knowledge: Linear Algebra, Classical Machine Learning

Prerequisites: Introduction to Quantum Computing for AI and Data Science

Description: This course explores the groundbreaking intersection of quantum computing and artificial intelligence, focusing on how quantum technologies can potentially revolutionize AI paradigms. The curriculum delves into quantum algorithms tailored for AI tasks, addressing complex problems that are currently intractable for classical computers. Students will gain an understanding of how quantum principles can enhance machine learning algorithms, improve optimization tasks, and facilitate data processing capabilities. Through theoretical lessons and practical laboratory sessions, students will learn about quantum mechanics fundamentals applicable to AI, quantum circuit design, and quantum algorithm development. Special emphasis will be placed on hybrid models that integrate classical and quantum computing techniques to solve real-world problems. The course will provide a mix of both theoretical and technical insights, as well as practical implementation details by using the main quantum programming languages and quantum software available.

Formal models that will be investigated: Various QAI algorithms and their possible applications for near term devices will be presented. The students will be guided through the steps of creating effective quantum models for supervised and unsupervised tasks and its evaluation in the near-term devices. Discussions will include essential quantum AI algorithms and quantum generalizations of classical learning models. Various quantum machine learning models including quantum neural networks, quantum support vector machines and quantum kernel estimator will be discussed in detail. Quantum algorithms for decision problems based on Hamiltonian time evolution, quantum search models based on Grover algorithm and quantum game theory will be introduced. A significant focus will be on developing efficient methods for encoding data into quantum states, one of the main problems in the current state of machine learning. We will then explore quantum machine learning algorithms applied to state preparation focusing on loading the underlying probability distribution of the dataset, as Quantum Generative Adversarial Networks (GANs) and Quantum Boltzmann Machine.

Knowledge and understanding: Students will gain a deep understanding of the intersection between artificial intelligence and quantum computing, learning to assess the capabilities and limitations of current quantum technologies in enhancing AI applications, exploring different quantum machine learning models.

Applying knowledge and understanding: Learners will apply theoretical concepts in practical settings, developing and implementing quantum algorithms and models that can be applied to machine learning and optimization real-world problems.

Making judgements: Students will evaluate the effectiveness of quantum AI solutions, making informed decisions about when and how to implement these technologies. This course will formulate and answer the questions: "how quantum computing can provide a computation boost to AI, enabling it to tackle more complex problems?" and "how can AI produce functional applications with quantum computers?".

Communication: Participants will enhance their ability to articulate complex quantum AI concepts clearly and effectively to a diverse audience, including those without a background in quantum physics.

Learning Skills: Learners will develop critical thinking and problem-solving skills in quantum computing and AI, fostering a mindset of continuous learning and adaptation to new technologies.

Study Material: Quantum Machine Learning Texts, Online Quantum Computing Simulators, Peer-reviewed Journal Articles

Assessment: Assignment based

Recommended literature: "Machine Learning with Quantum Computers" by M. Schuld, F. Petruccione, Second Edition

Additional literature: Research articles and papers will be provided throughout the course.

ECTS: 6

Quantum Information and Security (KEN4237)

Examiner: Dr. David Mestel

Desired prior knowledge: Quantum states, operators and measurements

Prerequisites: Introduction to Quantum Computing for AI and Data Science

Description: In this course we will consider the power of quantum mechanics not in accomplishing computational or 'algorithmic' tasks, but instead for communication- and security-related tasks. The strange properties of the quantum world turn out to be remarkably useful for these. For example, we can exchange secret messages in a way that is unconditionally secure: secrecy is guaranteed by the physical laws of nature, rather than (as in ordinary cryptography) based on an assumption that a particular computational problem is too hard for the adversary.

We will begin by covering the theoretical techniques needed to study security-related protocols, where it is fundamental that some parties will not know what state a particular quantum system is in. After a thorough grounding in the 'density matrix' formalism which is used to represent this uncertainty, we will cover quantitative measures of this kind of uncertainty, for instance quantum versions of classical entropy. We will then look at a variety of protocols (e.g. quantum money, quantum key distribution,...), and how to define and prove the desired properties.

Knowledge and understanding: Students will learn to use the `density matrix' formalism to reason about quantum states under uncertainty, and about quantitative measures of uncertainty and entanglement.  They will also learn about the fundamentally `contextual' nature of quantum mechanics, which is the foundation for all of the protocols we will study.

Applying knowledge and understanding: Students will be able to apply these theoretical techniques to analyse protocols and prove that they have desirable security properties.

Making judgements: Students will be able to judge whether protocols are suitable for particular goals.

Communication: Students will practice technical communication of research work in this area, describing and critically evaluating the work's contributions.

Learning skills: Students will learn from lectures/textbook and then use this knowledge to read a research paper which they will present in class.

Study material: Lectures, textbook.

Assessment: 70% final exam, 30% in-class presentation summarising a research paper in the topic.

Recommended literature: T. Vidick and S. Wehner, `Introduction to quantum cryptography', Cambridge University Press 2024

Additional literature: M. Wilde, `Quantum information theory', Cambridge University Press 2013

ECTS: 6

Model Identification and Data Fitting (KEN4242)

Examiners: Prof. dr. ir. Ralf Peeters & dr. ir. Philippe Dreesen

Desired prior knowledge: Basic knowledge of Matlab and some familiarity with linear systems theory and transforms (such as Fourier and Laplace) is helpful. This course offers a useful prior knowledge for the course Symbolic Computation and Control. Linear Algebra, Mathematical Modelling, Probability and Statistics.

Prerequisites: None.

Course description: This course is devoted to the various practical and theoretical aspects which involve the estimation (the identification) of a mathematical model within a given model class, starting from a record of observed measurement data (input-output data). First, we address distance measures, norms, and criterion functions. Then we discuss the prediction error identification of linear regression models, with special emphasis on the various interpretations of such models (deterministic, stochastic with Gaussian white noise and maximum likelihood estimation, stochastic in a Bayesian estimation context) and on numerical implementation aspects (recursion, numerical complexity, numerical conditioning and square root filtering). Next, we study identification within the important class of auto-regressive dynamical models, to which the Levinson algorithm applies. Other related topics receiving attention are identifiability, model reduction, and model approximation. Some techniques for the estimation of linear dynamical i/o-systems are illustrated with the system identification toolbox in Matlab.

Knowledge and understanding: Students learn to recognize the various aspects that play a key role in building a mathematical model from measurement data: the choice of model class (and order), the choice of parameterization, the criterion of fit, the model estimation method, the quality of the measurement data, and the validity of the estimated model.

Applying knowledge and understanding: Students are able to 1) estimate models from measurement data, particularly linear regression models and auto-regressive models, 2) to assess the quality of a (linear regression) model, and 3) assess whether a model is identifiable.

Making judgements: Students are able to predict and judge the quality of models that can be obtained from a record of measurement data.

Communication: Students learn to motivate the choice of a model class, the model order and an estimation method to identify a model from measurement data, to interpret the identification outcomes and to explain all this to specialists and non-specialists.

Learning skills: Students are able to read and interpret scientific literature on model estimation and system identification, and to use Matlab and work out ideas computationally.

Study material: Syllabus, provided electronically on the digital learning environment.

Recommended literature:
- L. Ljung, System Identification: Theory for the User (2nd ed.), Prentice-Hall, 1999.
- T. Soderstrom and P. Stoica, System Identification, Prentice-Hall, 1989.

Exam: Written exam.

ECTS: 6

Period 2.4

Data Fusion (KEN4223)

Coordinator: Prof. Anna Wilbik

Examiners: Prof. Anna Wilbik & dr. Marcin Pietrasik

Tutor: Afsana Khan

Desired prior knowledge: Statistics and basic machine learning

Prerequisites: None.

Description: ICT development, e.g., remote sensing, IoT, lead to an enormous growth of available data for analysis. To integrate this heterogeneous or multimodal data, data fusion approaches are used. Data fusion can be understood as a framework for the joint analysis of data from multiple sources (modalities) that allows achieving information/knowledge not recoverable by the individual ones.

During this course, several approaches to data fusion will be discussed, such as:
1. Low level data fusion, where data fusion methods are directly applied to raw data sets for exploratory or predictive purposes. A main advantage is the possibility to interpret the results

directly in terms of the original variables. An example of a low level data fusion is measuring the same signal or phenomena with different sensors, in order to discover the original one. Traditionally, PCA based methods are used for this type of data fusion.
2. Mid level data fusion, where data fusion operates on features extracted from each data set. The obtained features are then fused in a "new" data set, which is modeled to produce the desired outcome. A main advantage is that the variance can be removed in the features extraction step, and thus the final models may show better performance. An example of a mid level data fusion is extracting numerical features from an image, and building a decision model based on those features.
3. High level data fusion, also known as decision fusion, where decisions (models outcome) from processing of each data set are fused. It is used when the main objective is to improve the performance of the final model and reach an automatic decision. Several methods can be used for high-level DF, such as weighted decision methods, Bayesian inference, Dempstere Shafer's theory of evidence, and fuzzy set theory. There is a link between high-level data fusion and ensemble methods.
4. Federated learning. Federated learning enables multiple parties jointly train a machine learning model without exchanging the local data. In case of federated learning, we can talk about model fusion.

Moreover, we will discuss the outcome economy model, to show the possibilities where data fusion could be beneficial in a business setting.

Knowledge and understanding: The student can explain fusion on the different levels: low level, mid level and high level as well as federated learning. They can identify which approach is appropriate for a problem in hand.

Applying knowledge and understanding: Students are able to describe the advantages and disadvantages of different methods. Students have obtained the knowledge to develop, program, analyse, and apply fusion methods to a wide variety of problems in the context of data-driven decision making.

Making judgements: Students will be able to judge the quality of models, results and approaches (e.g., scientific publications).

Communication: Students will be able to present the results the fusion models to specialists or non-specialists.

Learning skills: Students will be able to familiarize themselves with fusion techniques beyond the scope of the course in order to solve a problem.

Study material: Course notes and research papers made available.

Assessment: Written exam (70%) + group assignment (30%)

Recommended literature: research articles on those topics

Additional literature: research articles on those topics

ECTS: 6

Computational Statistics (KEN4258)

Coordinator & examiner: Dr. Christof Seiler

Desired prior knowledge: Probability and Statistics

Prerequisites: None.

Description: In this course, we will start with where data comes from—experiments or observational studies. We will then devote a substantial amount of time studying linear models and their extensions. We will articulate their assumptions and limitations. We will see that sometimes it's not possible to estimate certain things from the data no matter what tools somebody tries to sell us. We will learn about the bootstrap, randomization tests, and Markov chain Monte Carlo—computational methods to quantify uncertainty for any estimation algorithm. Towards the end of the course, we will get a taste of how to test hypotheses for datasets with many variables and few observations—something that we often encounter in modern scientific and business contexts.

Knowledge and understanding: Knowing a wide range of modern statistical models and computational tools to draw inferences will provide the foundations for analyzing complex data in academia and industry.

Applying knowledge and understanding: Students can build statistical models and choose computational tools to perform inference.

Making judgements: In this course, we will discuss one of the most important aspects of analyzing data: being skeptical of results and avoiding wishful thinking.

Communication: Students will present their results using literate programming and reproducible workflows.

Learning skills: Students can understand, apply, and extend papers from statistics journals and machine learning conferences.

Study material: Lecture slides, selected chapters from textbooks, and research papers

Assessment: 20% homework assignments

Exam: 80% written final exam

Recommended literature: David A. Freedman, Statistical Models (2nd Edition, 2012)

ECTS: 6

Symbolic Computation and Control (KEN4252)

Examiner: dr. ir. Philippe Dreesen

Desired prior knowledge: Linear Algebra, Calculus, Mathematical Modelling.

Course description: This course consists of two interrelated parts. In the first part, we focus on basic techniques for the digital control of linear dynamical systems using feedback. We start by addressing system stability and we discuss the technique of pole placement by state feedback to solve the regulation problem. Then we introduce state observers to solve the regulation problem by output feedback. Next, we extend our scope to tracking problems. This involves the design of additional dynamics to characterize the relevant class of reference signals, which are then integrated with the earlier set-up for output feedback. Finally, we discuss the classical topic of optimal control, which can be employed to avoid using prototype systems for pole placement, and which allows the user to design a feedback law by trading off the cost involved in generating large inputs against the achieved tracking accuracy. In the second part, we address computational issues, related to the field of systems and control. Classically, computers have been designed primarily to perform approximate numerical arithmetic. Modern software packages for mathematical computation, such as Maple and Mathematica, allow one to perform exact and symbolic computation too. We shall explore this new area. It is demonstrated how speed, efficiency and memory usage considerations often lead to surprising and fundamentally different algorithmic solutions in a symbolic or exact context. Applications and examples involve stability of linear systems, model approximation, and linear matrix equations with free parameters. Practical classes serve to demonstrate the techniques and to make the student familiar with exact and symbolic computation.

Knowledge and understanding: Students familiarize themselves with state and output feedback to achieve control of dynamical systems. Concretely, they learn to (mathematically) build a basic stabilizing feedback controller for a linear input-output dynamical system, using a combination of different design techniques. Students learn methods for exact numerical and symbolic computation, as used in algebraic computation with unspecified parameters. They also learn in which ways these are different from the more commonly used approximate numerical (floating-point) methods: in terms of accuracy, speed (complexity), and memory usage.

Applying knowledge and understanding: Students will be able to construct and implement, for a given linear dynamical input-output system: (a) stabilizing state feedback, (b) full state observer, and (c) additional dynamics to perform tracking of a specified output trajectory. They will also be able to assess the quality of a controller, regarding an optimal control LQ criterion, and in view of the desired settling time and the trajectory approximation. Students will be able to determine the stability of a given linear dynamical system in an exact and/or symbolic algebraic way. They will also be able to efficiently solve linear systems of (matrix) equations involving symbolic parameters, avoiding pitfalls, which arise from techniques from approximate numerical computation.

Making judgements: Students will be able to judge the quality of a feedback design for stabilization (regulation) or tracking. Students will be able to indicate which exact and symbolic computation methods will and will not be useful for a given parameterized problem, regarding speed and memory usage.

Communication: Students will be able to motivate the design of a feedback controller, the construction of a trajectory approximation, the design of a full state observer, and the implementation choices of the weights in LQ-design. They will be able to explain the concept of feedback in the area of control. Students can adequately discuss speed and efficiency properties of an algorithm (approximate numerical, exact numerical, symbolic) to specialists and non-specialists.

Learning skills: Students will be able to read and interpret basic scientific literature on control theory and on numerical and symbolic computation. They can use Matlab and the Control Toolbox and work out ideas computationally. Students can use some of the exact and symbolic functionality of Mathematica and work out ideas computationally.

Study material: Syllabus, provided on the study portal. Handouts.

Recommended Literature: Richard J. Vaccaro, Digital Control - A State-Space Approach, McGraw-Hill International Editions, 1995. ISBN 0-07-066781-0.

Exam: Written exam by computer in two parts, each having a weight of 50% on the final grade: one midterm take-home exam with Matlab on part 1 (control), one final classroom exam with Mathematica on part 2 (symbolic computation). The resit exam is on both parts of the course in a classroom setting.

ECTS: 6

Algorithms for Big Data (KEN4254)

Examiner: Dr. Matus Mihalák

Desired prior knowledge: Discrete mathematics, algorithm design and analysis, elementary discrete probability

Prerequisites: None.

Description: The emergence of very large datasets poses new challenges for the algorithm designer. For example, the data may not fit into the main memory anymore, and caching from a hard-drive becomes a new bottleneck that needs to be addressed. Similarly, algorithms with larger than linear running time take simply too long on very large datasets. Moreover, simple sensory devices can observe large amount of data over time, but cannot store all the observed information due to insufficient storage, and an immediate decision of what to store and compute needs to be made. Classical algorithmic techniques do not address these challenges, and a new algorithmic toolkit needs to be developed. In this course, we will look at a number of algorithmic responses to these problems, such as: algorithms with (sub-)linear running times, algorithms where the data arrive as a stream, computational models where memory is organized hierarchically (with larger storage units, such as hard-drives, being slower to access than smaller, faster storage such as CPU cache memory). New programming paradigms and models such as MapReduce/Hadoop will be discussed. We will also look at a number of topics from classical algorithm design that have undiminished relevance in the era of big data such as approximation algorithms and multivariate algorithmic analysis.

Knowledge and understanding: Students will know, exemplified on selected topics, what can be provably achieved when designing and analysing algorithms for very large datasets, and will know some of the most successful state-of-the-art algorithmic techniques for dealing with algorithmic challenges posed by large data sets.

Applying knowledge and understanding: Students will be able to adjust and apply the gained knowledge about algorithmic techniques to various algorithmic challenges of handling large datasets.

Making judgements: Students will be able to categorize large-scale problems according to their computational feasibility, and select the appropriate algorithmic response.

Communication: Students will be able to reason about computational problems and algorithms addressing the problems in a clear, exact, and unambiguous way.

Learning skills: Additionally to the guiding material provided by the lecture, the students will autonomously search, read, and study the details from various sources.

Study material: Will be provided throughout the lecture.

Recommended literature: None.

Assessment: Written exam (80%) at the end of the course and graded exercises (20%) throughout the course.

ECTS: 6

3.6 Master Data Science for Decision Making

Data Science for Decision Making is the science of making informed and close-to-optimum decisions. It has widespread applications in health-care, society, business and engineering. In today's world, many companies and organizations collect all sorts of data in large amounts. They aim to extract useful information from it, to recognize patterns and anomalies, and based on that improve their decisions. Data Science for Decision Making provides the mathematical tools to analyze and model the underlying real-world processes and decision questions, and to process and analyze the (big) data that come with the processes. It also provides and uses the computational software that is the key to data science.

The two-year master's programme in Data Science for Decision Making teaches the use of applied mathematics to analyze and optimize processes, problems and operations. Examples of applications are: discovering patterns in data such as images and time series, predicting future values such as demand for products or traffic on the roads, scheduling customer service agents, optimizing supply chains, controlling dynamical systems, modelling biological processes, finding optimal strategies in negotiation, and extracting meaningful components from brain signals.

The master's programme Data Science for Decision Making covers a wide range of research topics, focusing on the following ones in its core:
1. Data mining to extract useful patterns and knowledge from large data repositories;
2. Mathematical modelling and parameter estimation from data, system identification, model approximation and reduction of model complexity;
3. Underlying mathematics and algorithm design and analysis to efficiently deal with the challenges that the ever-growing amount of data pose;
4. Statistical analysis, in the computational sense, of data.

The members of the teaching staff are actively involved in one or more of the research topics. As a result, the educational contents of the courses relate directly to the research performed.

3.7 Curriculum of the first year of the Master Programme Data Science for Decision Making

Year 1		ECTS
Period 1	1 Data Mining (KEN4113) *1 elective course from the following courses:* Mathematical Optimization (KEN4211) Stochastic Decision-Making (KEN4221) Signal and Image Processing (KEN4222) Research Project 1 (**)	6 6 6 6
Period 2	Model Identification and Data Fitting (KEN4242) *1 elective course from the following courses:* Advanced Concepts in Machine Learning (KEN4154) Advanced Natural Language Processing (KEN4259) Network Science (KEN4275) Research Project 1 (**)	6 6 6 6
Period 3	Research Project 1 (KEN4230)	6
Period 4	Computational Statistics (KEN4258) *1 elective course from the following courses:* Data Fusion (KEN4223) Explainable AI (***) (KEN4246) Dynamic Game Theory (KEN4251) Planning and Scheduling (KEN4253) Building and Mining Knowledge Graphs (KEN4256) Research Project 2 (**)	6 6 6 6 6 6
Period 5	Algorithms for Big Data (KEN4254) *1 elective course from the following courses* Information Retrieval and Text Mining (KEN4153) Introduction to Quantum Computing for AI and DS (KEN4155) (****) Symbolic Computation and Control (KEN4252) Computer Vision (KEN4255) Research Project 2 (**)	6 6 6 6 6
Period 6	Research Project 1-2 (KEN4231)	6

(*) ECTS credits obtained in year 1 of the programme cannot be used for exemptions in year 2 of the programme. 90 unique ECTS (course) credits (+ 30 ECTS for the Master's thesis) need to be obtained throughout the Master's programme.

(**) The Research Project 1 will start in period 1.1 and 1.2 with weekly meetings. The credits for the project will become available at the end of period 1.3. The Research Project 2 will start in period 1.4 and 1.5 with weekly meetings. The credits for the project will become available at the end of period 1.6.

(***) The course has a capacity of 60 students.

(****) This course is a prerequisite for the elective courses Quantum Algorithms, Quantum AI, and Quantum Information and Security. These four courses, together with a dedicated research project on quantum computing, forms the specialization in Quantum Computing for AI and Data Science.

3.8 Curriculum of the second year of the Master Programme Data Science for Decision Making

Period 1, 2 and 3 of year two of the master's program consist of electives to be chosen by the student. This optional program can be assembled at your own choice from the options provided, but within academic significance, level and relevance to your master's track. The choice of electives is subject to approval by the Board of Examiners. The electives consist of the following options to choose from: master courses to be followed at the Department of Advanced Computing Sciences, at other UM Master programmes, at another research university, a research project, an internship, a semester abroad at a foreign university. Note that you must have obtained at least 40 ECTS of course year 1 in order to enter the second year of the programme.

Electives at Maastricht University outside the Department of Advanced Computing Sciences
It is possible to take electives at other relevant master's programmes at Maastricht University for at most 13 ECTS in the second year of the programme. The following courses below will be automatically approved by the Board of Examiners of the master's programmes AI and DSDM. You should apply through the Special Course Approval procedure via the My UM Portal. Note that they may have limited capacity.

School of Business and Economics	ECTS
Collective Decision Making (EBC4005)	6.5
Supply Chain Operations Management (EBC4016)	6.5
Negotiation & Allocation (EBC4193)	6.5
Ethics, Privacy and Security in a Digital Society (EBC4026)	6.5
Big Data Econometrics (EBC4218)	6.5
Digital Business and Economics (EBC4083)	6.5

Faculty of Psychology and Neuroscience
Besides complying that you have passed 40 ECTS, for taking these electives at FPN you should have passed "Advanced Concepts in Machine Learning" and "Autonomous Robotic Systems" at the Department of Advanced Computing Sciences.

Faculty of Psychology and Neuroscience	ECTS
Auditory and Higher Order Language Processing (PSY4051)	4
Perception and Attention (PSY4052)	4
Sensorimotor Processing (PSY4054)	4

Exam: Depends on content of the elective program.

ECTS: 30

Internships:
Another option for the elective semester in the Master Programme is to conduct a Business or Research internship. The students can choose the company or research organisation themselves. Together with a supervisor from the Department of Advanced Computing Sciences and a representative of the host organisation, the student fills out an internship proposal (which can be found on Canvas) and this requires approval of the Board of Examiners prior to its start. For this reason, it is important to start this process early. The university uses a standard internship agreement that students must use.

Year 2		ECTS
Semester 1 *	• Internship (research or business) • Study abroad • Elective courses at other UM Master's programmes (at most 13 ECTS)	30
	AND/OR:	
Fall:	*At most 2 electives from the following courses:*	
Period 1	Intelligent Search and Games (KEN4123) Mathematical Optimization (KEN4211) Stochastic Decision-Making (KEN4221) Signal Image Processing (KEN4222) Quantum Algorithms (KEN4235)	6 6 6 6 6
Period 2	*At most 2 electives from the following courses:*	
	Advanced Concepts in Machine Learning (KEN4154) Quantum AI (KEN4236) Quantum Information and Security (KEN4237) Advanced Natural Language Processing (KEN4259) Network Science (KEN4275)	6 6 6 6 6
Period 1-3	Research Project 2-1	6
Spring:		
Period 4	*At most 2 electives from the following courses:*	
	Agents and Multi-Agent Systems (KEN4111) Data Fusion (KEN4223) Explainable AI (*) (KEN4246) Dynamic Game Theory (KEN4251) Planning and Scheduling (KEN4253) Building and Mining Knowledge Graphs (KEN4256)	6 6 6 6 6 6
Period 5	*At most 2 electives from the following courses:*	
	Autonomous Robotic Systems (KEN4114) Information Retrieval and Text Mining (KEN4153) Introduction Quantum Computing for AI and Data Science (**) (KEN4155) Reinforcement Learning (KEN4157) Symbolic Computation and Control (KEN4252) Computer Vision (KEN4255)	6 6 6 6 6 6
Period 4-6	Research Project 2-2	6
Semester 2	Master's thesis DSDM (KEN4260)	30

(*) Note: during the elective semester (first semester of year 2) of the master's programme it is possible to take electives from our other master's programme or relevant master's programmes at Maastricht University (maximum of 13 ECTS outside the Department of Advanced Computing Sciences) or to participate in a research project, a business internship or a study abroad semester at one of our partner universities. Please contact exchange officer and/or the Student Counsellor for more information.

(**) The course has a capacity of 60 students

(***) This course is a prerequisite for the elective courses Quantum Algorithms, Quantum AI and Quantum Information and Security. These four courses, together with a dedicated research project on quantum computing, form the specialization in Quantum Computing for AI and Data Sciences.

3.9 Core courses Master Programme Data Science for Decision Making

Period 1.1

Data Mining (KEN4113)

Examiner: Dr. E.N. Smirnov

Desired prior knowledge: Statistics and Basic Machine Learning

Prerequisites: None.

Description: Data mining is a major frontier field of machine learning. It allows extracting useful and interesting patterns and knowledge from large data repositories such as databases and the Web. Data mining integrates techniques from the fields of databases, machine learning, statistics, and artificial intelligence. This course will present basic and state-of-the-art techniques of data mining. The lectures and labs will emphasize the practical use of the presented techniques and the problems of developing real data-mining applications. A step-by-step introduction to data-mining and python-based environments will enable the students to achieve specific skills, autonomy, and hands-on experience. A number of real data sets will be analyzed and discussed.

Formal models that will be investigated: Parametric and non-parametric models for supervised and unsupervised learning.

Knowledge and understanding: Students will acquire knowledge on data preparation, data preprocessing, feature selection/generation, data mining, and model validation.

Applying knowledge and understanding: When confronted with real-life problems, students will be able to identify data-analysis tasks. Then, they will be able to apply data-mining techniques for supervised and unsupervised data-analysis. If necessary, students will be able to design data-mining algorithms specific for the tasks they have.

Making judgements: Students will be able to assess the quality of data-mining models, processes, results, and tools.

Communication: Students will be able to present the results of different stages of data-mining processes to specialists or non-specialists.

Learning skills: Students will be able to recognize their own lack of knowledge and understanding and take appropriate action such as consulting additional material or other sources of help.

Study material: Course notes, slides, and other information made available.

Assessment: Written exam + practical assignments.

Recommended Literature: Han, J., Pei, J., and Tong, H. (2022). Data Mining Concepts and Techniques, 4th Edition, ISBN-10: 9780128117606, ISBN-13: 978-0128117606

Additional literature: Pang-Ning, T., Steinbach, M., Karpatne, A., and Kumar, V. (2018). Introduction to Data Mining, 2nd Edition, Pearson, ISBN-10: 0133128903, ISBN-13: 978-0133128901

ECTS: 6

Model Identification and Data Fitting (KEN4242)

Examiners: Prof. dr. ir. Ralf Peeters & dr. ir. Philippe Dreesen

Desired prior knowledge: Basic knowledge of Matlab and some familiarity with linear systems theory and transforms (such as Fourier and Laplace) is helpful. This course offers a useful prior knowledge for the course Symbolic Computation and Control. Linear Algebra, Mathematical Modelling, Probability and Statistics.

Prerequisites: None.

Course description: This course is devoted to the various practical and theoretical aspects which involve the estimation (the identification) of a mathematical model within a given model class, starting from a record of observed measurement data (input-output data). First, we address distance measures, norms, and criterion functions. Then we discuss the prediction error identification of linear regression models, with special emphasis on the various interpretations of such models (deterministic, stochastic with Gaussian white noise and maximum likelihood estimation, stochastic in a Bayesian estimation context) and on numerical implementation aspects (recursion, numerical complexity, numerical conditioning and square root filtering). Next, we study identification within the important class of auto-regressive dynamical models, to which the Levinson algorithm applies. Other related topics receiving attention are identifiability, model reduction, and model approximation. Some techniques for the estimation of linear dynamical i/o-systems are illustrated with the system identification toolbox in Matlab.

Knowledge and understanding: Students learn to recognize the various aspects that play a key role in building a mathematical model from measurement data: the choice of model class (and order), the choice of parameterization, the criterion of fit, the model estimation method, the quality of the measurement data, and the validity of the estimated model.

Applying knowledge and understanding: Students are able to 1) estimate models from measurement data, particularly linear regression models and auto-regressive models, 2) to assess the quality of a (linear regression) model, and 3) assess whether a model is identifiable.

Making judgements: Students are able to predict and judge the quality of models that can be obtained from a record of measurement data.

Communication: Students learn to motivate the choice of a model class, the model order and an estimation method to identify a model from measurement data, to interpret the identification outcomes and to explain all this to specialists and non-specialists.

Learning skills: Students are able to read and interpret scientific literature on model estimation and system identification, and to use Matlab and work out ideas computationally.

Study material: Syllabus, provided electronically on the digital learning environment.

Recommended literature:
- L. Ljung, System Identification: Theory for the User (2nd ed.), Prentice-Hall, 1999.
- T. Soderstrom and P. Stoica, System Identification, Prentice-Hall, 1989.

Exam: Written exam.

ECTS: 6

Project 1-1 (KEN4230)

Coordinator: Dr. Linda Rieswijk

Examiner(s): To be announced

Tutors: Dr. Gijs Schoenmakers, Dr. Menica Dibenedetto & Dr. Linda Rieswijk

Desired prior knowledge: None.

Prerequisites: None.

Description: The research project takes place during the three periods of the semester. Project topics are presented at the start of the semester and assigned to students based on their preferences and availability. The emphasis in the first phase is on initial study of the context set out for the project and the development of a project plan. In the second period, the goal is to start modelling, prototyping and developing. In period 3, the implementation, model and/or experiments set out in the project plan has to be finished and reported on. The project results in a project presentation, a project report and possibly a public website and/or product.

Knowledge and understanding: Students get to know and possibly contribute to state of the art methods within the fields of Artificial Intelligence and/or Data Science for Decision Making to answer an open question.

Applying knowledge and understanding: Student write their own research plan in coordination with a staff member (plus possibly outsiders) who act as clients with an open question. Students with different backgrounds and from both masters work together in teams to build and evaluate an answer to an open question. Students find, judge the suitability, apply, and evaluate state of the art techniques to answer questions and construct applications in the field of Artificial Intelligence and Data Science. Students apply the accumulated knowledge from other educational activities in application specific areas

Making judgements: Students judge feasibility of tasks, attainability of goals, and the amount of work involved. Students think about the possible consequences of their work. Students evaluate state of the art and the applicability and scope of research results.

Communication: Students will learn to:
(1) orally communicate and cooperate with peers
(2) orally report on progress and intermediate results to superiors
(3) orally negotiate and communicate with clients
(4) communicate their ideas in written form, both for an academic and a general audience
(5) give effective presentations

Learning skills: Students increase their own level of knowledge in a specialized sub-discipline of the field of Artificial Intelligence and/or Data Science. Students perform research into recent state of the art techniques. Students learn that the field of Artificial Intelligence and Data Science are constantly evolving beyond what is taught in class

Study material: Slides provided at the end of joint information sessions. Literature provided by the project supervisors.

Assessment: Phase 1: project plan (15%); Phase 2: social media post+ presentation (15%); Phase 3: Project report + presentation (70%)

Skill classes: A number of skill classes will be offered

ECTS: 6

Computational Statistics (KEN4258)

Coordinator & examiner: Dr. Christof Seiler

Desired prior knowledge: Probability and Statistics

Prerequisites: None.

Description: In this course, we will start with where data comes from—experiments or observational studies. We will then devote a substantial amount of time studying linear models and their extensions. We will articulate their assumptions and limitations. We will see that sometimes it's not possible to estimate certain things from the data no matter what tools somebody tries to sell us. We will learn about the bootstrap, randomization tests, and Markov chain Monte Carlo—computational methods to quantify uncertainty for any estimation algorithm. Towards the end of the course, we will get a taste of how to test hypotheses for datasets with many variables and few observations—something that we often encounter in modern scientific and business contexts.

Knowledge and understanding: Knowing a wide range of modern statistical models and computational tools to draw inferences will provide the foundations for analyzing complex data in academia and industry.

Applying knowledge and understanding: Students can build statistical models and choose computational tools to perform inference.

Making judgements: In this course, we will discuss one of the most important aspects of analyzing data: being skeptical of results and avoiding wishful thinking.

Communication: Students will present their results using literate programming and reproducible workflows.

Learning skills: Students can understand, apply, and extend papers from statistics journals and machine learning conferences.

Study material: Lecture slides, selected chapters from textbooks, and research papers

Assessment: 20% homework assignments

Exam: 80% written final exam

Recommended literature: David A. Freedman, Statistical Models (2nd Edition, 2012)

ECTS: 6

Algorithms for Big Data (KEN4254)

Examiner: Dr. Matus Mihalák

Desired prior knowledge: Discrete mathematics, algorithm design and analysis, elementary discrete probability

Prerequisites: None.

Description: The emergence of very large datasets poses new challenges for the algorithm designer. For example, the data may not fit into the main memory anymore, and caching from a hard-drive becomes a new bottleneck that needs to be addressed. Similarly, algorithms with larger than linear running time take simply too long on very large datasets. Moreover, simple sensory devices can observe large amount of data over time, but cannot store all the observed information due to insufficient storage, and an immediate decision of what to store and compute needs to be made. Classical algorithmic techniques do not address these challenges, and a new algorithmic toolkit needs to be developed. In this course, we will look at a number of algorithmic responses to these problems, such as: algorithms with (sub-)linear running times, algorithms where the data arrive as a stream, computational models where memory is organized hierarchically (with larger storage units, such as hard-drives, being slower to access than smaller, faster storage such as CPU cache memory). New programming paradigms and models such as MapReduce/Hadoop will be discussed. We will also look at a number of topics from classical algorithm design that have undiminished relevance in the era of big data such as approximation algorithms and multivariate algorithmic analysis.

Knowledge and understanding: Students will know, exemplified on selected topics, what can be provably achieved when designing and analysing algorithms for very large datasets, and will know some of the most successful state-of-the-art algorithmic techniques for dealing with algorithmic challenges posed by large data sets.

Applying knowledge and understanding: Students will be able to adjust and apply the gained knowledge about algorithmic techniques to various algorithmic challenges of handling large datasets.

Making judgements: Students will be able to categorize large-scale problems according to their computational feasibility, and select the appropriate algorithmic response.

Communication: Students will be able to reason about computational problems and algorithms addressing the problems in a clear, exact, and unambiguous way.

Learning skills: Additionally to the guiding material provided by the lecture, the students will autonomously search, read, and study the details from various sources.

Study material: Will be provided throughout the lecture.

Recommended literature: None.

Exam: Written exam (75%) at the end of the course and graded exercises (25%) throughout the course.

ECTS: 6

Project 1-2 (KEN4231)

Coordinator: Dr. Linda Rieswijk

Examiner(s): T.B.A.

Tutors: Dr. Gijs Schoenmakers, Dr. Menica Dibenedetto & Dr. Linda Rieswijk

Desired prior knowledge: None.

Prerequisites: None.

Description: The research project takes place during the three periods of the semester. Project topics are presented at the start of the semester and assigned to students based on their preferences and availability. The emphasis in the first phase is on initial study of the context set out for the project and the development of a project plan. In the second period, the goal is to start modelling, prototyping and developing. In period 3, the implementation, model and/or experiments set out in the project plan has to be finished and reported on. The project results in a project presentation, a project report and possibly a public website and/or product.

Knowledge and understanding: Students get to know and possibly contribute to state of the art methods within the fields of Artificial Intelligence and/or Data Science for Decision Making to answer an open question.

Applying knowledge and understanding: Student write their own research plan in coordination with a staff member (plus possibly outsiders) who act as clients with an open question. Students with different backgrounds and from both masters work together in teams to build and evaluate an answer to an open question. Students find, judge the suitability, apply, and evaluate state of the art techniques to answer questions and construct applications in the field of Artificial Intelligence and Data Science. Students apply the accumulated knowledge from other educational activities in application specific areas

Making judgements: Students judge feasibility of tasks, attainability of goals, and the amount of work involved. Students think about the possible consequences of their work. Students evaluate state of the art and the applicability and scope of research results.

Communication: Students will learn to:
1. orally communicate and cooperate with peers
2. orally report on progress and intermediate results to superiors
3. orally negotiate and communicate with clients
4. communicate their ideas in written form, both for an academic and a general audience
5. give effective presentations

Learning skills: Students increase their own level of knowledge in a specialized sub-discipline of the field of Artificial Intelligence and/or Data Science. Students perform research into recent state of the art techniques. Students learn that the field of Artificial Intelligence and Data Science are constantly evolving beyond what is taught in class

Study material: Slides provided at the end of joint information sessions. Literature provided by the project supervisors.

Assessment: Phase 1: project plan (15%); Phase 2: social media post+ presentation (15%); Phase 3: Project report + presentation (70%)

Skill classes: A number of skill classes will be offered

ECTS: 6

3.10 Elective courses Master Programme Data Science for Decision Making

Period 1.1 & 2.1

Mathematical Optimization (KEN4211)

Coordinator & examiner: Dr. Pieter Collins

Desired prior knowledge: Calculus, Linear Algebra, Linear Programming

Prerequisites: None.

Description: Optimization is the subject of finding the best or optimal solution to a problem from a set of potential or feasible solutions. Optimization problems are fundamental in all forms of decision-making, since one wishes to make the best decision in any context, and in the analysis of data, where one wishes to find the best model describing experimental data. This course treats two different areas of optimization: nonlinear optimization and combinatorial optimization, building on knowledge of linear programming using the simplex algorithm. Together, nonlinear and combinatorial optimization cover a wide range of real life optimization problems. Nonlinear optimization deals with the situation that there is a continuum of available solutions. A best solution is then usually approximated with one of several available general purpose algorithms, such as Brent's method for one-dimensional problems, (quasi-)Newton and conjugate gradient methods for unconstrained problems, and Lagrangian methods, including active-set methods, sequential quadratic programming and interior-point methods for general constrained problems. Combinatorial optimization deals with situations that a best solution from a discrete set of available choices must be found. A variety of techniques, such as linear programming, branch and cut, Lagrange relaxation and approximation algorithms are employed to tackle this type of problem. Throughout the course, we aim to provide a coherent framework for the subject, with a focus on optimality conditions (notably the Karush-Kuhn-Tucker conditions), Lagrange multipliers and duality, relaxation and approximate problems, and on convergence rates and computational complexity. The methods will be illustrated by in-class computer demonstrations, exercises illustrating the main concepts and algorithms, modelling and computational work on case studies of practical interest, and a discussion of advanced stochastic and batch optimization methods for machine-learning.

Formal models that will be investigated: Unconstrained and constrained nonlinear programming problems; integer-linear programming problems.

Knowledge and understanding: By the end of this course, students will have a strong foundation in nonlinear and combinatorial optimization. You will be able to formulate real-life problems as optimization problems. You will understand optimality conditions, including the Karush-Kuhn-Tucker conditions and be able to test for optimality. You will know how to solve a variety of general optimization problems, including constrained nonlinear problems, and (mixed-)integer linear problems. You will understand notions of duality and Lagrange multipliers, and be able to apply techniques based on relaxation and approximation.

Applying knowledge and understanding: Students will know the advantages and disadvantages of different methods, and be able to choose an appropriate method for a given problem. You will be able to implement and test optimization algorithms on a computer. You will be able to apply your knowledge to the solution of practical problems and in developing new efficient algorithms.

Making judgements: Students will be able to select an appropriate solution method for a given optimization problem, and judge the quality of the solution obtained.

Communication: Students will be able to discuss the development and use of optimization algorithms.

Learning skills: Students will learn how to develop and implement mathematical methods for optimization, select and evaluate algorithms, and formulate mathematical model of real-world problems.

Study material: Lecture notes, handouts.

Exam: Written examination, closed book (100%).

Recommended literature: J. Nocedal and S.J. Wright, "Numerical Optimization", Springer, 2006; ISBN: 978-0-387-30303-1. C.H. Papadimitriou and K. Steiglitz, "Combinatorial Optimization: Algorithms and Complexity", Dover Publications, 1998; ISBN: 978-0-13152-462-0. W.J. Cook, W.H. Cunningham, W.R. Pulleyblank and A. Schrijver, "Combinatorial Optimization", Wiley-Interscience, 1998; ISBN: 978-0-47155-894-1.

ECTS: 6

Stochastic Decision-Making (KEN4221)

Coordinator: Dr. Gijs Schoenmakers

Examiners: Dr. Gijs Schoenmakers & Dr. Dennis Soemers

Desired prior knowledge: Probability & Statistics

Description: Any realistic model of a real-world phenomenon must take into account the possibility of randomness. That is, more often than not, the quantities we are interested in will not be predictable in advance but, rather, will exhibit an inherent variation that should be taken into account by the model. Mathematically, this is usually accomplished by allowing the model to be probabilistic in nature. In this course, the following topics will be discussed:

1. Basic concepts of probability theory: Probabilities, conditional probabilities, random variables, probability distribution functions, density functions, expectations and variances.
2. Finding probabilities, expectations and variances of random variables in complex probabilistic experiments.
3. Discrete and continuous time Markov chains and related stochastic processes like random walks, branching processes, Poisson processes, birth and death processes, queueing theory.
4. Markov decision problems.
5. Multi-armed bandit problems, bandit algorithms, contextual bandits, cumulative regret, and simple regret

Knowledge and understanding: In this course, the students acquire tools for modelling complex processes involving randomness, providing a basis for originality in developing and/or applying ideas in a research context.

Applying knowledge and understanding: When confronted with complex problems that involve probabilistic experiments, students have the tools to create and analyse appropriate models.

Making judgements: The students are able to analyse complex problems as stochastic processes and solve them. Furthermore, students can find optimal solutions in decision problems that are based on these stochastic processes.

Communication: The students will be able to communicate their conclusions and the underlying rationale to expert and non-expert audiences.

Learning skills: The students have obtained the skills to study related material in a largely autonomous manner.

Study material: Introduction to Probability Models by Sheldon M. Ross (9 th or 10th ed.) + Lecture notes that are provided via Student Portal.

Exam: Written exam.

Recommended literature: Probability: A Lively Introduction by Henk Tijms; Reinforcement Learning by Richard S. Sutton and Andrew G. Barto (2nd ed.) (chapter 2); Bandit Algorithms by Tor Lattimore and Csaba Szepesvári

ECTS: 6

Signal and Image Processing (KEN4222)

Examiner: Dr. Joel Karel & dr. Pietro Bonizzi

Desired Prior Knowledge: Linear algebra, Calculus, basic knowledge of Matlab. Some familiarity with linear systems theory and transforms (such as Fourier and Laplace) is helpful.

Prerequisites: None.

Description: This course offers the student a hands-on introduction into the area of digital signal and image processing. We start with the fundamental concepts and mathematical foundation. This includes a brief review of Fourier analysis, z-transforms and digital filters. Classical filtering from a linear systems perspective is discussed. Next wavelet transforms and principal component analysis are introduced. Wavelets are used to deal with morphological structures in signals. Principal component analysis is used to extract information from high-dimensional datasets. We then discuss Hilbert-Huang Transform to perform detailed time-frequency analysis of signals. Attention is given to a variety of objectives, such as detection, noise removal, compression, prediction, reconstruction and feature extraction. We discuss a few cases from biomedical engineering, for instance involving ECG and EEG signals. The techniques are explained for both 1D and 2D (images) signal processing. The subject matter is clarified through exercises and examples involving various applications. In the practical classes, students will apply the techniques discussed in the lectures using the software package Matlab.

Knowledge and understanding: Students are able to explain fundamental concepts of signal and image processing and their mathematical foundation. They are able to 1) describe various types of filters and their properties, 2) explain orthogonal wavelet filter banks and describe their properties, 3) explain a construction scheme and elicit a wavelet-based noise-filtering scheme, 4) explain principal component analysis and empirical signal processing techniques and how they complement the other techniques discussed.

Applying knowledge and understanding: Students are able to use the various techniques discussed during the lectures to solve real-world problems, such as being able to apply wavelet filtering and principal component analysis on various signals. They are also able to analyse a signal by using Matlab, and independently interpret the outcome of an analysis.

Making judgements: Students are able to assess what technique is suited for a signal processing problem at hand, and to independently and critically look at a signal or image, and understand if and what type of pre-processing is required.

Communication: Students are able to communicate signal and image processing techniques and strategies, and the results of their analyses to experts and non-experts.

Learning skills: Students are able to independently master signal and image processing techniques, from classical signal processing techniques to more empirical techniques, and they are able to stay up to date with the state of the art in the field.

Study material: Discrete Wavelet Transformations: An Elementary Approach with Applications, Patrick J. Van Fleet, Wiley, ISBN: 978-0-470-18311-3.
Additional material provided electronically on Student Portal.

Recommended literature: Principal Component Analysis, Ian T. Jolliffe, Springer, ISBN13: 978-0387954424.

Exam: Written exam/Computer exam.

ECTS: 6

Period 1.2 & 2.2

Advanced Concepts in Machine Learning (KEN4154)

Coordinator & examiner: Dr. Enrique Hortal

Desired prior knowledge: Machine Learning

Prerequisites: None.

Description: This course will introduce a number of advanced concepts in the field of machine learning such as Support Vector Machines, Gaussian Processes, Deep Neural Networks, Neuromorphic Learning, etc. All of these are approached from the view that the right data representation is imperative for machine learning solutions. Additionally, different knowledge representation formats used in machine learning are introduced. This course counts on the fact that the basics of machine learning were introduced in other courses so that it can focus on more recent developments and state-of-the-art machine learning research. Labs and assignments will give the students the opportunity to implement or work with some of these techniques and will require them to read and understand published scientific papers from recent Machine Learning conferences.

Knowledge and understanding: Students can explain, construct and adapt powerful machine learning techniques, most with a statistical background. Students recognise the need for non-standard techniques and representations that can be used for complex/structured data. They can explain the strengths and weaknesses of different machine learning approaches.

Applying knowledge and understanding: Students will be able to select, adapt and apply a number of advanced machine learning approaches. They will be able to select the correct representation for a machine learning problem and translate a machine learning problem into a suited representational format.

Making judgements: Students will be able to judge which machine learning approach and data representation is best suited. They will also be able to comprehend and judge machine learning research.

Communication: Students will be able to relate different machine learning techniques to each other and explain their working, benefits, and disadvantages to non-experts. They will also be able to discuss the need and use of structured representation with both experts and non-experts.

Learning skills: Students will be able to relate information from different sources, and read, process and evaluate recent research developments in the field of machine learning.

Study material: Slides that support the lectures and collected notes and chapters from freely available books and course notes.

Assessment: Closed-book written exam (80%) + Assignments (20%)

Recommended literature:
- Pattern Recognition and Machine Learning - C.M. Bishop
- Bayesian Reasoning and Machine Learning - D. Barber
- Gaussian Processes for Machine Learning - C.E. Rasmussen & C. Williams
- The Elements of Statistical Learning - T. Hastie et al.

ECTS: 6

Advanced Natural Language Processing (KEN4259)

Examiners: Prof dr. ir. J.C. Scholtes & Dr. Aki Härmä

Desired prior knowledge: None.

Prerequisites: None.

Description: How do I say, "Where is the next Italian restaurant" in Dutch? Can I actually use speech recognition instead of typing my question? Can I get a summary of today's lecture? Can your chatbot assist me in finding the right information, answer my question or solve my problem? How do I know for sure that they chatbot does not hallucinate? How can I integrate multi-modal information in my language task?

Computers able to answer these questions are a long-time dream of humankind. For many years, computers underperformed using linguistic skills compared to humans. However, the development of Large Language Models (LLM) allowed us to make huge progress and perform at the human level for tasks such as machine translation, Q&A, abstracting, speech recognition, summarization and having a conversation with a computer program.

This course will provide the skills and knowledge to develop state-of-the-art (SOTA) solutions for these natural language processing (NLP) tasks.

After a short introduction to traditional grammatical and statistical approaches to NLP, the course will focus on deep learning techniques to solve these problems. In the first part of the course, we will investigate methods to model basic sequence labeling tasks like Part-of-Speech techniques. The second part of the lecture will focus on deep-learning models to solve many NLP tasks like machine translation, summarization and question answering.

In this course, major challenges when building the systems will be address: representing words in neural networks, neural network architectures to model language, methods to train complex models and algorithms to find the most probable output. Most of the lectures will focus on transformer-based models, both encoder, decoder and encoder-decoder models as well as multi-modal approaches. In addition, we discuss important aspects of Large Language Models (LLM) such as quantitative measuring of quality, fine-tuning LLM's, limitations to prompt engineering, ethics, energy consumption and eXplainable AI (XAI).

The theory discussed in the course is supported by various (Python) tutorials where the students can experience the inner-workings of the algorithms themselves.

Linear Algebra, Statistics, Deep Learning and Natural Language Processing play an important role in this course.

This course is complementary to the course Information Retrieval and Text-Mining. Overlap is reduced to the necessary minimum. Both courses can be followed in any particular order. In the Information Retrieval and Text Mining course we focus more on creating an optimal search experience, in the Advanced Natural Language Processing course, we do a deep dive into the algorithms and models used for different language-related problems such as machine translation, abstracting, and dialogs with chatbots. Tutorials are shared between the two courses.

Knowledge and understanding: Student will be taught traditional approaches in NLP, as well as statistical models. Finally, state-of-the-art (SOTA) deep learning techniques for natural language processing (including multi-modal information), including understanding methods to evaluate the performance of such models. They will learn techniques to address the major challenges when building a natural language processing tool, including explainability and more efficient energy usage of such models.

Appling knowledge and understanding: The achievements in deep learning have significantly improved the quality of state-of-the-art methods for natural language processing. With the knowledge acquired in the course, students will be able to build SOTA solutions. Students will also understand why deep learning models are outperforming traditional grammatical and statistical models and what the limitations and risks are of deep learning models in terms of applying explainable AI and more energy-efficient models.

Making Judgements: Students will be able to analyze the specific challenges of a task in NLP. Based on the gather knowledge on different ways to model tasks they are able to select and implement a fitting model to solve the task.

Communication: Through reporting on tutorials, students will be enabled to communicate their findings and explain the rationale behind their choices in deep learning techniques for natural language processing.

Learning Skills: After successful completion of the course, students will be able to develop natural language processing tools and perform research on new ideas in the field.

Study material: Mostly based on the lecture notes and the provided material including recent papers published in this field. We will also provide references to a number of good books that are on-line available for more background information.

Recommended literature: Papers published in top international conferences and journals in machine learning field.

Assessment: Participation in the tutorials (30%), final exam (70%). The exam is open book.

ECTS: 6

Network Science (KEN4275)

Coordinator: Adriana Iamnitchi

Desired prior knowledge: Introductory knowledge of programming for data analysis, particularly in Python; algorithms; algorithmic complexity. Introductory courses in algorithms, data structures, and data analytics.

Prerequisites: None.

Description: Many aspects of everyday life and science can be represented as networks: social networks represent relationship (links) between people (nodes); brain activity can be represented via synapses (links) between neurons (nodes); the street map is formed of roads (links) that connect at intersections (nodes); authors of scientific papers connect to each others in a citation network, with directed links from the paper cited to the paper citing it; communication networks connect routers via physical or logical links; etc. Network analysis plays a significant role in the "big data" analytics because of size, data velocity, or computational complexity.

This course focuses on the study of network structures and dynamic processes on networks using real data from various disciplines, including socio-technological platforms, biology, social science, and economics. Topics cover the analysis and modeling of complex networks, network dynamics, community detection, network resilience and contagion, as well as processing of network structures for machine learning tasks.

Formal models that will be investigated: random graphs, scale-free networks, preferential attachment model, Watts and Strogatz model, epidemics models.

Knowledge and understanding: Students will acquire a solid understanding of the key concepts and terminology in network science, will comprehend the theoretical underpinnings of various network models, and recognize relevant network characteristics across different contexts and applications.

Applying knowledge and understanding: Students will employ computational tools to model, analyze, and visualize networks from various real-world sources, and implement simulations to study network dynamics and evolution.

Making judgements: Students will critically assess different network models and their applicability to real-world problems. They will evaluate the implications of network structure on system dynamics. They will evaluate the benefits and limitations of various network embedding techniques for machine learning tasks.

Communication: Students will be able to present complex network concepts clearly to both specialist and non-specialist audiences and collaborate effectively in teams on network analysis projects.

Learning skills: In addition to the guiding material formally provided in the course, students will research independently from various sources.

Study material: Will be provided throughout the lecture.

Assessment: Assignments and group project.

Recommended literature: "Networks, Crowds, and Markets: Reasoning About a Highly Connected World" by David Easley and Jon Kleinberg.

Additional literature: Graph Theory and Complex Networks: An Introduction by Maarten van Steen

ECTS: 6

Period 1.4 & 2.4

Data Fusion (KEN4223)

Coordinator: Prof. Anna Wilbik

Examiners: Prof. Anna Wilbik & dr. Marcin Pietrasik

Tutor: Afsana Khan

Desired prior knowledge: Statistics and basic machine learning

Prerequisites: None.

Description: ICT development, e.g., remote sensing, IoT, lead to an enormous growth of available data for analysis. To integrate this heterogeneous or multimodal data, data fusion approaches are used. Data fusion can be understood as a framework for the joint analysis of data from multiple sources (modalities) that allows achieving information/knowledge not recoverable by the individual ones.

During this course, several approaches to data fusion will be discussed, such as:
1. Low level data fusion, where data fusion methods are directly applied to raw data sets for exploratory or predictive purposes. A main advantage is the possibility to interpret the results directly in terms of the original variables. An example of a low level data fusion is measuring the same signal or phenomena with different sensors, in order to discover the original one. Traditionally, PCA based methods are used for this type of data fusion.
2. Mid level data fusion, where data fusion operates on features extracted from each data set. The obtained features are then fused in a "new" data set, which is modeled to produce the desired outcome. A main advantage is that the variance can be removed in the features extraction step, and thus the final models may show better performance. An example of a mid level data fusion is extracting numerical features from an image, and building a decision model based on those features.
3. High level data fusion, also known as decision fusion, where decisions (models outcome) from processing of each data set are fused. It is used when the main objective is to improve the performance of the final model and reach an automatic decision. Several methods can be used for high-level DF, such as weighted decision methods, Bayesian inference, Dempstere Shafer's theory of evidence, and fuzzy set theory. There is a link between high-level data fusion and ensemble methods.
4. Federated learning. Federated learning enables multiple parties jointly train a machine learning model without exchanging the local data. In case of federated learning, we can talk about model fusion.

Moreover, we will discuss the outcome economy model, to show the possibilities where data fusion could be beneficial in a business setting.

Knowledge and understanding: The student can explain fusion on the different levels: low level, mid level and high level as well as federated learning. They can identify which approach is appropriate for a problem in hand.

Applying knowledge and understanding: Students are able to describe the advantages and disadvantages of different methods. Students have obtained the knowledge to develop, program, analyse, and apply fusion methods to a wide variety of problems in the context of data-driven decision making.

Making judgements: Students will be able to judge the quality of models, results and approaches (e.g., scientific publications).

Communication: Students will be able to present the results the fusion models to specialists or non-specialists.

Learning skills: Students will be able to familiarize themselves with fusion techniques beyond the scope of the course in order to solve a problem.

Study material: Course notes and research papers made available.

Assessment: Written exam (70%) + group assignment (30%)

Recommended literature: research articles on those topics

Additional literature: research articles on those topics

ECTS: 6

Explainable AI (KEN4246)

Coordinators: Prof. Dr. Nava Tintarev & Dr. Tjitze Rienstra

Examiners: Prof. Dr. Nava Tintarev & Dr. Tjitze Rienstra

Tutor: Aashutosh Ganesh

Desired prior knowledge: Data Analysis and Data Mining or ACML

Prerequisites: None.

Description: A key component of an artificially intelligent system is the ability to explain to a human agent the decisions, recommendations, predictions, or actions made by it and the process through which they are made. Such explainable artificial intelligence (XAI) can be required in a wide range of applications. For example, a regulator of waterways may use a decision support system to decide which boats to check for legal infringements, a concerned citizen might use a system to find reliable information about a new disease, or an employer might use an artificial advice-giver to choose between potential candidates fairly. For explanations from intelligent systems to be useful, they need to be able to justify the advice they give in a human-understandable way. This creates a necessity for techniques for automatic generation of satisfactory explanations that are intelligible for users interacting with the system. This interpretation goes beyond a literal explanation. Further, understanding is rarely an end-goal. Pragmatically, it is more useful to operationalize the effectiveness of explanations in terms of a specific notion of usefulness or explanatory goals such as improved decision support or user trust. One aspect of intelligibility of an explainable system (often cited for domains such as health) is the ability for users to accurately identify, or correct, an error made by the system. In that case it may be preferable to generate explanations that induce appropriate levels of reliance (in contrast to over- or under-reliance), supporting the user in discarding advice when the system is incorrect, but also accepting correct advice.

The following subjects will be discussed:
1. Intrinsically interpretable models, e.g., decision trees, decision rules, linear regression.
2. Identification of violations of assumptions, such as distribution of features, feature interaction, non-linear relationships between features; and what to do about them.
3. Model agnostic explanations, e.g., LIME, scoped Rules (Anchors), SHAP (and Shapley values)
4. Ethics for explanations, e.g., fairness and bias in data, models, and outputs.
5. Symbolic approaches to AI
6. (Adaptive) User Interfaces for explainable AI
7. Evaluation of explanation understandability

Knowledge and understanding: Students can explain the difference between different explanation approaches (e.g., global versus local models) and can identify which are suitable to use based on underlying assumptions and relative advantages and limitations.

Applying knowledge and understanding: Students can critically choose and apply XAI methods. Students can formulate evaluation protocols to validate the understandability of explanations, demonstrating awareness of the ethical, normative, and social consequences of their applications.

Making judgements: Students will be able to critically evaluate the quality (rigor of methodology), and ethical consequences, of approaches (systems or scientific publications) based on the XAI techniques taught.

Communication: Students will be able to communicate their ideas effectively in written form. They will be able to actively contribute to group-wise communication, and in both oral and written form present their models and outputs to specialists.

Learning skills: Students will be able to familiarize themselves, and critically assess XAI techniques beyond the scope of the course in order to solve a problem.

Study material: Course notes, required reading of scientific articles.

Assessment: Group project and individual written assignment

Recommended literature:
- Molnar, Christoph. Interpretable Machine Learning (second edition). Lulu.com, 2022 (available free online)
- Rothman, Denis. Hands-On Explainable AI (XAI) with Python: Interpret, visualize, explain, and integrate reliable AI for fair, secure, and trustworthy AI apps, Packt, 2020.

ECTS: 6

Dynamic Game Theory (KEN4251)

Examiner: Prof. dr. Frank Thuijsman & dr. Monica Salvioli

Desired prior knowledge: Students are expected to be familiar with basic concepts from linear algebra, calculus, Markov chains and differential equations.

Prerequisites: None.

Description: The course will focus on non-cooperative games and on dynamic games in the following order: matrix and bimatrix games, repeated games, differential games, specific models of stochastic games, Stackelberg games, games in extensive form and evolutionary games. These are games in which the players are acting as strategic decision makers, who cannot make binding agreements to achieve their goals. Instead, threats may be applied to establish stable outcomes. Besides, relations with population dynamics and with "learning" will be examined. Several examples will be taken from biological settings.

Knowledge and understanding: Students are able to recognize and classify the main types of dynamic games, i.e. repeated games, stochastic games, Stackelberg games, differential games, and evolutionary games and formulate the main solution concepts value, optimal strategies, Nash- and Stackelberg equilibrium

Applying knowledge and understanding: Students are able calculate solutions of the different types of dynamic games.

Making judgements: Students are able to explain advantages and disadvantages of different solution concepts. They are able to judge correctness of solutions presented.

Communication: Students are able to explain and defend correctness of their solutions.

Learning skills: By the end of the course, students will be able to autonomously and critically reflect upon the pros and cons of different types of games for modelling competition and cooperation. This includes considerations on the computational aspects with respect to different solution concepts.

Study material: Handouts will be provided.

Recommended literature: none.

Exam: There will be a closed book written exam at the end of the course.

ECTS: 6

Planning and Scheduling (KEN4253)

Examiner: Dr. Steven Kelk

Desired prior knowledge: Data Structures & Algorithms. Discrete Mathematics. Graph Theory.

Prerequisites: None.

Description: In many real-world processes, particularly in industrial processes and logistics, decisions need to be taken about the time of the completion of (sub)tasks, and the decision about what production machines complete the tasks. There are often constraints on the order in which tasks, or 'jobs' can be performed, and there are usually capacity constraints of the machines. This leads to natural, industrially critical optimization problems. For example, a company might choose to buy many machines to process jobs, but then there is a risk that the machines will be underused, which is economically inefficient. On the other hand, too few machines, or an inappropriate ordering of tasks, may lead to machines spending a significant amount of time standing idle, waiting for the output of other machines, which are overcrowded with tasks. In this course, we look at various mathematical models and techniques for optimizing planning and scheduling problems, subject to different optimality criteria. We will discuss, among others, single-machine models, parallel-machine models, job-shop models, and algorithms for planning and scheduling (exact, approximate, heuristic) and we also touch upon the computational complexity (distinguishing between 'easy' and 'difficult' problems) of the underlying problems. Last but not least, we will also introduce integer linear programming as a uniform and generic tool to model and solve planning and scheduling problems.

Knowledge and understanding: Students will possess the mathematical and algorithmic tools to model and solve planning/scheduling problems. Students will be able to recognize real-world problems in the unified theory and established language of planning and scheduling.

Applying knowledge and understanding: Students will be able to apply the new techniques to various problems arising in real-world applications. Students will be able to deploy the standard algorithmic techniques, and be able to design new algorithmic solutions, and to argue about their performance properties.

Making judgements: Students will understand under which circumstances different planning/scheduling problems are computationally tractable, and will judge algorithmic technique can be used to exactly or approximately solve these problems.

Communication: Students will be able to analytically argue about correctness of the used algorithmic approaches. Students will be able to explain modelling approaches to planning and scheduling problems in the language of the theory of planning and scheduling.

Learning skills: Students will enhance their study skills such as time management, effective reading, critical thinking and reading, exact and unambiguous writing and formulating of ideas and statements, and reflection on marked work. Along the way, students will improve general learning skills such as self-motivation, careful listening and giving instructions, and openness to new knowledge. Students will also be exposed to autonomous self-study.

Study material: Appropriate study material will be provided throughout the course.

Recommended literature: None

Assessment: Written exam (75%) at the end of the course, and graded exercises (25%) throughout the course.

ECTS: 6

Building and Mining Knowledge Graphs (KEN4256)

Examiner: Prof. dr. Michel Dumontier

Desired prior knowledge: Introduction to Computer Science

Prerequisites: None.

Description: Knowledge graphs are, seen through the semantic-web lens, large-scale, machine-processable representations of entities, their attributes, and their relationships. Knowledge graphs enable both people and machines to explore, understand, and reuse information in a wide variety of applications such as answering questions, finding relevant content, understanding social structures, and making scientific discoveries. However, the sheer size and complexity of these graphs present a formidable challenge particularly when mining across different topic areas.

In this course, we will examine approaches to construct and use knowledge graphs across a diverse set of applications using cutting-edge technologies such as machine learning and deep learning, graph databases, ontologies and automated reasoning, and other relevant techniques in the area of data mining and knowledge representation.

Knowledge and understanding: Students will be able to:
- Define and describe the nature and attributes of a Knowledge Graph
- Identify and describe the components of a Knowledge Graph
- Distinguish between different representations for Knowledge Graphs
- Describe applications of Knowledge Graphs
- Identify advantages and disadvantages of Knowledge Graphs as compared to other formalisms
- Describe and execute approaches to construct and maintain Knowledge Graphs from structured and unstructured sources, across different domains
- Construct and query Knowledge Graphs to answer questions about their content using open standards such as RDF and SPARQL
- Use Large Language Models to construct Knowledge Graphs, and to retrieve their contents
- Execute link prediction and associated graph mining techniques to enrich information in Knowledge Graphs
- Describe the FAIR principles and construct Knowledge Graph metadata using available standards
- Describe Knowledge Graph quality metrics and evaluate the quality of a Knowledge Graph
- Develop own Knowledge Graph solution for a problem of interest

Applying knowledge and understanding: Students will be able to identify requirements and steps to convert knowledge in traditional data formats to Knowledge Graph formats. Students will also be able to implement such strategies. Students will be able to query Knowledge Graphs (for instance using SPARQL query language) to answer basic to intermediately advanced questions. Students will be able to implement basic reasoning strategies on Knowledge Graphs to answer intermediately advanced questions, which cannot be answered by SPARQL queries alone. Students will be able to implement popular methods to integrate different data sources by transferring them into a Knowledge Graph. Students will be able to enrich existing Knowledge Graphs with missing information using basic predictive algorithms. Students will be able to perform basic data quality assessment on Knowledge Graphs. Students will be able to assess the degree of compliance that Knowledge Graphs have with FAIR principles.

Making judgements: Students will be able to select which tools are most suitable for constructing, querying, visualising & reasoning with Knowledge Graphs. Students will be able to differentiate between different types of Knowledge Graphs, according to their representation, coverage and content. Students will be able to select which Knowledge Graph is appropriate for answering a particular question. Students will be able to diagnose incompleteness in a Knowledge Graph with respect to answering a particular question. Students will be able to evaluate the data quality and FAIRness of a Knowledge Graph.

Communication: Students will be able to explain the advantages of representing information on the web in Knowledge Graphs. Students will be able to communicate the steps required to convert information to a Knowledge Graph format. Students will be able to communicate to non-experts the main content and representational components of a Knowledge Graph. Students will be able to outline to non-experts the steps required to answer a question by querying a Knowledge Graph.

Learning skills: Students will be able to reflect critically on the challenges and open problems remaining in Knowledge Graphs research. Students will be able to formulate and propose strategies to answer complex questions using Knowledge Graphs. Students will be able to assess the feasibility of different combinations of methods for answering questions using Knowledge Graphs.

Study material: Slides for the labs and lectures will be released on Canvas just before the respective session in PDF format.

Assessment: Individual project for application of knowledge and three group assignments to demonstrate understanding of core concepts. Assessments will be released in PDF format on Canvas according to the dates indicated in the previous slide for evaluation.

Recommended literature: Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., Melo, G. D., Gutierrez, C., ... & Zimmermann, A. (2021). Knowledge graphs. ACM Computing Surveys (Csur), 54(4), 1-37.

Additional literature: A Semantic Web Primer. 3rd Edition. Grigoris Antoniou, Paul Groth, Frank van Harmelen and Rinke Hoekstra. 2012. MIT Press, ISBN: 9780262018289.
Semantic Web for the Working Ontologist. 3rd Edition. James Hendler, Fabien Gandon, Dean Allemang. 2020. Morgan Kaufmann. ISBN-13: 978-1450376174; ISBN-10: 1450376177.
Practical RDF. Shelley Powers. 2003. O'Reilly Media, Inc. ISBN: 9780596002633.
Learning SPARQL. Bob DuCharme. 2011. O'Reilly media, Inc. ISBN: 9781449306595.
Programming the Semantic Web. Toby Segaran, Colin Evans, Jamie Taylor. 2009. O'Reilly Media, Inc. ISBN: 9780596153816.

ECTS: 6

Period 1.5 & 2.5

Information Retrieval and Text Mining (KEN4153)

Examiner: Prof dr. ir. J.C. Scholtes

Desired prior knowledge: None.

Prerequisites: None.

Description: Normal search is about “Finding the needle in the haystack”. This course focusses on a more complex problem: “How does the needle look like and where is the haystack? Also explain me why!”

Building a full-text search engine may look trivial, but it is not! How do you search hundreds of billions of documents that can be located anywhere, with sub-second responds times? How do you find exactly what you are looking for without missing relevant information or having to plough through hundreds of irrelevant documents? How can you find if you do not know exactly what you are looking for? How can you find information which is deliberately hidden? How do you know that your search engine has given you the right information? Where does it come from? Is the answer factually correct?

In this course, we will teach you how to address these questions in three steps: (1) how is a search engine is constructed, optimized and used effectively, (2) How can techniques from the word of text-mining, information extraction, text classification, clustering, topic modeling and data visualization add to a better search experience, and (3) What is the best way to integrate chatbots with search engines. How to best guarantee factuality, avoid hallucinations and provide provenance and explainability of the chatbots’ recommendations.

Linear Algebra, Statistics, Deep Learning and Natural Language Processing play an important role in this course.

This course is complementary to the course Advanced Natural Language Processing (ANLP). Overlap is reduced to the necessary minimum. Both courses can be followed in any particular order. In the Information Retrieval and Text Mining course we focus more on creating a optimal search experience, in the Advanced Natural Language Processing course, we do a deep dive into the algorithms and models used for different language-related problems such as machine translation, abstracting, and dialogs with chatbots. Tutorials are shared between the two courses.

Knowledge and understanding: The student will be able to select, understand and apply different phases and methods used to create applications that exhibit an optimal search experience or provide excellent analytical insights for natural language. In addition, the student learns to evaluate the quality of such methods according to best-practice standards as used in the field.

Applying knowledge and understanding: Students will be able to recognize applications of text mining, information retrieval and conversational AI in different domains such as consumer search, legal services, medical research, regulatory oversight, compliance, digital humanities, and customer services. After the course, the student can formulate an opinion or course of action when dealing with text-based AI-problems based on incomplete, limited and in part unreliable information. After the course, students can apply their knowledge and understanding in a manner that shows a scientific approach to their work or vocation. They are able to handle complex and ill-defined text-based problems for which it is not a priori known if there is an appropriate solution, they know how to acquire the necessary information to solve the sub-problems involved, and they know how to proceed with problems for which there is no standard or reliable route to the solution.

Making judgements: Upon completion of the course, students are able to recommend the most appropriate methods from the fields of text mining, information retrieval and conversational AI when confronted with problems involving search and analysis of textual unstructured data.

Communication: Students are able to communicate the (dis)advantages of several methods from the field of text mining and information retrieval to both an audience of non-experts.

Learning skills: After the course, the student has developed those learning skills that are necessary for a successful further career in text mining or information retrieval at the highest professional level. The student will be able to continue to develop their text-mining and information retrieval skills. The student is able to detect missing knowledge and abilities and to deal with them appropriately by finding and consulting resources that can help them to fill the gaps and new developments.

Study material: A syllabus and copies of the course slides will be used.

Recommended literature (not mandatory): Introduction to Information Retrieval. Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze. Cambridge University Press, 2008. In bookstore and online: http://informationretrieval.org and Feldman, R., and Sanger, J. (2006). The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press.

Assessment: The result of various Colab tutorials and a project contributes 30% to the final examination of the course. The other 70% is determined by the theoretical exam. The theoretical exam is open book. For the project, students can select a research topic and a text corpus from the provided list (or another relevant open source collection) and implement a number of relevant search, text-mining or conversational AI operations by using the methods discussed in the course. The delivery of the project are the results of the experiments (presented at the end of the course) and a report discussing the methods used and the quantitative quality of the efforts undertaken.

ECTS: 6

Introduction to Quantum Computing for AI and Data Science (KEN4155)

Examiners: Dr. Menica Dibenedetto & Dr. Georgios Stamoulis

Desired prior knowledge: Probability theory, linear algebra, design and analysis of algorithms

Prerequisites: None.

Description: In this course, we lay down the foundations and basic concepts of quantum computing. We will use the mathematical formalism borrowed from quantum mechanics to describe quantum systems and their interactions. We introduce the concept of a quantum bit and discuss different physical realizations of it. We then introduce the basic building blocks of quantum computing: quantum measurements and quantum circuits, single and multi-qubit gates, the difference between correlated (entangled) and uncorrelated states and their representation, quantum communication, and basic quantum protocols and quantum algorithms. Finally, we discuss the different types of noise involved in real quantum computers (coherent and incoherent errors, state preparation, projection and measurement) and their effect on performance, and outline current efforts for mitigating the issues.

Knowledge and understanding: Students will learn the fundamental principles and concepts behind quantum computing, protocols, and algorithms. Students will understand the differences between classical and quantum computation, and where the (theoretical) computational power of quantum machines comes from. Students will also get to understand the current challenges in building and using quantum computers.

Applying knowledge and understanding: Students will be able to apply existing quantum algorithms as black-box to various simple computational problem. Students will further be able to analyse simple quantum algorithms for different computational problems.

Making judgements: Students will be able to judge how the potential computational power of quantum computing can be leveraged, and how it can be applied to other fields in a beneficial way.

Communication: Students will be able to discuss quantum computation critically and judge not only its benefits but, equally important, its shortcomings. Students will especially be able to communicate potential benefits of quantum computation to the fields of artificial intelligence and data science.

Learning skills: Students will practice learning entirely new computational concepts, and how to relate existing concepts (classical computation) to new concepts (quantum computation). Students will learn to critically reflect on both the scientific literature and the societal expectations. Students will learn to self-study from state-of-the-art research articles, when classical text-books are not available.

Study material: To be announced.

Exam: Written exam (100%)

ECTS: 6

Symbolic Computation and Control (KEN4252)

Examiner: dr. ir. Philippe Dreesen

Desired prior knowledge: Linear Algebra, Calculus, Mathematical Modelling.

Description: This course consists of two interrelated parts. In the first part, we focus on basic techniques for the digital control of linear dynamical systems using feedback. We start by addressing system stability and we discuss the technique of pole placement by state feedback to solve the regulation problem. Then we introduce state observers to solve the regulation problem by output feedback. Next, we extend our scope to tracking problems. This involves the design of additional dynamics to characterize the relevant class of reference signals, which are then integrated with the earlier set-up for output feedback. Finally, we discuss the classical topic of optimal control, which can be employed to avoid using prototype systems for pole placement, and which allows the user to design a feedback law by trading off the cost involved in generating large inputs against the achieved tracking accuracy. In the second part, we address computational issues, related to the field of systems and control. Classically, computers have been designed primarily to perform approximate numerical arithmetic. Modern software packages for mathematical computation, such as Maple and Mathematica, allow one to perform exact and symbolic computation too. We shall explore this new area. It is demonstrated how speed, efficiency and memory usage considerations often lead to surprising and fundamentally different algorithmic solutions in a symbolic or exact context. Applications and examples involve stability of linear systems, model approximation, and linear matrix equations with free parameters. Practical classes serve to demonstrate the techniques and to make the student familiar with exact and symbolic computation.

Knowledge and understanding: Students familiarize themselves with state and output feedback to achieve control of dynamical systems. Concretely, they learn to (mathematically) build a basic stabilizing feedback controller for a linear input-output dynamical system, using a combination of different design techniques. Students learn methods for exact numerical and symbolic computation, as used in algebraic computation with unspecified parameters. They also learn in which ways these are different from the more commonly used approximate numerical (floating-point) methods: in terms of accuracy, speed (complexity), and memory usage.

Applying knowledge and understanding: Students will be able to construct and implement, for a given linear dynamical input-output system: (a) stabilizing state feedback, (b) full state observer, and (c) additional dynamics to perform tracking of a specified output trajectory. They will also be able to assess the quality of a controller, regarding an optimal control LQ criterion, and in view of the desired settling time and the trajectory approximation. Students will be able to determine the stability of a given linear dynamical system in an exact and/or symbolic algebraic way. They will also be able to efficiently solve linear systems of (matrix) equations involving symbolic parameters, avoiding pitfalls, which arise from techniques from approximate numerical computation.

Making judgements: Students will be able to judge the quality of a feedback design for stabilization (regulation) or tracking. Students will be able to indicate which exact and symbolic computation methods will and will not be useful for a given parameterized problem, regarding speed and memory usage.

Communication: Students will be able to motivate the design of a feedback controller, the construction of a trajectory approximation, the design of a full state observer, and the implementation choices of the weights in LQ-design. They will be able to explain the concept of feedback in the area of control. Students can adequately discuss speed and efficiency properties of an algorithm (approximate numerical, exact numerical, symbolic) to specialists and non-specialists.

Learning skills: Students will be able to read and interpret basic scientific literature on control theory and on numerical and symbolic computation. They can use Matlab and the Control Toolbox and work out ideas computationally. Students can use some of the exact and symbolic functionality of Mathematica and work out ideas computationally.

Study material: Syllabus, provided on the study portal. Handouts.

Recommended Literature: Richard J. Vaccaro, Digital Control - A State-Space Approach, McGraw-Hill International Editions, 1995. ISBN 0-07-066781-0.Exam: Written exam by computer in two parts, each having a weight of 50% on the final grade: one midterm take-home exam with Matlab on part 1 (control), one final classroom exam with Mathematica on part 2 (symbolic computation). The resit exam is on both parts of the course in a classroom setting.

ECTS: 6

Computer Vision (KEN4255)

Coordinator & examiner: Dr. Mirela Popa

Tutors: Subilal Vattimunda Purayil, Aashutosh Ganesh

Desired prior knowledge: Basic knowledge of Python, linear algebra and machine learning. This course offers the basics on image processing although prior knowledge is also a plus.

Prerequisites: None.

Description: Can we make machines look, understand and interpret the world around them? Can we make cars that can autonomously navigate in the world, robots that can recognize and grasp objects and, ultimately, recognize humans and communicate with them? How do search engines index and retrieve billions of images? This course will provide the knowledge and skills that are fundamental to core vision tasks of one of the fastest growing fields in academia and industry:

visual computing. Topics include introduction to fundamental problems of computer vision, mathematical models and computational methodologies for their solution, implementation of real life applications and experimentation with various techniques in the field of scene analysis and understanding. In particular, after a recap of basic image analysis tools (enhancement, restoration, color spaces, edge detection), students will learn about feature detectors and trackers, fitting, image geometric transformation and mosaicing techniques, texture analysis and classification using unsupervised techniques, face analysis, deep learning based object classification, detection and tracking, camera models, epipolar geometry and 3D reconstruction from 2D views.

Knowledge and understanding: Students will be able to apply the most suitable techniques for image pre-processing (e.g. enhancement, restoration), feature extraction, texture analysis, perspective geometry, camera models and topics on object recognition. In addition, they will be able to identify the most suitable techniques in a series of visual computing problems.

Applying knowledge and understanding: Students will be able to choose and/or construct solutions in a variety of professional/vocational contexts requiring image processing and computer vision (robotics, manufacturing, AI, web applications, surveillance). They will be able to build and assess methodologies for handling real-world complex problems in computer vision, making use of pre-existing data for training their models.

Making judgements: Students will be able to choose and combine methods to tackle real-world computer vision problems, captured in real-life settings and having no obvious solutions. They will be able to propose and build techniques combining computer vision methods with machine learning instruments for scene understanding and object recognition.

Communication: Through small research projects, students will be able to communicate their findings and explain the rationale behind their choices in computer vision techniques for image/video analysis.

Learning skills: After successful completion of the course, students will be able to analyze images and videos and retrieve or process content in order to derive useful information, applicable in a variety of domains (e.g. satellite imagery, surveillance, robotics, medical imaging, ambient assisted living).

Study material: A syllabus and copies of the course slides will be used along with the recommended literature.

Assessment: Written exam (50%) and two assignments (50%)

Recommended literature:
- “Computer vision: algorithms and applications”. Szeliski, Richard. Springer Science & Business Media, 2010 (available online)
- “Computer Vision: A Modern Approach, 2nd Edition”. David A. Forsyth, University of Illinois at Urbana-Champaign .Jean Ponce, Ecole Normale Superieure, Paris
- “Computer Vision: Models, Learning and Inference”, Simon J.D. Prince 2012.

Additional literature:
- “Digital Image Processing”, Rafael C. Gonzalez & Richard E. Woods, Pearson, 3rd Edition, 2016.
- “Machine Vision: Automated Visual Inspection and Robot Vision”, David Vernon, Prentice Hall, (available online at: https://homepages.inf.ed.ac.uk/rbf/BOOKS/VERNON/)
- OpenCV/Pytorch/Tensor Flow tutorials:
  https://docs.python.org/3/tutorial/
  https://pytorch.org/tutorials/
  https://colab.research.google.com
  https://docs.opencv.org/4.x/d9/df8/tutorial_root.html

ECTS: 6

Intelligent Search and Games (KEN4123)

Examiners: Prof. dr. Mark Winands & dr. Dennis Soemers.

Desired prior knowledge: Data Structures & Algorithms

Course description: In this course, the students learn how to apply advanced techniques in the framework of game-playing programs. The following subjects will be discussed:
1. Basic search techniques. Alpha-beta; A*.
2. Advanced search techniques. IDA*; B*, transposition tables; retrograde analysis and endgame databases; proof-number search and variants; multi-player search methods; Expectimax and *-minimax variants.
3. Heuristics. killer moves; history heuristic, PVS; windowing techniques; null-moves; forward-pruning techniques; selective search.
4. Monte Carlo methods. Monte Carlo Tree Search (MCTS) techniques, enhancements, and applications; AlphaGo and AlphaZero approaches.
5. Video game AI techniques: World representations, GOAP, hierarchical task networks, behaviour trees.

Knowledge and understanding: The student can explain basic and advanced search techniques and can identify which of them to use either in a game context, or in problems with a similar structure. Applying knowledge and understanding: Students have obtained the knowledge to develop, program, analyse, and apply advanced techniques autonomously to a wide variety of problems. They will also learn that adapting known techniques to fit a given problem can achieve a better performance.

Making judgements: Students will be able to judge the quality of approaches (systems or scientific publications) based on the techniques taught.

Communication: Students will be able to present the results of their game programs and search algorithms to specialists or non-specialists.

Learning skills: Students will be able to familiarize themselves with Game AI techniques beyond the scope of the course in order to solve a problem.

Study material: Course notes and other information made available.

Recommended Literature:
- Millington, I. (2019). Artificial Intelligence for Games, 3rd Edition, CRC Press, ISBN: 978-1138483972
- Russell, S.J. and Norvig, P. (2020). Artificial Intelligence: A Modern Approach, 4th edition.
- Pearson. ISBN‑ 0-13-461099-7.
- Yannakakis, G.N. and Togelius, J. (2018) Artificial Intelligence and Games, Springer, Berlin. ISBN 978-3-319-63519-4 (eBook) 978-3-319-63518-7 (hardcover)

Assessment: Written exam (50%) + a large practical task (50%).

ECTS: 6

Quantum Algorithms (KEN4235)

Examiner: Dr. Georgios Stamoulis

Desired prior knowledge: Fundamentals of Quantum Computation, Very Good command of Linear Algebra, Algorithms and Complexity

Prerequisites: Introduction to Quantum Computing for AI and Data Science

Description: This course will provide a thorough examination of the most important Quantum Algorithms. We will see how the quantum mechanical formalism gives rise to a new algorithmic design paradigm with the potential of performing certain computational tasks faster than we could do using a classical computer. The course will start with some basic algorithms like Bernstein-Vazirani and Simon's algorithm, then we will move on to Quantum Fourier Transform and Phase Estimation. Then, a thorough discussion of Shor's celebrated algorithm for factoring will follow, together with a detailed coverage of Grover's unstructured search algorithm, its optimality, adaptations, and applications. Further, we will move on to the HHL algorithm for solving systems of linear equations, a crucial component of many quantum algorithms, including Machine Learning quantum algorithms. In the last part of the course, we will present algorithms for quantum simulation, discuss quantum walks, and basics of quantum complexity theory by introducing and discussing the BQP and QMA complexity classes.

Knowledge and understanding: Students will learn how and why certain computational tasks can be performed faster in a quantum computer, what are the major techniques used in the design of such algorithms and, equally important, what are the limitations of the quantum algorithm design.

Applying knowledge and understanding: Students will be able to apply these theoretical techniques to design and analyze algorithms for many problems that could benefit from a quantum computation point of view.

Making judgements: Students will be able to judge whether proposed quantum algorithms indeed offer speedups over classical ones and how they may be able to achieve that.

Communication: Students will practice technical communication of research work in this area, describing and critically evaluating the work's contributions.

Learning skills: Students will learn from lectures/textbook/notes and then use this knowledge to read relevant research papers.

Study material: Lectures, textbook.

Assessment: 75% final exam, 25% in-class presentation summarizing a topic of relevant interest in class.

Recommended literature: Quantum Computation and Quantum Information: 10th Anniversary Edition Anniversary Edition, Michael A. Nielsen, and Isaac L. Chuang

Additional literature: Papers, notes and other relevant material will be distributed in class.

ECTS: 6

Quantum AI (KEN4236)

Coordinator & examiner: Dr. Menica Dibenedetto

Tutors: Vincenzo Lipardi

Desired prior knowledge: Linear Algebra, Classical Machine Learning

Prerequisites: Introduction to Quantum Computing for AI and Data Science

Description: This course explores the groundbreaking intersection of quantum computing and artificial intelligence, focusing on how quantum technologies can potentially revolutionize AI paradigms. The curriculum delves into quantum algorithms tailored for AI tasks, addressing complex problems that are currently intractable for classical computers. Students will gain an understanding of how quantum principles can enhance machine learning algorithms, improve optimization tasks, and facilitate data processing capabilities. Through theoretical lessons and practical laboratory sessions, students will learn about quantum mechanics fundamentals applicable to AI, quantum circuit design, and quantum algorithm development. Special emphasis will be placed on hybrid models that integrate classical and quantum computing techniques to solve real-world problems. The course will provide a mix of both theoretical and technical insights, as well as practical implementation details by using the main quantum programming languages and quantum software available.

Formal models that will be investigated: Various QAI algorithms and their possible applications for near term devices will be presented. The students will be guided through the steps of creating effective quantum models for supervised and unsupervised tasks and its evaluation in the near-term devices. Discussions will include essential quantum AI algorithms and quantum generalizations of classical learning models. Various quantum machine learning models including quantum neural networks, quantum support vector machines and quantum kernel estimator will be discussed in detail. Quantum algorithms for decision problems based on Hamiltonian time evolution, quantum search models based on Grover algorithm and quantum game theory will be introduced. A significant focus will be on developing efficient methods for encoding data into quantum states, one of the main problems in the current state of machine learning. We will then explore quantum machine learning algorithms applied to state preparation focusing on loading the underlying probability distribution of the dataset, as Quantum Generative Adversarial Networks (GANs) and Quantum Boltzmann Machine.

Knowledge and understanding: Students will gain a deep understanding of the intersection between artificial intelligence and quantum computing, learning to assess the capabilities and limitations of current quantum technologies in enhancing AI applications, exploring different quantum machine learning models.

Applying knowledge and understanding: Learners will apply theoretical concepts in practical settings, developing and implementing quantum algorithms and models that can be applied to machine learning and optimization real-world problems.

Making judgements: Students will evaluate the effectiveness of quantum AI solutions, making informed decisions about when and how to implement these technologies. This course will formulate and answer the questions: “how quantum computing can provide a computation boost to AI, enabling it to tackle more complex problems?” and “how can AI produce functional applications with quantum computers?”.

Communication: Participants will enhance their ability to articulate complex quantum AI concepts clearly and effectively to a diverse audience, including those without a background in quantum physics.

Learning Skills: Learners will develop critical thinking and problem-solving skills in quantum computing and AI, fostering a mindset of continuous learning and adaptation to new technologies.

Study Material: Quantum Machine Learning Texts, Online Quantum Computing Simulators, Peer-reviewed Journal Articles

Assessment: Assignment based

Recommended literature: "Machine Learning with Quantum Computers" by M. Schuld, F. Petruccione, Second Edition

Additional literature: Research articles and papers will be provided throughout the course.

ECTS: 6

Quantum Information and Security (KEN4237)

Examiner: Dr. David Mestel

Desired prior knowledge: Quantum states, operators and measurements

Prerequisites: Introduction to Quantum Computing for AI and Data Science

Description: In this course we will consider the power of quantum mechanics not in accomplishing computational or 'algorithmic' tasks, but instead for communication- and security-related tasks. The strange properties of the quantum world turn out to be remarkably useful for these. For example, we can exchange secret messages in a way that is unconditionally secure: secrecy is guaranteed by the physical laws of nature, rather than (as in ordinary cryptography) based on an assumption that a particular computational problem is too hard for the adversary.

We will begin by covering the theoretical techniques needed to study security-related protocols, where it is fundamental that some parties will not know what state a particular quantum system is in. After a thorough grounding in the 'density matrix' formalism which is used to represent this uncertainty, we will cover quantitative measures of this kind of uncertainty, for instance quantum versions of classical entropy. We will then look at a variety of protocols (e.g. quantum money, quantum key distribution,...), and how to define and prove the desired properties.

Knowledge and understanding: Students will learn to use the `density matrix' formalism to reason about quantum states under uncertainty, and about quantitative measures of uncertainty and entanglement. They will also learn about the fundamentally `contextual' nature of quantum mechanics, which is the foundation for all of the protocols we will study.

Applying knowledge and understanding: Students will be able to apply these theoretical techniques to analyse protocols and prove that they have desirable security properties.

Making judgements: Students will be able to judge whether protocols are suitable for particular goals.

Communication: Students will practice technical communication of research work in this area, describing and critically evaluating the work's contributions.

Learning skills: Students will learn from lectures/textbook and then use this knowledge to read a research paper which they will present in class.

Study material: Lectures, textbook.

Assessment: 70% final exam, 30% in-class presentation summarising a research paper in the topic.

Recommended literature: T. Vidick and S. Wehner, `Introduction to quantum cryptography', Cambridge University Press 2024

Additional literature: M. Wilde, `Quantum information theory', Cambridge University Press 2013

ECTS: 6

Period 2.4

Agents and Multi-Agent Systems (KEN4111)

Coordinator & examiner: Prof. dr. Gerhard Weiss

Desired prior knowledge: Basic knowledge and skills in programming.

Description: The notion of an (intelligent) agent is fundamental to the field of artificial intelligence. Thereby an agent is viewed as a computational entity such as a software program or a robot that is situated in some environment and that to some extent is able to act autonomously in order to achieve its design objectives. The course covers important conceptual, theoretical and practical foundations of single-agent systems (where the focus is on agent-environment interaction) and multi-agent systems (where the focus is on agent-agent interaction). Both types of agent-based systems have found their way to real-world applications in a variety of domains such as e-commerce, logistics, supply chain management, telecommunication, health care, and manufacturing. Examples of topics treated in the course are agent architectures, computational autonomy, game-theoretic principles of agent-based systems, coordination mechanisms (including auctions and voting), and automated negotiation and argumentation. Other topics such as ethical or legal aspects raised by computational agency may also be covered. In the exercises and in the practical part of the course students have the opportunity to apply the covered concepts and methods.

Formal models that will be investigated: Coordination and interaction models from game theory and social choice theory.

Knowledge and understanding: The student is able to describe and explain single- and multi-agent concepts and methods, and to analyse their strengths and shortcomings.

Applying knowledge and understanding: The student is be able to apply the gained knowledge in concrete application scenarios and practical applications.

Making judgements: The student is be able to judge for a given problem whether and in how far it is beneficial to use an agent-based approach for its solution.

Communication: The student is able to motivate and explain benefits and shortcomings of their usage in a given application, and thereby showing sufficient understanding of single- and multi-agent concepts.

Learning skills: The student is able to study independently and critically literature on single- and multi-agent technology, including, in particular, literature describing new developments in the methods and techniques covered in this course.

Study material: Course slides; supplementary material to be announced.

Assessment: Practical assignments (30%) and written exam (70%)

Recommended literature:
- Stuart Russell and Peter Norvig (2010). Artificial Intelligence. A Modern Approach. 3rd edition. Prentice Hall.
- Gerhard Weiss (Ed.) (2013, 2nd edition): Multi-agent Systems. MIT Press.
- Mike Wooldridge (2009, 2nd edition): An Introduction to Multi Agent Systems, John Wiley & Sons Ltd.
- Yoav Shoham and Kevin Leyton-Brown (2009): Multi-agent Systems. Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press.

ECTS: 6

Period 2.5

Autonomous Robotics Systems (KEN4114)

Examiner: Dr. Rico Möckel.

Desired prior knowledge: Discrete Mathematics, Linear Algebra, Probabilities and Statistics, Data Structures and Algorithms, Machine Learning, Search Techniques.

Prerequisites: None.

Description: Operating autonomously in unknown and dynamically changing environments is a core challenge that all robotic systems must solve to work successfully in industrial, public and private areas. Currently popular robotic systems that must demonstrate such capabilities include self-driving cars, autonomously operating drones, and personal robotic assistants. In this course, students obtain deep knowledge in creating autonomous robotic systems that can operate in unknown and dynamically changing environments by autonomously modelling and navigating in such environments. Students learn to approach these challenging tasks through three main techniques: swarm intelligence, model-based probabilistic frameworks, and (mostly) model-free techniques from artificial evolution and machine learning.

Knowledge and understanding: Students gain a deep understanding of the challenges in autonomous robotic systems and how these challenges are addressed in state-of-the-art systems. Students learn about and practice techniques for autonomous mapping, localization, navigation, sensing, modelling robot motion, planning, and decision-making. Through the course, students obtain in-depth knowledge and hands-on experience in a variety of algorithms and techniques including Bayesian filters (like Kalman Filters, Extended Kalman Filters, Histogram Filters, and Particle Filters), artificial neural networks, evolutionary algorithms, and swarm intelligence.

Applying knowledge and understanding: After successful completion of the course, students will have obtained in-depth knowledge to understand, adapt, apply, and autonomous robotics systems. Students obtain the ability to select from a variety of available tools feasible solutions for the complex and rather ill defined problem domains of autonomous robotic systems and to predict the resulting consequences of their choices. Furthermore, students learn how to choose, apply, formulate, and validate models of autonomous robotic systems and of appropriate control techniques from artificial intelligence for these systems.

Making judgements: Students will be able to comprehend and to critically judge scientific publications on autonomous systems, artificial evolution, and swarm intelligence. From this literature, students are able to search for and to critically process information to solve given ill-defined but in practice highly relevant problems in autonomous systems. Students are able to critically discuss social, economic, and ethical consequences of artificial intelligence and autonomous decision-making.

Communication: Students learn to critically discuss challenges and professional solutions in autonomous robotic applications with both experts and non-experts.

Learning skills: The course prepares students to work on robotic applications in professional research and business environments. Students will be able to autonomously acquire new skills and knowledge to develop, program, analyse and apply advanced techniques to a wide variety of problems.

Study material: Thrun et al. (2005), Probabilistic Robotics, The MIT press, ISBN-13: 978-0262201629. Lecture material and publications provided during the lecture.

Assessment: The final course grade is 80% of the final written "closed-book" exam grade plus 20% of the practical group assignments grades.

Recommended literature: Floreano and Nolfi (2000), Evolutionary Robotics, The MIT press. ISBN-13: 978-0262640565.
Dario Floreano und Claudio Mattiussi (2008), Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies, ISBN-13: 978-0262062718

ECTS: 6

Reinforcement Learning (KEN4157)

Coordinators: Dr. ir. Kurt Driessens & Dr. Dennis Soemers

Examiners: Dr. ir. Kurt Driessens & Dr. Dennis Soemers

Desired prior knowledge: Machine Learning

Prerequisites: None.

Description: Reinforcement learning is a type of machine learning problem in which the learner gets a (delayed) numerical feedback signal about its demonstrated performance. It is the toughest type of machine learning problem to solve, but also the one that best encompasses the idea of artificial intelligence as a whole. In this course we will define the components that make up a reinforcement learning problem and will see what the important concepts are when trying to solve such a problem, such as state and action values, policies and performance feedback. We will look at the different properties a reinforcement learning problem can have and what the consequences of these properties are with respect to solvability. We will discuss value based techniques as well as direct policy learning and learn how to implement these techniques. We will study the influence of generalisation on learning performance and see how supervised learning (and specifically deep learning) can be used to help reinforcement learning techniques tackle larger problems. We will also look at the evaluation of learned policies and the development of performance over time.

Formal models that will be investigated: Markov Decision Processes

Knowledge and understanding: Students will be able to explain the setup of a reinforcement learning problem and list its formal components, explain the difficulties faced when adding function approximation to reinforcement learning, explain the origins of the learning signal for policy gradient methods for reinforcement learning.

Applying knowledge and understanding: Students can implement and apply online and offline tabular techniques of value based reinforcement learning algorithms, apply the use of function approximation in value based reinforcement learning algorithms, implement and apply policy gradient methods for discrete and continuous action tasks and deep learning methods to reinforcement problems.

Making judgements: Students will be able to judge the suitability of reinforcement learning techniques as a solution for an AI problem, choose/select between exploration and exploitation tradeoff methods suited to the problem faced, interpret and judge the results of a reinforcement learning agent.

Communication: Students will gain a working knowledge of reinforcement learning as a problem, and of the state of the art in solution techniques and will be able to motivate his/her choices concerning the application of these techniques.

Learning skills: Students will learn that the state of the art in reinforcement learning continues to develop at a rapid pace and that becoming and staying an expert in the domain will require continued learning.

Study material: Course slides to support the lectures; supplementary material consisting of research papers and book chapters.

Assessment: Assessment for this course is based on the construction of a portfolio with which students prove that they attained all learning goals, at which point they will pass the course. The level of a passing grade is determined by the quality of a large final research and implementation assignment.

Recommended literature: Reinforcement Learning: An Introduction by R. Sutton and A. Barton

Additional literature: Algorithms for Reinforcement Learning by C. Szepesvári; Reinforcement Learning and Optimal Control by D. Bertsekas

ECTS: 6

3.11 Master's thesis AI & DSDM (KEN4160 & KEN4260)

The Master's Artificial Intelligence and Data Science for Decision Making will be completed by writing a master's thesis. The thesis is produced individually and is the result of a master's research project that runs during the second semester of year 2 of the master's programme. In the preliminary phase, the emphasis is on self-study, subject determination, planning and some preliminary research. Then the actual research is started. The final phase is used to finalize the master's thesis. The master's project is completed by a public on-site presentation and discussion of the results (also known as a defence of the thesis). The master's thesis is supervised by one of the senior researchers of the Department of Advanced Computing Sciences. In principle, there should be no confidentiality agreements for a thesis, and staff members cannot be expected to commit to these.

Exam: Master's thesis and presentation.

ECTS: 30

** Note that when you enrol in February, you follow your electives in period 2.4, 2.5 and 2.6 and work on your master's thesis in period 2.1, 2.2, 2.3.*

Master's thesis Artificial Intelligence and Data Science for Decision Making

Master thesis is an individual work. The research is done by the guidance of the supervisor(s), the thesis is written individually, taking a feedback of the supervisor into account, and the thesis is concluded by a thesis defense. If relevant, the thesis is handed in together with the source code, data, or any other supporting material. In order to start working on the thesis, a student needs to have obtained at least 70 ECTS (among which are 40 credits of the first year).

General procedure
The process of writing a master's thesis consists of 6 phases. With the exception of the first phase, it is scheduled in the last semester of the master's study. The time frame given below is an indication for these phases.

Phase 1: Topic selection
At the end of the first semester of an academic year, the students are informed of the main directions of research in the research areas at the Department of Advanced Computing Sciences. Next to this, the master's programmes maintains a website of concrete thesis topics, which is updated annually. Based on this information, students acquire more information about specific possibilities in the areas by means of individual discussions with relevant researchers available. These discussions take place upon the initiative of the student.

Phase 2: Thesis Research Plan
Before the actual work on the thesis, each student must have chosen a thesis topic and a principal thesis supervisor. The student creates a thesis research plan, which is to be signed by the student and the two prospective thesis examiners, and then handed over to the master's thesis coordinator. The plan is sent to the Board of Examiners for approval.

The students will be invited to present their thesis topic, and the related work in a Master-thesis seminar, in the presence of fellow students and thesis supervisors.

Phase 3: Research
After the thesis plan has been approved, the student carries out his/her own research. This research process will be guided by the thesis supervisor through a series of regular appointments, preferably on a weekly basis.

The students will be invited to present the first research achievements, with audience staff members of the Department of Advanced Computing Sciences and fellow students.

Phase 4: Writing
At the end of the main research phase, the focus is on writing of the thesis. The student can expect at least one feedback on the text from the supervisor, before handing-in the final version of the thesis. The two examiners will evaluate the final version of the thesis shortly before the thesis defence.

Phase 5: Preparation for presentation
In the last week, , the student prepares a final presentation of the thesis research. This individual presentation will have a maximum length of 30 minutes, followed by 15 minutes of discussion between the student and the two examiners.

Phase 6: Presentation
The master's thesis defense takes place upon agreement of the supervision team. The defence is public and takes place on the premises of the department.

Requirements and assessment
For the master's thesis research, every student has to conduct a short scientific research. This can be an empirical or a theoretical research. The topic is open, as long as it fits into the field of the master's programme. Staff of the Department of Advanced Computing Sciences will briefly introduce their main areas of research, but students are encouraged to propose a research topic themselves. The topic and the research question have first to be approved by the prospective examiner. This plan will be signed by the student and the prospective examiners and then handed in to the Board of Examiners for the formal approval. It is possible to execute the master's thesis research as an external training period. This should be well defined in the master's research plan. In this case, the plan should also include the name of the company, the name of the external supervisor, the size of the project and any agreements about payment and confidentiality. The plan should also be signed by the external supervisor.

The research needs to be original in such a way that the thesis supervisor is convinced that this research has not been done before. The research also needs enough depth and still it must be possible to finish it in the set amount of time. Every thesis is an individual work.

The thesis is graded by the examiners using a standard assessment form, available on Canvas. The weighing of these aspects is up to the examiners.

Content aspects
The thesis describes the problem statement, research questions, approach and results of the research. This has to be done in a clear, structured and scientific manner. This includes:

- a clear introduction in which the problem statement and research questions are presented;
- the master's student shows proper analysis of complex issues in a new context and is able to formulate a proper problem statement;
- a clear conclusion, based solely on the already used thought out principles and derived results;
- a clear line is shown between problem statements, approach, methods and the derived results;
- a motivation of the followed approach, reflecting on standard methods and their presuppositions,
- an adequate description of the followed approach;
- a purposeful and systematic way of collecting data;
- an honest, clear and concise description of the derived results, if necessary using tables;
- an analysis and discussion of the results;
- the usage of relevant and recent literature for the reasoning in the thesis.
- the correct usage of references.

Design aspects
Correct scientific references have to be used. Images and tables are accompanied by an index and caption. Mathematical formula, definitions, etc. have to be properly designed and numbered. The start and end of mathematical formulae have to be properly defined.

Language aspects
The thesis has to be written in English, considering correct spelling, syntactical structure of sentences and structure of content in paragraphs. The target audience consists of fellow master's students and lecturers. Any jargon and/or abbreviations have to be explained unless they are common knowledge for this audience.

Citations
It is allowed to use several short citations. These citations have to be clearly referenced and have to be typographically distinguishable (that is, citations are placed in quotes). Non-allowed citations or missing references will result in a non-pass.

4 Facilities for Students

In this chapter, you will get an overview of the facilities that Maastricht University offers its students.

4.1 Student Affairs Office

The Student Affairs Office, among other things, takes care of the organization and administration of the education.

Visiting address: Paul-Henri Spaaklaan 1, 6229 GT Maastricht
Postal address: P.O. Box 616, 6200 MD Maastricht, the Netherlands.

Office hours:
PHS, C.1006 daily between 10h00 - 11h00 and 15h00 - 16h00.

Contact:
Admissions: dacs-admissions@maastrichtuniversity.nl
Tel.: +31(0)43 388 26 77

Exam Administration: dacs-examination@maastrichtuniversity.nl
Tel.: +31(0)43 388 35 25

Scheduling: dacs-scheduling@maastrichtuniversity.nl
Tel.: +31(0)43 388 35 25

International Relations (exchange): dacs-iro@maastrichtuniversity.nl

Student Affairs Office: dacs-studentaffairs@maastrichtuniversity.nl

4.2 Administrative structure of the Faculty

The administrative structure of the Faculty is laid down in the faculty regulations.

The dean is responsible for the faculty's administration. More information is to be found on the website: https://www.maastrichtuniversity.nl/nl/over-de-um/faculteiten/faculty-science-and-engineering.

Faculty Board
The Faculty Board, chaired by the dean of the Faculty of Science and Engineering, runs the Faculty. The Faculty Board is charged with the general management and administration, as well as its policy regarding academic research and education.

Faculty Council
The Faculty Council is entitled to submit proposals and present their opinion to the Faculty Board regarding any matters relating to faculty administration, policy, education and research. The Faculty Council has rights of approval, e.g. regarding faculty regulations, research programmes, and the implementation of a binding study advice, and rights of advice, e.g. regarding the budget.

Directors of Studies
The programme directors dr. Pietro Bonizzi for both Bachelor programmes and dr. Matus Mihalak for the Master programmes are responsible for the organization and coordination of all teaching activities. The Education Programme Committee (EPC) advises the programme directors.

Education Programme Committee
There is one EPC for the Bachelors Data Science and Artificial Intelligence, Computer Science and the Masters Data Science for Decision Making and Artificial Intelligence. The EPC is responsible for advising the Faculty Board, the Programme Directors and the Board of Examiners. Furthermore, the EPC is entitled to advice in any subject related to the programme, and consists out of ten members, five students and five members of the academic staff. In addition, the quality assurance officer has the role of advisor in the committee.

All correspondence for the Education Programme Committee should be addressed to dacs-secretariat@maastrichtuniversity.nl or by postal mail to:
Department of Advanced Computing Sciences - Maastricht University
P.O. Box 616, 6200 MD Maastricht.

Board of Examiners
The Board of Examiners is in charge of the organization and supervision of the examinations and is appointed by the Faculty Board.

All correspondence for the Board of Examiners should be addressed to
dacs-boe@maastrichtuniversity.nl or by postal mail to:
Department of Advanced Computing Sciences - Maastricht University
Board of Examiners,
P.O. Box 616, 6200 MD
Maastricht

Board of Admissions for the Master's Programmes
The Board of Admissions is responsible for granting the admission requests for entering a master's programme and is appointed by the Faculty Board. All correspondence for the Board of Admissions should be addressed to dacs-admissions@maastrichtuniversity.nl or by postal mail to:
Department of Advanced Computing Sciences - Maastricht University
Student Affairs Office,
P.O. Box 616, 6200 MD
Maastricht

It is possible to follow incidental courses at the transnational University Limburg (located at Hasselt University, Belgium). Students who want to make use of this possibility should individually ask permission to the Board of Examiners of the master's programmes AI and DSDM, Maastricht University. More information on the transnational University Limburg, its staff members, and information on the content of the courses can be found at: www.uhasselt.be/informatica

4.3 Teaching material

For each project in the Bachelor programmes, a project book is published. The project books and the education schedules of each period are available two weeks before the start of a new period, at the latest. The study material (= obligatory literature) or recommended literature of a course is announced at the first lecture of the course and is available on the course page in Canvas.

4.4. Participation in the Education

The students are expected to be available from Monday through Friday from 08.30 to 18.00. for educational activities.

4.5. Announcements on Educational Matters

Announcements concerning educational matters will be published through Canvas. Students are mainly approached through e-mail and through Canvas. We advise students to check for new announcements/emails daily.

4.6. Change of Address Student

Except for Canvas, the Student Affairs Office makes use of mailings to students. If there is a change in the study address or the address of the student's parents, this should immediately be changed in Studielink. Do not forget to mention the commencing date of the change. During the academic year, the student's study address is considered as their postal address.
You may contact dacs-admissions@maastrichtuniversity.nl for help.

4.7. Project Rooms

Scheduled practical lectures have priority over private use of the project rooms by students. The rooms are open to students from Monday through Friday from 08.00 until 18.00 outside of project teaching hours. Wireless internet is available throughout the whole building. On this webpage you can find out how to log in.
For questions about the system management, please refer to the system managers of the Department of Advanced Computing Sciences, tel. +31(0)43-388 54 93 or by mail:
Lo-fse@maastrichtuniversity.nl.

House rules for all project/meeting rooms;

- Users are not allowed to download illegally acquired materials;
- Users are not allowed to illegally download materials
- Users are not allowed to install illegally acquired software;
- Users should use their own devices for saving data, or save your data on your personal network drive (I:);
- Users should handle the furniture with care;
- For the regulation of the air conditioning system, students may contact the Student Support staff.

4.8. Faculty Counsellors for Students

Study Adviser
The student counsellors Tessa Fox, Wendy Brandt and Eva Knip are staff members whom you can contact if you have any questions concerning your study and can be reached at telephone number 043-3883561, in rooms C2.012, C2.014 & C2.016 at PHS 1,
or through dacs-studyadvice@maastrichtuniversity.nl.

They are familiar with the organization of the education, the faculty organization and the study. The student counsellor is a primary advisor for students. If your study comes to a standstill for whatever reason, you can contact the student counsellor. It is also the right person to talk to if you have any questions to which you cannot find any answers in the faculty prospectus or during faculty information meetings. But also in case of personal circumstances due to which your study or personal life are suffering, for instance illness, mental health problems or family circumstances, your student counsellor can listen to you. Conversations are confidential. Based on these talks the student counsellor can direct you to some further assistance. The student counsellor may also call up students for a talk if it appears that their results are falling back. More information and a scheduling tool can be found here.

Internationalization
For any questions you may have about studying for a semester at a foreign university, or about a practical training abroad, for support, and for direct information you can contact during opening hours the international relations officer of the Department of Advanced Computing Sciences Luc Giezenaars via dacs-international@maastrichtuniversity.nl.

4.9 Student Services Centre

The Student Services Centre is responsible for the preparation and execution of the policy of Maastricht University in the area of general student provisions. In short, this department has a number of specialized service units for student-related issues such as accommodation, sports, information on studies and work and career advice. In addition, there is a central information desk in the main entrance hall of the Visitors' Centre, to which current and prospective students may address their questions.
Visiting address: Bonnefantenstraat 2, Tel.: +31(0)43-388 53 88,
www.maastrichtuniversity.nl/ssc.

4.9.1 Visitors' Centre and student registration

Information Desk
The information desk in the UM Visitors' Centre at Bonnefantenstraat 2 is the first point of contact for current and new students. It provides the following services:
- Help with admission and (re)registration;
- Information on and help with visas, scholarships, bank accounts and (health) insurance;
- Changing of address;
- Payment of tuition fees;
- Cancellation of registration;
- Reimbursement of tuition fees;
- Proof of payment/registration;
- Collection of your first UM-card;
- Help with housing;
- Appointments with student deans, psychologists, and career services;
- UM gifts.

Tel.: +31(0)43-388 53 88,
e-mail address: study@maastrichtuniversity.nl
FAQ: https://www.maastrichtuniversity.nl/frequently-asked-questions-faqs
Opening hours Monday-Friday 08h30 - 18h00.

Visa and Scholarship Office
The Visa and Scholarship Office is responsible for immigration matters and scholarships for prospective and current students.
For any questions on visas, please visit our website: www.maastrichtuniversity.nl/visa
or e-mail address: visa@maastrichtuniversity.nl.

UM Career Services
UM Career Services offers workshops, job interview simulations, Quick career advise and more intensive counseling. For more information, please see www.maastrichtuniversity.nl/careerservices or contact your student counsellors Wendy, Tessa and Eva at the Department of Advanced Computing Sciences

Student Guidance
Psychological support (Student psychologists)
Student Psychologists may be consulted in case of personal problems. Examples of complaints and problems include:

- Study related problems like study stress and fear of failure;
- Psychological complaints such as anxiety, depression, eating disorders, stress-related complaints, lack of confidence, dealing with traumatic experiences.

The student psychologists can help you by means of individual guidance and/or group training (in Dutch and English). Check the current offer via: https://www.maastrichtuniversity.nl/support/your-career/lectures-workshops-and-training-courses

For making an appointment use the online tool on the website.

Study related legal support (Student Deans).
For more information: www.maastrichtuniversity.nl/studentguidance
E-mail address: studentendecanen@maastrichtuniversity.nl
Open visiting hours at the SSC, please check the website for correct timeslots

Studying with a disability, chronic illness or dyslexia
It is important to Maastricht University that students with a functional impairment can successfully complete their studies without too much delay. By functional impairment UM means all disorders that are of a permanent or temporary character. Amongst these are all motor, sensory or psychological disorders, but also non-visible disorders, such as dyslexia, chronic illness, physical complaints, depression and the like. Disability Support is available to students (with a functional impairment), prospective students, student counsellors, teachers, parents and others who are interested.

For more info: https://www.maastrichtuniversity.nl/studyingwithdisability
E-mail address: disability@maastrichtuniversity.nl
Open visiting hours: Monday - Thursday from 11h00 to 13h00.
Tel.: +31(0)43-388 52 72.

5 Staff

5.1 Academic Staff

Dr. Francesco Barile

Dr. Olivier Bilenne

Dr. Pietro Bonizzi

Dr. Martijn Boussé

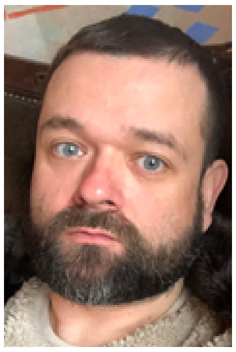
Kamil Bujnarowski

Dr. Rachel Cavill

Dr. Steven Chaplick

Dr. Remzi Celebi

Dr. Pieter Collins


Dr. Walter Crist III

Lucas Dahl

Dr. Otti D'Huys

Dr. Menica Dibenedetto

Dr. Phippe Dreesen

Dr. Kurt Driessens

Prof. dr. Michel Dumontier

Dr. Barbara Franci

Dr. Tony Garnock-Jones

Dr. Rishav Hada

Dr. Enrique Hortal Quesada

Spriha Joshi

Dr. Joël Karel

Dr. Steven Kelk

Dr. Charis Kouzinopoulos

Dr. Stefan Maubach

Dr. Matus Mihalak

Dr. Rico Möckel

Dr. Marieke Musegaas

Prof. Dr. Ir. Ralf Peeters

Dr. Eric Piette

Dr. Mirella Popa

Dr. Tjitze Rienstra

Dr. Linda Rieswijk

Dr. Ir. Nico Roos

Dr. Gijs Schoenmakers

Dr. Katharina Schneider

Dr. Christof Seiler

Dr. Yusuf Can Semerci

Prof. Dr. Jan Scholtes

Dr. Evgueni Smirnov

Dr. Jerry Spanakis

Dr. Georgios Stamoulis

Dr. Chang Sun

Dr.ir. Marijn ten Thij

Prof. Dr. Frank Thuijsman

Prof. Dr. Nava Tintarev

Prof. Dr. Gerhard Weiss

Prof. Dr. Anna Wilbik

Prof. Dr. Mark Winands

Dr. Anirudh Wodeyar

5.2 Lab assistants

Dean Boonen

5.3 Support Staff

Wendy Brandt

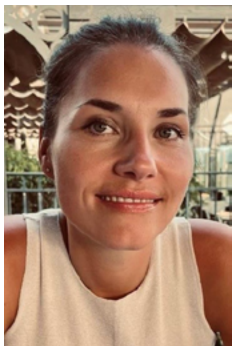
Esther Breuls

Aleksandra Draper

Céline Duijsens - Rondagh

Tessa Fox

André Fraats


Crissie Gielissen

Luc Giezenaar

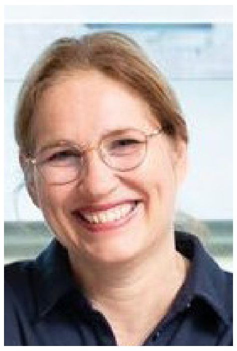
Charlotte Hamelers - Geelen

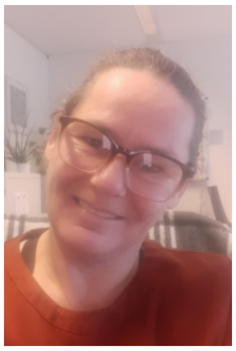
Iris Hoogsteder

Eva Knip

Astrid Lamers

Bas Lemmens


Ellen Narinx - Schrauwen


José Oudman

Desirée Parren

Anouk Quaden

Dénise van Spingelen

Miranda Vermeer

Claudia Wijler-Lardenoye