\WOOLF/

MS in CS Program Handbook

Date of Publication: August 2025

Foreword

This manual has been designed to familiarize you with the policies and procedures that shape the Woolf MS in CS Program. This manual should not be viewed as complete

and is not designed to replace the Woolf Academic Handbook. It is intended to provide information you will need in order to make decisions about your graduate studies and to acquaint you with the administrative requirements, policies and procedures you will be expected to meet that are outside the scope of the Woolf Academic Handbook. This document should thus be used in tandem with the Woolf Academic handbook. Where either manual seems incomplete, you are encouraged to inquire with your Faculty Advisor. For questions beyond the domain of your Advisor you are encouraged to reach out to help@woolf.university.

We hope that your experience within Woolf's program will be fulfilling, stimulating, and engaging. We are excited to welcome you to the Woolf community.

The Mission of Woolf

Woolf exists to promote academic excellence, broaden access to higher education, and guard values that are humane, democratic, and international. Above all, Woolf values freedom of thought, freedom of inquiry, and freedom of expression.

We do this through our commitment to high-quality education and through widening the horizon of opportunity by connecting students with quality academics across the world.

Through Woolf's world-class platform and programs students gain exposure to new ideas, new ways of understanding, and new ways of learning. By uniting exceptional faculty with motivated students, Woolf is able to build an outstanding academic community leading to journeys of intellectual transformation. From this we hope that students will share their academic, intellectual and other talents in serving their communities across the world.

Woolf is guided by the following tenets:

Background. Talent may be evenly distributed but opportunity is not – we are working to widen the horizon of opportunity by connecting students and teachers across the world.

Education. Woolf prioritizes an education that will serve its students both in the near-term and in the long-term. Woolf seeks to provide a personalized, bespoke education. In all fields, Woolf seeks to instill values of curiosity, intellectual discipline, and clarity of expression.

Research. Woolf prioritizes research-driven teaching that uses the latest academic scholarship, and Woolf encourages its students to engage in groundbreaking research.

Society. Woolf encourages partnerships with governments, educational institutions, research centers, schools, and businesses of all kinds – provided these partnerships do not infringe on the values of Woolf.

Technology. To the extent that existing or new technologies can improve the educational outcomes for students, widen access to the Woolf global network, improve the career experience of academics, better secure credible governance, lower the costs of institutional management, and generally support the mission of Woolf – these are embraced.

In all things, Woolf values excellence and measures itself against the highest international standards. Woolf seeks to raise those standards further.

Academic Program Information

Program Description

The Master of Science in Computer Science (MS in CS) is a 45-credit program that includes a variety of online courses. Students may choose to receive a general MS in CS or may specialize in one of the following areas: Artificial Intelligence and Machine Learning, Cloud Computing, Software Engineering, Data Science and Data Analytics. The program is an integrated, sequential course of study in which students obtain and demonstrate the knowledge and skills required of the computer science industry.

The MS in CS program teaches students comprehensive and specialized subjects in computer science; it teaches students cutting edge engineering skills to solve real-world problems using computational thinking and tools, as well as soft skills in communication, collaboration, and project management that enable students to succeed in real-world business environments.

Most of this program is case (or) project-based where students learn by solving real-world problems end to end. This program has core courses that focus on computational thinking and problems solving from first principles. The core courses are followed by specialization courses that teach various aspects of building real-world systems. This is followed by more advanced courses that focus on research level topics, which cover state of the art methods. The program also has a capstone project at the end, wherein students can either work on building end to end solutions to real world problems (or) work on a research topic. The program also focuses on teaching the students the “ability to learn” so that they can be lifelong learners constantly upgrading their skills. Students can choose from a spectrum of courses to specialize in a specific sub-area of Computer Science like Artificial Intelligence and Machine Learning, Cloud Computing, Software Engineering, or Data Science, etc.

The Ms in CS is delivered completely digitally by combining asynchronous components (lecture videos, readings, and written assignments) and synchronous cohort meetings attended by students and an instructor or faculty member during a video call.

The asynchronous components support the schedules of students from diverse work-life situations, and synchronous meetings provide accountability, motivation, and a sense of community presence for students. The synchronous sessions allow unparalleled access to high quality instruction and enhanced collaboration among students through using face-to-face online interaction.

Faculty conduct live office hours with students and interaction between faculty members and students, both individually and as a group, is enhanced in the online environment by blending asynchronous content with real-time student responses. Faculty and enrolled students have 24/7/365 access to technical support through Woolf’s support system.

Woolf’s digital campus allows students to complete the program in as little as 50 weeks of continuous study or within 5 years as part of a part-time course of study. The degree on the students’ transcripts is a Master of Science in Computer Science, which attests to their completion of the requirements.

Admission Requirements

The Woolf Ms in CS is a fast-paced, rigorous degree focused on teaching comprehensive and specialized subjects in business administration. Candidates should have a bachelor's degree in a technical field, or have at least 5 years of related experience with at least some undergraduate level courses or an associate's degree.

English language competency at an IELTS 6.5 or equivalent is required of all applicants.

This program is designed for individuals who wish to enhance their knowledge of computer science and its various applications used in different fields of employment. It is designed for those that will have responsibility for planning, organizing, and directing technological operations.

In all cases, the target group should be prepared to pursue substantial academic studies.

Curriculum Areas

The program is organized into a course structure of three tiered areas. Each tiered area sequentially builds off of the previous, so students must complete each tier before advancing to the next.

Each course consists of regular lessons and cumulative lessons devoted to cumulative examination. Each course requires about 75 hours to complete (see individual courses for details). A full-time student completes two lessons per week with an assignment submitted for each lesson; this pattern continues for each regular lesson in the class.

Summative examination lessons allow an appropriate amount of time for students to review and revise their prior work and deepen their synthetic grasp of the materials in preparation for cumulative examination or project.

The degree has a capstone project consisting of the Advanced Applied Computer Science Project. The capstone represents a synthesis of knowledge and skills gained throughout the graduate program. Over the course of the capstone, students use multidisciplinary approaches to perform critical analyses of real technical issues in situations of uncertainty and incomplete information and develop an actionable solution, which is presented and assessed at the end of the capstone.

Learning Outcomes

The program teaches students comprehensive and specialized subjects in computer science; it develops skills in critical thinking and strategic planning for changing and fast-paced environments; and it develops competences in leadership, including autonomous decision-making, and communication with employers, stakeholders, and other members of a team.

Knowledge

- Students will have a cutting-edge knowledge and understanding of computer science allowing them to solve real-world engineering and specific computational problems using advanced techniques at the forefront of computer science
- Students will be able to analyze the societal, regulatory, and technological contexts for key computer science applications

- Students will be able to apply their technological abilities to produce innovative solutions to real-world problems and that implement techniques learned in the course
- Students will display original thinking on the basis of the knowledge they gain in the course

Skills

- Develop advanced, innovative, and multi-disciplinary problem-solving skills
- Communicate computer science methods and tools clearly and unambiguously to specialized and non-specialised audiences
- Develop advanced abilities related to computer science operational procedures and implement them in response to changing environments
- Critically evaluate alternative approaches to solving real world engineering and technological problems using cutting edge techniques in computer science on the basis of academic scholarship and case studies, demonstrating reflection on social and ethical responsibilities
- Formulate technological judgments and plans despite incomplete information by integrating knowledge and approaches from various computer science domains including machine learning, distributed computing, and cloud computing.
- Enquire critically into the theoretical strategies for solving real-world problems using computational thinking and tools.
- Develop new skills in response to emerging knowledge and techniques and demonstrate leadership skills and innovation in complex and unpredictable contexts

Competences

- Formulate research-based solutions to practical problems in environments of incomplete information
- Manage decisions with autonomy in complex and unpredictable environments
- Organize projects and people in a way that is responsive to changes in the wider technological environment
- Demonstrate learning skills needed to maintain continued, self-directed study

Faculty

All instruction is provided by competent academics with qualifications commensurate to their role. All teachers are also expected to have relevant teaching experience in the domain of their expertise. All faculty members at Woolf are expected to be in possession of a research doctorate in the domain of their teaching or supervision; moreover, they are expected to have a record of research or a research agenda reflecting the capacity for research.

Woolf uses clear, fair, and transparent processes for teaching recruitment, conditions of employment, and professional advancement. Notices of availability are publicly listed on the Woolf platform and, when available, other sites visited by academics. Criteria for teaching positions, including any associated conditions of ongoing employment, are clearly stated. Applications for teaching are reviewed by the Administrative Board, or a committee of the Board, until a position(s) is filled. Notices state the supporting documentation required as evidence for the review of an applicant. All applicants are required to demonstrate their competence for the teaching position by providing a copy of their credentials to be verified

before the position is filled. This policy applies to all teaching roles of Woolf Education Ltd, including any teaching services provided by third party vendors, which are subject to the same process of review. In all cases, the final decision for filling a role in accordance with the criteria stated on the public notice is made by the Administrative Board.

Woolf's policies and procedures apply consistently to full-time, part-time, *ad hoc*, and third-party teaching activities. All teaching activities fall within the scope of Woolf policy. Teaching staff, including part-time or *ad hoc* teaching staff are directed towards updates and developments in their field as well as the methodological requirements for their programs.

All Faculty Members are encouraged to discuss innovative forms of teaching, formulate how these may be implemented, and propose those implementations in the Faculty Council. At the end of all courses, students provide feedback on their learning experience, and twice per year faculty provide feedback by survey. All teachers are expected to maintain a record of student outcomes, and teaching activities are periodically reviewed or observed. In cases of disagreement, or suspected misconduct, fraud, or prejudice, a Red Flag should be submitted under the Red Flag Procedure.

All courses and programs are subject to processes of quality enhancement to improve student outcomes, including the course's continued review to assess its scholastic rigor and value.

Faculty Requirements

Academic Staff are called Faculty members at Woolf. Faculty members at Woolf must possess a research doctorate and are expected to have a record of peer-reviewed research. All teaching is under the authority and oversight of a faculty member – including instructional design, synchronous meetings, and lectures. In cases where pre-recorded lectures or podcasts are provided that contain content from outside of Woolf, any such content is to be produced by lecturers who are experts with a research doctorate in the relevant domain, or where relevant, by those with at least 7 years of industry-specific experience.

Teaching Staff are called Expert Instructors, Domain Experts, or simply Instructors. Expert instructors are used in courses to provide domain-specific industry insights, including insights and feedback on student work during synchronous meeting sessions. Expert instructors must have management experience and be in possession of at least a master's level qualification. Expert instructors are under the direct authority of the Faculty Members and must be trained in Woolf's pedagogical methods.

Faculty Advisors

Colleges at Woolf exist to support their members and provide helpful resources to students.

In the tradition of Harvard's "Houses" and Oxford's "Colleges", Woolf provides every student with membership in a Woolf college for support during their academic journey. Every student should be assigned a Faculty Advisor, who is a faculty member from within the student's own college, and who acts as the first point of contact for non-technical academic issues related to the student's progress, particularly where these may benefit from an independent point of view. Students are strongly encouraged to meet with their advisors at least once each semester.

Thus every faculty member oversees their own registered students through the normal synchronous teaching sessions, and office hours, and additionally provides availability that can be booked for advisees, should the need arise.

Program Outline

Outline of Program		
Course Title	**Credits**	**Course Description**
1. Data Structures	2.5	This course is aimed to build a strong foundational knowledge of data structures (DS) used extensively in computing.

2. Design and Analysis of Algorithms	2.5	This is a foundational and mandatory course which aims to build student's ability to apply various algorithmic design methods to provide an optimal solution to computational problems.
3. Relational Databases	2.5	This is a core and foundational course which aims to equip the student with the ability to model, design, implement and query relational database systems for real-world data storage & processing needs.
4. Numerical Programming in Python	2.5	This course helps students translate mathematical/statistical/scientific concepts into code. This is a foundational course for writing code to solve Data Science ML & AI problems.

5. Applied Statistics	2.5	This course introduces basic probability theory , statistical methods and computational algorithms to perform mathematically rigorous data analysis.
6. Introduction to Machine Learning	2.5	This course focuses on building basic classification and regression models and understanding these models rigorously both with a mathematical and an applicative focus.
7. High Dimensional Data Analysis	2.5	This course is aimed to help learners understand various techniques and algorithms to visualize, analyze and understand high dimensional data which is very common in Data Science and ML.

8. Advanced Machine Learning	2.5	This course introduces more advanced ML techniques like ensembles: bagging, boosting, cascading and stacking classifiers and regressors.
9. Distributed Machine Learning	2.5	This course provides an in-depth understanding of distributed systems for ML and Deep Learning using CPU,GPU and TPU clusters.
10. Introduction to Deep Learning	2.5	This course provides a strong mathematical and applicative introduction to Deep Learning.

11. Deep Learning for Computer Vision	2.5	This course provides a comprehensive overview of Computer vision problems and how they can be tackled using various Convolutional Neural networks (CNNs).
12. Deep Learning for Natural Language Processing (NLP)	2.5	This course focuses on modeling sequences (text, music, time-series, genes) using deep-learning models.
13. Productionisation of Machine Learning Systems	2.5	This course aims to build the core competency of building real world end-to-end ML systems and deploy them into production for a variety of problems and scenarios.

14. System Design	2.5	This course is aimed at equipping students with skills to architect the high level design (a.k.a. system design) of software and data systems.
15. DevOps	2.5	This course provides students with hands-on experience on deploying high velocity applications and services reliably on complex and distributed infrastructure.
16. Front End UI/UX Development	2.5	This is a hands-on course on designing responsive, modern and light-weight UI for web, mobile and desktop applications using HTML5, CSS and Frameworks like Bootstrap 4.

17. JavaScript	2.5	This course is a hands-on course covering JavaScript from basics to advanced concepts in detail using multiple examples.
18. Front End Development	2.5	This course builds upon the introductory JavaScript course to acquaint students of popular and modern frameworks to build the front end.
19. Back End Development	2.5	This is a foundational course on building server-side (or backend) applications using popular JavaScript runtime environments like Node.js.

20. Foundations of Cloud Computing	2.5	This is a course that focuses both on architectural design and practical hands-on learning of the most used cloud services.
21. Advanced Back End Development	2.5	This course provides a dive deep into more advanced concepts in server-side programming using Node.js to enable initiative, real-time and scalable web applications.
Distributed Cloud Computing	2.5	This course provides an in-depth architectural overview and hands- on experience with building scalable data processing and distributed computing via various cloud systems.

23. Advanced Cloud Computing	2.5	This is a course that focuses both on architectural design and practical hands-on learning of advanced cloud-based services.
24. NoSQL Cloud Datastores	2.5	This course provides a comprehensive overview and practical knowledge of various NoSQL data stores and how they can be used on the Cloud (AWS).
25. Design Patterns	2.5	This course provides a practical understanding of popular object-oriented design patterns so that students can reuse design strategies developed for commonly occurring problems in software development.

26. Advanced Applied Computer Science	15	Advanced Applied Computer Science is a capstone project, an end-to-end deployable solution to a real-world computational problem that students build in the last phase of the program.
27. Introduction to Computer Programming: Part 1	2.5	This course helps students translate advanced mathematical/statistical/scientific concepts into code. This is a course for writing code to solve real-world problems.
28. Introduction to Problem-Solving Techniques: Part 1	2.5	Building a toolbox of problem-solving strategies will improve problem solving skills. With practice, students will be able to recognize and choose among multiple strategies to find the most appropriate one to solve complex problems. The course will focus on developing problem-solving strategies such as abstraction, modularity, recursion, iteration, bisection, and exhaustive enumeration.

29. Introduction to Problem-Solving Techniques: Part 2	2.5	This course is a follow-up to Introduction to Problem-Solving Techniques: Part 1, and as part of their academic planning process with Woolf staff, students will ordinarily take that course first.
30. Mathematics for Computer Science	2.5	This course covers discrete mathematics for computer science and engineering. Topics may include asymptotic notation and growth of functions; permutations and combinations; counting principles; discrete probability.
31. Advanced Algorithms	2.5	This course covers general approaches to the construction of efficient solutions to problems.

32. Computer Systems and Their Fundamentals	2.5	This core course equips the student with knowledge of database management systems, operating systems and computer networks.
33. Low-Level Design and Design Patterns	2.5	Low-Level Design & Design Patterns focuses on modularity and reusability in software design, common design vocabularies, refactoring and how to reduce it, and how to incorporate design patterns into iterative development processes. The course pays significant attention to the interaction between system architecture and components, including data organization.
34. Practical Software Engineering	2.5	This course gives the detailed overview on how to approach Low Level Design problems with real-world case studies discussed such as Designing a Pen (Mac/Windows), TicTacToe, BookMyShow (most used event booking app, manages millions of users), Email campaign Management System and detailed design of Splitwise.

35. Distributed Systems with High-Level System Design	2.5	In this course, students will learn to execute a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks.
36. Data Visualization Tools	2.5	This course is aimed to build a strong foundational knowledge of Data Analytics tools used extensively in the Data Science field.
37. Power BI for Data Analysis and Exploration	2.5	Students will learn how to handle data sources in Power BI, connecting to various data sources using Power BI, query editors, managing data relationships, and cross filter direction.

38. Statistical Programming	2.5	This course focuses on representing statistical techniques in code, and may be conducted in Python, R, or another relevant language.
39. Spreadsheets for Data Understanding	2.5	Spreadsheets for Data Understanding introduces students to the principles and techniques of data cleaning, handling data sets of varying sizes, and visualizing data/data storytelling.
40. Advanced Python Programming	2.5	Advanced Python Programming builds on introductory programming courses to illustrate object-oriented programming concepts, database design in Python, and the basics of Machine Learning with Python libraries.

41. Foundations of Machine Learning	2.5	This course focuses on building basic classification and regression models and understanding these models rigorously both with a mathematical and an applicative focus.
42. Business Case Studies	2.5	A business case study is a course designed for the learner to identify a business real world problem and its objective is to help students rigorously solve a real-world, technically-challenging business problem where they would apply all of the concepts, techniques and tools learnt in the program. Students typically pick a problem from a known business problem or identify business cases where data analytics can be used to solve a problem.
43. Studies in Data Science and Data Analytics	2.5	This advanced graduate class addresses a unique topic on a rotating basis in order to keep the program at the forefront of scholarly research and industry practice. Every year the academic staff member will approve of a new topic to be covered. The bibliography will contain not less than 8 peer-reviewed articles or scholarly publications reflecting the current topic.

44. Further Studies in Data Science and Data Analytics	2.5	This advanced graduate class addresses a unique topic on a rotating basis in order to keep the program at the forefront of scholarly research and industry practice. Every year the academic staff member will approve of a new topic to be covered. The bibliography will contain not less than 8 peer-reviewed articles or scholarly publications reflecting the current topic.
45. SQL for Data Analytics	2.5	Structured Query Language (SQL) is key to working with data in relational databases, a task at the core of data science and analytics. In this course, students will learn all the major keywords and clauses used to extract data, best practices for formatting SQL queries, and how to generate meaningful insights from the results.
46. Product Analytics	2.5	This course teaches students how to analyze the ways users engage with a service. This method, called product analytics, helps businesses track and analyze user data. Students will learn more deeply what is required to move a product from idea to implementation, through to launch, and then on to iterative improvements.

47. Data Engineering	2.5	Students will learn a comprehensive view of the complete Data Engineering lifecycle.
48. Product Management for Software Engineers	2.5	In this course, students will get a fundamental understanding of product management practices.
49. Applied Computer Science Project	5	This is a project-based course, with the aim of building the required skills for creating web-based software systems. The course covers the entire lifecycle of building software projects, from requirement gathering and scope definition from a product document, to designing the architecture of the system, and all the way to delivery and maintenance of the software system.

50. Introduction to Computer Programming: Part 2	2.5	This course provides a practical and detailed understanding of popular programming paradigms and data storage types. Students learning this will be able to write and solve programming problems.

Assessment and Grading

General Procedures

Academic assessment at Woolf is of two kinds: regular and cumulative. Regular assessment applies to the continuous evaluation of student progress, concentrating on the proficiency of submitted assignments, and the ability of the student to respond to issues raised by the instructor during an instructional session. Cumulative assessment applies to the final project assignment. This requires the students to deepen and extend the scholarly engagements initiated in their prior work.

Students who fail any one course of the degree, cannot progress to complete the degree, except by approval of the College Dean or College Academic Committee. Failed courses may be retaken at the approval of the College Dean or College Academic Committee, or by appeal, at the discretion of the Quality Assurance, Enhancement, and Technology Alignment Committee (QAETAC). For more information about QAETAC please see the Woolf Academic Handbook.

Cumulative Examination of Courses

Traditionally, cumulative examination of courses is by a submitted final project in the form of a long assignment.

The long assignment is meant to synthesize, deepen, and extend the learning outcomes of the regular lessons while introducing new material and insights. Not more than 50% of the long assignment may be material taken from other assignments. The topic of the assignment must be agreed in advance with the instructor. Examination assignments are expected to be completed at a high research standard, and must be well-structured, well-crafted, and contain appropriate citations to the primary and secondary literature of the course.

Mode of Teaching and Assessment

The Online Campus

Woolf University's courses are offered almost exclusively through its proprietary learning platform. The platform supports both asynchronous and synchronous modes of learning. The asynchronous portion of programs includes structured course materials that the course lead and course instructors prepare ahead of time. These courses are done independently of students' classmates and according to the student's own schedule, but prior to the synchronous sessions. Synchronous sessions are held in the "virtual classroom," where students and faculty use internet technology such as video conferencing and web cameras to ensure they are actively engaged in the learning process. It is essential for students to connect with each other, share information, and create professional networks and relationships as they would in a traditional program or within a professional setting. For information about the technical specific requirements please see the Technical Requirements section within the Woolf Academic Handbook.

Structure of the Courses

The MS in CS combines asynchronous components (lecture videos, readings, and written assignments) and synchronous meetings attended by students and an instructor or faculty member during a video call.

Asynchronous components support the schedules of students from diverse work-life situations, and synchronous meetings provide accountability and motivation for students.

The program is composed of multiple short foundation courses, extended specialist courses, and a capstone project – the Digital Action Program for Business Administration.

Each short foundation course, and the extended specialist courses, consists of regular lessons and cumulative lessons devoted to summative examination. Each short foundation course requires 75 hours of learner time to complete, and each extended specialist course requires 250 hours to complete.

As is typical for synchronous teaching, the student will compose one assignment (such as a report, 1,000 word essay, financial model, or presentation) per lesson, which is the topic of meeting discussion. A full-time student completes two lessons per week with an assignment submitted for each lesson; this pattern continues for each regular lesson in the course.

Summative examination lessons allow an appropriate amount of time for the student to review and revise his or her prior work and deepen their synthetic grasp of the materials in preparation for cumulative examination or project.

Contact Hours

For the breakdown of hours, consult the Pedagogical Procedures and Assessment sections.

Students engage in 8 synchronous meeting sessions in which questions and answers about asynchronous study materials are addressed.

Synchronous meeting engagements include not only 60-75 minutes of intensive contact time between the students and faculty member in every lesson (up to twice per week), but also extends beyond the synchronous meeting session itself to include ongoing supervisory support on an ad hoc basis (typically by email or brief virtual meeting). Students are closely supervised by asynchronous direction, oversight, feedback, and guidance; and occasionally new handouts or other scholarly materials are provided to follow up on issues raised in a synchronous meeting discussion. Thus, on our calculation, we allocate four further hours of contact time beyond the synchronous session to capture the individual, personalized, intensive, bespoke form of guidance that a student receives. We calculate all other forms of contact time at a 1:1 ratio, including lectures, whether delivered synchronously or as pre-recorded videos or podcasts.

Example regular lesson Breakdown

* = contact hours = 66

† = assessment hours = 1.25

Hours	**Activity**
1.25*†	synchronous session
4*	Time under the direction and control of a tutor
1.5*	Lecture videos or podcasts
8.25	Independent reading & note-taking
7.5	Assignment composition
22.5	**Total**

Pedagogical Procedures

The MS in CS will be delivered using online and blended learning techniques, which support a variety of teaching and learning methods, including the following:

- synchronous meetings;
- assigned lectures by video or podcast;
- assigned readings;
- handouts delivered electronically;
- digital material, including slideshow presentations and other assigned media provided in course packets and by weblink.

The core pedagogical method used in this course will include synchronous meetings between a faculty member, or a subject expert instructor under the oversight of a faculty

member, and a small group of students. Student interaction plays a key role in the organization of each synchronous meeting, which focuses on a discussion of a student's pre-submitted assignment.

Online delivery, although identical in content for an in-person session, provides significant advantages to students in terms of accessibility – students can more easily reach academic experts across borders and more easily integrate study within the pattern of their own life or career. Further advantages of the online delivery include the digital quality assurance techniques outlined within the Academic Handbook.

Preparing for a single synchronous meeting requires about 21.25 hours. A representative workload consists of the following: students must review about 100 pages of reading material (or equivalent video content, audio content, or interactive content) and prepare a piece of written work of 2-4 pages in response to a specific set assignment question. Before the start of the synchronous session, this work is submitted to the instructor for review.

Every student must then be prepared to discuss and defend his or her written work directly with the instructor (who is an expert in the field) and the other students for up to 75 continuous minutes. Faculty members and subject experts seek to deepen the students' understanding of the material and probe aspects of the written assignment that may benefit from clarification, revision, or further exploration.

At the end of the synchronous session, the instructor provides the student with feedback on the meeting assignment, including a mark, and provides bespoke guidance for the next assignment. Students are provided with a curated reading list from the instructors, assigned pre-recorded lectures, and a research question to guide the organization of their next synchronous meeting assignment or assignment. Engaging in this activity twice per week is a full workload of about 45 hours.

The synchronous meeting system is designed to be mentally demanding and personally engaging. The pedagogical style is known for producing high-quality domain-specific learning outcomes because students must learn assigned materials and related case studies for themselves, before presenting their work to an instructor in their own words for discussion twice per week. By requiring students to describe and analyze topics in their own words, synchronous meetings engage and extend a student's existing range of abilities.

The synchronous method is also known for producing high-quality domain-agnostic learning outcomes because students must be prepared to organize and present their perspective on an assignment twice per week, and be prepared to think analytically and creatively about what they have done. Students must learn to present their viewpoint, even while being prepared to adopt a new position in light of the evidence and under the questioning of the teacher.

Assessment

General Procedures

For the MS in CS, assessment is of two kinds: regular assessment and summative assessment.

Regular assessment applies to the continuous evaluation of student progress, concentrating on the proficiency of submitted assignments, and the ability of the student to respond to issues raised by the instructor during a synchronous meeting session.

Cumulative assessment applies to the final project assignment. This requires the students to deepen and extend the scholarly engagements initiated in their prior work.

Students who fail any one course of the degree, cannot progress to complete the degree, and will by default fail the MS in CS. Failed courses may be re-taken at the discretion of Woolf's Faculty Members.

Cumulative Examination of courses

For each course, a percentage of the grade derives from the average of the regular assignments, and a larger percentage of the grade derives from the cumulative examination. The cumulative examination of courses is by a submitted final project in the form of a long assignment.

The cumulative examination/long assignment is by summative assignment (3,000 word essay, or similarly-sized financial model, or presentation), which must synthesize, deepen, and extend the learning outcomes of the regular units while introducing new material and insights. Not more than 50% of the long assignment may be material taken from synchronous meeting assignments. The topic of the assignment must be agreed in advance with the faculty member and subject matter experts. Examination assignments are expected to be completed at a high research standard, and must be well-structured, well-crafted, and contain appropriate citations to the primary and secondary literature of the course.

Grading Progress

Students receive grades and feedback on each assignment throughout a course. Depending on the specific course, students may receive both a grade and a written or audio comment from an instructor. Courses display the weight assigned to each grade-bearing category (such as regular assignments, attendance, and/or final projects). Students have access to their grade book at all times and can see a record of all grades, attendance records, and assignment submissions.

The grade book displays the current running average for the grades in the course in accordance to the grade weights, inclusive of all those assignments which the student has submitted and on which the instructor has provided grades. At the end of the course, grades are finalized and added to the student's transcript, which is accessible at all times for students enrolled in credit-bearing programs.

Grading System

The final grade for a course is determined by the weighting rules stated in the course offering. Unless otherwise stated, all courses are weighted as follows: 30% of the grade derives from the average of the instructional assignment session, and 70% of the grade derives from the cumulative examination.

The final grade on a degree is weighted in proportion to the credits of individual courses. For example, a degree composed of a 3 credit course and a 6 credit course will weigh the 6 credit courses proportionately more, according to the number of credits.

Woolf's International Grade Classification

Woolf's teachers are trained in a number of different grading scales; these scales are cross-referenced. This handbook employs the American grading system and classification,

with US grades as the default.[1] US grades are the most granular and distributed with the least number of gaps, which is why we have selected them as a default marking scheme for transcripts. Woolf's international conversion scheme is as follows:

US GPA	US Grade	US Per Cent	UK Mark	UK Classification	Malta Grade	Malta Mark	Malta Classification
4	A+	97 - 100	70+	First class honors	A	80-100%	First class honors
3.8-4.0	A	94-96	67-69	Upper-second class honors	B	70-79%	Upper-second class honors
3.7	A-	90-93	65-67	Upper-second class honors			
3.3	B+	87-89	60-64	Lower-second class honors	C	55-69%	Lower-second class honors
3	B	84-86					
2.7	B-	80-83	55-59	Lower-second class honors			
2.3	C+	77-79	50-54	Third class honors	D	50-54%	Third-class honors
2	C	74–76					
1.7	C-	70–73	45-49	Third class honors			
1.3	D+	67–69	40-44	Ordinary/Unclassified			
1	D	64–66	35-39	Ordinary/Unclassified			
0.7	D-	60–63					
0	F	Below 60	Below 35		F	45-54%	

Woolf Grading Criteria, Definition of Grades, and Classification

Grading of student work keeps in view the scale of work that the student can reasonably be expected to have undertaken in order to complete the task. The Woolf grading scheme draws heavily from the marking scheme set out by the University of Oxford (cf. History Faculty Course Handbook 2016-2018).

a. The assessment of work for the course is defined according to the following rubric of general criteria:

i. i. Engagement:

- Directness of engagement with the question or task
- Range of issues addressed or problems solved

[1] Cf. the Fulbright Commission (http://www.fulbright.org.uk/going-to-the-usa/pre-departure/academics), Princeton Review (https://www.princetonreview.com/college-advice/gpa-college-admissions), European Commission (https://eacea.ec.europa.eu/national-policies/eurydice/content/second-cycle-programmes-49_en), and University of Malta (https://www.um.edu.mt/__data/assets/pdf_file/0005/47390/harmonisedregs-09.pdf).

- Depth, complexity, and sophistication of comprehension of issues and implications of the question or task
- Effective and appropriate use of imagination and intellectual curiosity

ii. Argument or solution:
- Coherence, mastery, control, and independence of work
- Conceptual and analytical precision
- Flexibility, e.g. discussion of a variety of views, ability to navigate through challenges in creative ways

iii. Evidence (as relevant):
- Depth, precision, detail, range and relevance of evidence cited
- Accuracy of facts
- Knowledge of first principles and demonstrated ability reason from them
- Understanding of theoretical principles and/or historical debate
- Critical engagement with primary and/or secondary sources

iv. Organization and presentation:
- Clarity and coherence of structure
- Clarity and fluency of writing, code, prose, or presentation (as relevant)
- Correctness of conformity to conventions (code, grammar, spelling, punctuation or similar relevant conventions)

b. US grades for courses are defined according to the following rubric:
97-100

Work will be so outstanding that it could not be better within the scope of the assignment. These grades will be used for work that shows exceptional excellence in the relevant domain; including (as relevant to the domain): remarkable sophistication and mastery, originality or creativity, persuasive and well-grounded new ideas or methods, or making unexpected connections or solutions to problems.

94-96

Work will excel against each of the General Criteria. In at least one area, the work will be merely highly competent.

90-93

Work will excel in more than one area, and be at least highly competent in other respects. It must be excellent and contain: a combination of sophisticated engagement with the issues; analytical precision and independence of solution; go beyond paraphrasing or boilerplate code techniques; demonstrating quality of awareness and analysis of both first principles or primary evidence and scholarly debate or practical tradeoffs; and clarity and coherence of presentation. Truly outstanding work measured against some of these criteria may compensate for mere high competence against others.

87-89

Work will be at least very highly competent across the board, and excel in at least one group of the General Criteria. Relative weaknesses in some areas may be compensated by conspicuous strengths in others.

84-86

Work will demonstrate considerable competence across the General Criteria. They must exhibit some essential features addressing the issue directly and relevantly across a good range of aspects; offer a coherent solution or argument involving (where relevant) consideration of alternative approaches; be substantiated with accurate use of resources (including if relevant, primary evidence) and contextualization in debate (if relevant); and be clearly presented. Nevertheless, additional strengths (for instance, the range of problems addressed, the sophistication of the arguments or solutions, or the use of first principles) may compensate for other weaknesses.

80-83

Work will be competent and should manifest the essential features described above, in that they must offer direct, coherent, substantiated and clear arguments; but they will do so with less range, depth, precision and perhaps clarity. Again, qualities of a higher order may compensate for some weaknesses.

77-79

Work will show solid competence in solving problems or providing analysis. But it will be marred by weakness under one or more criteria: failure to fully solve the problem or discuss the question directly; some irrelevant use of technologies or citing of information; factual error, or error in selection of technologies; narrowness in the scope of solution or range of issues addressed or evidence adduced; shortage of detailed evidence or engagement with the problem; poor organization or presentation, including incorrect conformity to convention or written formatting. They may be characterized by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument or solution.

74-76

Work will show evidence of some competence in solving problems or providing analysis. It will also be clearly marred by weakness in multiple General Criteria, including: failure to solve the problem or discuss the question directly; irrelevant use of technologies or citing of information; factual errors or multiple errors in selection of technologies; narrowness in the scope of solution or range of issues addressed or evidence adduced; shortage of detailed evidence or engagement with the problem; poor organization or presentation, including incorrect conformity to convention or written formatting. They may be characterized by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument or solution.

70-73

Work will show evidence of competence in solving problems or providing analysis, but this evidence will be limited. It will be clearly marred by weakness in multiple General Criteria. It will still make substantive progress in addressing the primary task or question, but the work will lack a full solution or directly address the task; the work will contain irrelevant material; the work will show multiple errors of fact or judgment; and the work may fail to conform to conventions.

67-69

Work will fall down on a number of criteria, but will exhibit some of the qualities required, such as the ability to grasp the purpose of the assignment, to deploy substantive information or solutions in an effort to complete the assignment; or to offer some coherent analysis or work towards the assignment. Such qualities will not be displayed at a high level,

and may be marred by irrelevance, incoherence, error and poor organization and presentation.

64-66

Work will fall down on multiple General Criteria, but will exhibit some vestiges of the qualities required, such as the ability to see the point of the question, to deploy information, or to offer some coherent work. Such qualities will be substantially marred by irrelevance, incoherence, error and poor organization and presentation.

60-63

Work will display a modicum of knowledge or understanding of some points, but will display almost none of the higher qualities described in the criteria. They will be marred by high levels of factual or technology error and irrelevance, generalization or boilerplate code and lack of information, and poor organization and presentation.

0-60

Work will fail to exhibit any of the required qualities. Candidates who fail to observe rubrics and rules beyond what the grading schemes allow for may also be failed.

c. Synchronous Meeting Discussions and *Viva Voce* Examination Template

Synchronous meeting discussions and *viva voce* examinations are conducted on the same format: written work is submitted in advance, and a discussion follows. This provides students an opportunity to clarify and explain their written claims, and it also tests whether the work is a product of the student's own research or has been plagiarized.

For the *viva voce* examination, the submitted work is graded, and the grade is recorded prior to the oral examination.

The synchronous discussion and *viva voce* examination acts to shift the recorded grade on the submitted essay according to the following rubric:

+3

Up to three points are added for excellent performance; the student displays a high degree of competence across the range of questions, and excels in at least one group of criteria. Relative weaknesses in some areas may be compensated by conspicuous strengths in others.

+/- 0

The marked script is unchanged for fair performance. Answers to questions must show evidence of some solid competence in expounding evidence and analysis. But they will be marred by some weakness under one or more criteria: failure to discuss the question directly; appeal to irrelevant information; factual error; narrowness in the range of issues addressed or evidence adduced; shortage of detailed evidence; or poor organization and presentation, including consistently incorrect grammar. Answers may be characterized by unsubstantiated assertion rather than argument, or by unresolved contradictions in the argument.

-3 (up to three points)

Up to three are subtracted points for an inability to answer multiple basic questions about themes in the written work. Answers to questions will fall down on a number of criteria, but will exhibit some vestiges of the qualities required, such as the ability to see the point of the question, to deploy information, or to offer some coherent analysis towards an argument.

Such qualities will not be displayed at a high level or consistently, and will be marred by irrelevance, incoherence, error and poor organization and presentation.

0

Written work and the oral examination will both be failed if the oral examination clearly demonstrates that the work was plagiarized. The student is unfamiliar with the arguments of the essay or the sources used for those arguments.

Curriculum

Advanced Algorithms

Course Description

This course covers general approaches to the construction of efficient solutions to problems.

Such methods are of interest because:

1. They provide templates suited to solving a broad range of diverse problems.
2. They can be translated into common control and data structures provided by most high-level languages.
3. The temporal and spatial requirements of the algorithms which result can be precisely analyzed.

This course will provide a solid foundation and background to design and analysis of algorithms. In particular, upon successful completion of this course, students will be able to understand, explain and apply key algorithmic concepts and principles, which might include:

1. Greedy algorithms (Activity Selection, 0-1 Knapsack Problem, Fractional Knapsack Problem)
2. Dynamic programming (Longest Common Subsequence, 0-1 Knapsack Problem)
3. Minimum Spanning Trees (Prim's Algorithm, Kruskal's Algorithm)
4. Graph Algorithms (Dijkstra's Shortest Path Algorithm, Bipartite Graphs, Minimum Vertex Cover)

Although more than one technique may be applicable to a specific problem, it is often the case that an algorithm constructed by one approach is clearly superior to equivalent solutions built using alternative techniques. This course will help students assess these choices.

Advanced Applied Computer Science

Course Description

Advanced Applied Computer Science is a capstone project, an end-to-end deployable solution to a real-world computational problem that students build in the last

phase of the programme. Its objective is to help students rigorously solve a technically challenging problem where they would apply all of the concepts, techniques and tools learned in the programme. Students typically pick a problem from their specialization after discussing it with the course instructor(s). Students also have the option of working on a real-world problem in their company/organization/institution. They can be mentored by an expert supervisor from their organization along with an academic supervisor from Woolf. All external expert-supervisors and projects need to be approved by the instructor(s) to ensure that the project is technically challenging and the solution being built is rigorous and of high quality. Students start with identifying a technically challenging problem. Once approved by the instructor(s), they start the literature survey to read research papers and technical reports of prior related work. Then, they build the system design and write a design document to solve the problem. This would be followed by designing and implementing individual courses and testing them. This would be followed by deploying the solution and making it available to end users while satisfying the problem's real-world constraints and objectives. Students then document their work into a detailed technical report.

Advanced Back End Development

Course Description

This course provides a dive deep into more advanced concepts in server-side programming using Node.js to enable initiative, real-time and scalable web applications. We dive into threading and thread pools in Node.js and how they can be leveraged to build more responsive web apps. We learn socket programming using socket.io and Node.js for instant messaging, document collaboration, real time analytics and streaming applications. Students also learn to use Caching using distributed in-memory key-value stores (like Redis) to rescue latency while serving web-apps. Students also learn how to use Node.js with popular NoSQL data stores like MongoDB for storing unstructured data. We also cover GraphQL which is an open source data query and manipulation language for APIs, which is gaining popularity more recently. We learn popular protocols like OAuth to enable cross platform logins. Students also learn the architecture and practical aspects of Web-RTC to enable multimedia applications like video-chat, live-streaming, music-streaming etc.

Advanced Cloud Computing

Course Description

This is a course that focuses both on architectural design and practical hands-on learning of advanced cloud-based services. We begin with the serverless computing model and how it is achieved by most cloud providers. We learn to use it for building web applications, data and file processing and analytics applications. We then learn of the architecture of distributed messaging queues and how they can be used for plumbing complex cloud systems with many components and services. Monitoring the resources in your cloud setup is a key to ensure low costs and high availability and the smooth functioning of your overall setup. We learn to use AWS CloudWatch to track various key metrics, trigger alarms, detect anomalous behavior and act upon them in near real-time. We

learn the architecture and design of load balancers and how they play a key role in most horizontally scalable web-applications. Students also learn of the architecture and design of Content delivery networks (CDNs) from Akamai and Amazon. We learn how CDNs can be used to deliver live streaming and website content fast using globally distributed servers and caching. Most of this course involves learning the internal architecture of various cloud systems and using them to solve real world engineering problems.

Advanced Machine Learning

Course Description

This course introduces more advanced ML techniques like ensembles: bagging, boosting, cascading and stacking classifiers and regressors. It covers both the theoretical foundations and applicative details of these techniques along with popular implementations of boosting like LightGBM, CatBoost and XGBoost. Students also delve into kernel methods with specific focus on SVMs for classification and regression. Students will study state of the art model agnostic feature importance and model-interpretability techniques like LIME and SHAP. Students also study classical NLP based text encoding methods like Bag-of-words, TF-IDF etc. The course teaches various classical methods in time series analysis and forecasting like ARMA, ARIMA etc. Students also learn how to pose time series forecasting problems as regression and classification problems to leverage well studied ML techniques. This is followed by various domain and problem specific Feature engineering techniques that are often helpful in real world problem solving. Students will study methods like error analysis, ablative analysis etc., to debug and understand why and where a model is performing well and where it is not performing well. This will further help us in designing appropriate features. Students study model calibration techniques like Platt Scaling, Isotonic Regression etc. Later in this course, we cover how to build recommender systems using content-based and collaborative filtering methods. The course also teaches the detailed solution of the Netflix prize (2009) and various recent advances in RecSys.

Advanced Python Programming

Course Description

Advanced Python Programming builds on introductory programming courses to illustrate object-oriented programming concepts, database design in Python, and the basics of Machine Learning with Python libraries. Students will learn how to solve problems in Python, develop design patterns in Python code, develop internet applications with Python, and collaborate with other students to implement projects. The course introduces advanced features such as decorators and generators, as well as a thorough exploration of the Python development environment.

This course is designed to prepare students for an entry-level developer position.

Applied Computer Science Project

Course Description

This is a project-based course, with the aim of building the required skills for creating web-based software systems. The course covers the entire lifecycle of building software projects, from requirement gathering and scope definition from a product document, to designing the architecture of the system, and all the way to delivery and maintenance of the software system.

The course covers both frontend, which is, building browser-based interfaces for users, using frontend web frameworks, and also building the backend, which is the server running an API to serve the information to the frontend, and running on an SQL or similar database management system for storage.

All aspects of delivering a software project, including security, user authentication and authorisation, monitoring and analytics, and maintaining the project are covered. The course also covers the aspects of project maintenance, like using a version control system, setting up continuous integration and deployment pipelines and bug trackers.

Applied Statistics

Course Description

This course introduces basic probability theory , statistical methods and computational algorithms to perform mathematically rigorous data analysis. The course starts with basic foundational concepts of random variables, histograms, and various plots (PMF, PDF and CDF). Students learn various popular discrete and continuous distributions like Bernoulli, Binomial, Poisson, Gaussian, Exponential, Pareto, log-normal etc., both mathematically and from an applicative perspective. Students learn various measures like mean, median, percentiles, quantiles, variance and interquartile-range. Students learn the pros and cons of each metric and understand when and how to use them in practice. Students will learn conditional probability and Bayes theorem in the applied context of real-world problems in medicine and healthcare. The course teaches the foundations of non-parametric statistics and applies them to solve problems using computational tools. Students learn various methods to determine correlations rigorously in data. This is followed by applied and mathematical understanding of the statistics underlying control-treatment (A/B) experiments and hypothesis testing. The course engages computation tools in modern statics like Bootstrapping, Monte-Carlo methods, RANSAC etc.

Back End Development

Course Description

This is a foundational course on building server-side (or backend) applications using popular JavaScript runtime environments like Node.js. Students will learn event driven programming for building scalable backend for web applications. The course teaches various

aspects of Node.js like setup, package manager, client-server programming and connecting to various databases and REST APIs. Most of these concepts would be covered in a hands-on manner with real world examples and applications built from scratch using Node.js on Linux servers. This course also provides an introduction to Linux server administration and scripting with special focus on web-development and networking. Students learn to use Linux monitoring tools (like Monit) to track the health of the servers. The course also provides an introduction to Express.js which is a popular light-weight framework for Node.js applications. Given the practical nature of this course, this would involve building actual website backends via assignments/projects for ecommerce, online learning and/or photo-sharing.

Business Case Studies

Course Description

A business case study is a course designed for the learner to identify a business real world problem and its objective is to help students rigorously solve a real-world, technically-challenging business problem where they would apply all of the concepts, techniques and tools learnt in the program. Students typically pick a problem from a known business problem or identify business cases where data analytics can be used to solve a problem. The choosing of a topic can be done after discussing it with the course instructor(s). Students also have an option of choosing a business problem in their professional organization but the external supervisors should be approved by the instructor(s). Students start by identifying a business problem and proposing a methodology to solve the said business problem. Students then decide what technical and business tools will be used for the solution methodology. Students will first work on the real-world data, clean and process it using techniques learnt in this program. Students then will use algorithms and approach with a coding language and tool they think will get the best results. At the end of the case study student should be able to present the business problem and solution either via Jupyter notebooks or via a blog.

Computer Systems and Their Fundamentals

Course Description

This core course equips the student with knowledge of database management systems, operating systems and computer networks. At the end of the course, students will have a critical understanding of the architecture of computers and networks, as well as how programs interact with these. Students begin with mapping data storage problems (as they had done in Relational Databases) to understand how data is stored in a distributed network, and related issues such as concurrency. Subsequently, students cover operating systems with an overview of process scheduling, process synchronization and memory management techniques with disk scheduling. The course concludes with computer networks, where we will be discussing all of the computer network layers and their protocols in detail.

Data Engineering

Course Description

Data is the fuel driving all major organizations. This course helps you understand how to process data at scale.

From understanding the fundamentals of distributed processing to designing data warehousing and writing ETL (Extract Transform Load) pipelines to process batch and streaming data.

Students will learn a comprehensive view of the complete Data Engineering lifecycle.

Data Structures

Course Description

This course is aimed to build a strong foundational knowledge of data structures (DS) used extensively in computing. The course starts with introducing time and space complexity notations and estimation for code snippets. This helps students be able to make trade-offs between various Data Structures while solving real world computational problems. The course introduces most widely used basic data structures like Dynamic arrays, multi-dimensional arrays, Lists, Strings, Hash Tables, Binary Trees, Balanced Binary Trees, Priority Queues and Graphs. The course discusses multiple implementation variations for each of the above data-structures along with trade-offs in space and time for each implementation. In this course, students implement these data-structures from scratch to gain a solid understanding of their inner workings. Students are also introduced to how to use the built-in data-structures available in various programming languages/libraries like Python/NumPy/C++ STL/Java/JavaScript. Students solve real-world problems where they must use an optimal DS to solve a computational problem at hand.

Data Visualization Tools

Course Description

This course is aimed to build a strong foundational knowledge of Data Analytics tools used extensively in the Data Science field. There are now powerful data visualization tools used in the business analytics industry to process and visualize raw business data in a very presentable and understandable format. A good example is Tableau, used by all data analytics departments of companies and in data analytics companies in various fields for its ease of use and efficiency. Tableau uses relational databases, Online Analytical Processing Cubes, Spreadsheets, cloud databases to generate graphical type visualizations. Course starts with visualizations and moves to an in-depth look at the different chart and graph functions, calculations, mapping and other functionality. Students will be taught quick table calculations, reference lines, different types of visualizations, bands and distributions, parameters, motion charts, trends and forecasting, formatting, stories, performance recording and advanced mapping.

At the end of this course, students will be prepared, if they desire, to earn such industry desktop certifications as a Tableau Desktop Specialist, a Tableau Certified Associate, or a Tableau Certified Professional.

Deep Learning for Computers

Course Description

This course provides a comprehensive overview of Computer vision problems and how they can be tackled using various Convolutional Neural networks (CNNs). Students start with classical image processing operations like edge detection, convolution, shape detectors and color space conversions. This is followed by a foundational understanding of Deep-Convolutional Neural networks and how their training and evaluation works. We introduce various CNN specific layers like pooling-layers and upsampling layers. We also introduce various Data Augmentation techniques that are very helpful for image-related problems. This is followed by a dive deep into the internals of popular CNN architectures like: AlexNet, VGGNet, ResNet etc. Students also learn how to use these methods practically for transfer learning. Students will study how various computer-vision related tasks like image segmentation, image-generation, object detection and localization, contrastive learning etc., can be performed using state of the art algorithms for each of these tasks. Most of these techniques would be studied directly from the original research papers and open-source code provided by the authors. Students would also implement some of these algorithms from scratch in this course.

Deep Learning for Natural Language Processing (NLP)

Course Description

This course focuses on modeling sequences (text, music, time-series, genes) using deep-learning models. We start with a simple Recurrent Neural Network and its limitations with long-sequences. Students learn LSTMs and GRUs which can handle significantly longer sequences to model sequence data like text, music, gene-sequences and time-series data. We study variations of LSTM like bi-directional LSTMs and encoder-decoder architectures. This is followed by a detailed study of attention mechanism and Transformer based models which are currently the state-of-the-art for NLP and sequence modeling. The course teaches encoder-decoder Transformers, BERT, BERT-variations, GPT-1,2 & 3 models from both the architectural and mathematical viewpoints and also a practical viewpoint. Students learn to implement many of these complex models from scratch (using TensorFlow 2 and Keras) to gain a deeper understanding of how they work internally. Students will study popular applications of deep-learning in NLP like parts-of-speech tagging, question-answering systems, conversational engines (chatbots), Semantic search with low-latency etc. For each of these problems, Students will study cutting edge deep-learning models along with code implementations.

Design and Analysis of Algorithms

Course Description

This is a foundational and mandatory course which aims to build student's ability to apply various algorithmic design methods to provide an optimal solution to computational problems. This course starts with time and space complexity analysis of divide and conquer algorithms using recursion-tree based methods and Master's theorem. Students would also learn about amortized time and space complexity analysis for randomized/probabilistic algorithms. Various algorithmic design strategies would be introduced via real world examples and problems. Students would learn when, where and how to optimally use Divide and Conquer, Dynamic programming (top-down and button-up), Greedy, Backtracking and Randomization strategies with examples. The course uses various practical examples from Array manipulations, Sorting, Searching, String manipulations, Tree & Graphs traversals, Graph path-finding, Spanning Trees etc., to introduce the above algorithmic strategies in action. Students would implement many of the above algorithmic design methods from scratch as part of the assignments. The course also introduces how some of these popular algorithms are readily available via popular libraries in various programming languages.

Design Patterns

Course Description

This course provides a practical understanding of popular object-oriented design patterns so that students can reuse design strategies developed for commonly occurring problems in software development. We begin the course with a revision of object-oriented programming and an overview of UML (unified modeling language) diagrams to represent software design diagrammatically. We then dive into 10-12 most popular design patterns motivating each of them from real world scenarios. We would also showcase multiple open source code bases which use the specific design pattern to solve a real-world design problem. This would help students gain an appreciation of how each of the theoretical patterns they learn actually translate to code. We also take up real world cases and dive into various design patterns that can be used to solve the problem. Sometimes, there could be multiple valid designs. We would five into the pros and cons of each design decision and trade-offs involved. Our objective is to build the problem-solving ability amongst students to recognize the appropriate design pattern to tackle a real-world problem. The course briefly discusses domain specific design patterns in their respective contexts.

DevOps

Course Description

This course provides students with hands-on experience on deploying high velocity applications and services reliably on complex and distributed infrastructure. DevOps as a philosophy is a key driver of the modern software life cycle which prefers rapid and reliable delivery of functionality and features via code. We start with a solid introduction to Linux scripting and networking. Then, we learn popular methodologies to deploy complex and

distributed software like microservices, containerization (Docker) and orchestration (Kubernetes). All of this would be introduced with real world examples from the industry. We also focus on Continuous Integration and Continuous Delivery (CI/CD) methodology and how it can be achieved using popular toolchains like Jenkins. We dive into how automated testing of software can be achieved using libraries like Selenium. This shall be followed by more advanced techniques like serverless-compute, Platform as a service model and Cloud-DevOps. Students would learn to monitor and log key data points to ensure they maintain a healthy system and adapt it as needed. Infrastructure-as-code is a key component of modern DevOps especially on cloud and containerized applications which would also be covered with real-world examples.

Distributed Cloud Computing

Course Description

This course provides an in-depth architectural overview and hands- on experience with building scalable data processing and distributed computing via various cloud systems. We focus a lot on Spark which is one of the most popular and powerful distributed systems to perform petabyte scale data processing. We learn various components of Spark like HDFS, Resilient Distributed Datasets (RDDs), Programming models like Map-reduce. Students also learn SparkSQL and Hive and how they can be used for querying large datastores. We focus on how various services in a cloud (like AWS) can be used together to build scalable data-pipelines for both batch and near real-time processing. We show various examples of real world systems and their architectures from various companies and organizations. We learn how graphX can be used to process large graphs using Spark. Students use AWS Elastic Map Reduce (EMR) for cloud based Spark clusters. We learn the design and architecture of distributed inverted indices and how they can be used for implementing search scalably. Students learn to use ElasticSearch, a very popular distributed inverted index for implementing search functionality on websites and on unstructured data.

Distributed Machine Learning

Course Description

This course provides an in-depth understanding of distributed systems for ML and Deep Learning using CPU,GPU and TPU clusters. It starts with foundations of Map-reduce framework and in-memory distributed and resilient data structures that form the backbone of Spark. Students will learn the architectural details of these distributed system platforms and how they can be leveraged to perform data analysis and model training on petabyte scale datasets. We cover how distributed training is achieved for popular ML algorithms on Spark by understanding the internal working of SparkMLLib. The course then focuses on understanding distributed graph processing using GraphX. Students move on to Deep-Learning algorithms and how distributed algorithms can be designed for them when we have GPU or TPU clusters at our disposal. We also dive deep into how TensorFlow archives distributed computing for popular Deep Learning algorithms. Students will study

distributed data stores and how they can be used for ML using popular datastore systems like Hive and SparkSQL. The course concludes by discussing state of the art distributed, low-latency approximate nearest neighbor algorithms along with their implementations in ElasticSearch.

Distributed Systems with High-Level Design

Course Description

A distributed system is an application that executes a collection of protocols to coordinate the actions of multiple processes on a network, such that all components cooperate together to perform a single or small set of related tasks.

Goals of a Distributed System:

- Transparency -> End user does not know what lies behind and how the system is working internally.
- Scalability - > Refers to the growth of the system.
- Availability -> Refers to the system's uptime.

The course will carefully examine three case studies, with attention to such topics as:

- Basics of High Level System Design and consistent Hashing
- Caching
- CAP Theorem
- Replication and Master-Slave
- NoSQL
- Differences between SQL and NoSQL
- Multi Master
- Apache Zookeeper & Apache Kafka
- Case Study on ElasticSearch
- AWS S3 and Quad Trees
- Design Distributed Crawler
- Microservices and Containerisation
- Hotstar & IRCTC System design

Foundations of Cloud Computing

Course Description

This is a course that focuses both on architectural design and practical hands-on learning of the most used cloud services. The course extensively uses Amazon Web services (AWS) to show real world code examples of various cloud services. It also covers the core concepts and architectures in a platform agnostic manner so that students can easily translate these learnings to other cloud platforms (like Azure, GCP etc.). The course starts with virtualization and how virtualized computer instances are created and configured. Students also learn how to auto-scale applications using load balancers and build fault tolerant applications across a geographically distributed cloud. As relational databases are widely used in most enterprises, students learn how to migrate and scale (both vertically and horizontally) these databases on the cloud while ensuring enterprise grade security. Virtual

private clouds enable us to create a logically isolated virtual network of compute resources. Students learn to set up a VPC using virtualized-compute-servers on AWS. The course also covers the basics of networking while setting up a VPC. Students learn of the architecture and practical aspects of distributed object storage and how it enables low latency and high availability data storage on the cloud.

Foundations of Machine Learning

Course Description

This course focuses on building basic classification and regression models and understanding these models rigorously both with a mathematical and an applicative focus. It opens with a basic introduction to high dimensional geometry of points, distance-metrics, hyperplanes and hyperspheres. Then, it introduces the mathematical formulation of logistic regression to find a separating hyperplane. Vector calculus and gradient descent (GD)-based algorithms are explored to learn to solve the optimization problem, including computational variations of GD like mini-batch and stochastic gradient descent. The course also covers other popular classification and regression methods like k-Nearest Neighbours, Naive Bayes, Decision Trees, Linear Regression etc, to show how each of these techniques performs under various real-world situations like the presence of outliers, imbalanced data, multi class classification etc. Lectures on bias and variance tradeoff and various techniques to avoid overfitting and underfitting are incorporated. Algorithms are taught from a Bayesian viewpoint along with geometric intuition. This course would be heavily hands-on where students apply all these classical techniques to real world problems.

Front End Development

Course Description

This course builds upon the introductory JavaScript course to acquaint students of popular and modern frameworks to build the front end. We focus on three very popular frameworks/libraries in use: React.js, jQuery and AngularJS. We start with React.js, one of the most popular and advanced ones amongst the three. students learn various components and data flow to learn to architect real world front end using React.js. This would be achieved via multiple code examples and code-walkthroughs from scratch. We would also dive into React Native which is a cross platform Framework to build native mobile and smart-TV apps using JavaScript. This helps students to build applications for various platforms using only JavaScript. jQuery is one of the oldest and most widely used JavaScript libraries, which students cover in detail. Students specifically focus on how jQuery can simplify event handling, AJAX, HTML DOM tree manipulation and create CSS animations. We also provide a hands-on introduction to AngularJS to architect model-view-controller (MVC) based dynamic web pages.

Front End UI/UX Development

Course Description

This is a hands-on course on designing responsive, modern and light-weight UI for web, mobile and desktop applications using HTML5, CSS and Frameworks like Bootstrap 4. This course starts with an introduction on how web browsers, mobile apps and web servers work. We then dive into each of the nitty gritty details of HTML5 to build webpages. We would start with simple web pages and then graduate to more complex layouts and features in HTML like forms, iFrames, multimedia-playback and using web-APIs. We then go on to learn stylesheets based on CSS 4 and how browsers interpret CSS files to render web pages. Once again, we use multiple real world example web pages to learn the internals of CSS4. We learn popular good practices on writing responsive HTML and CSS code which is also interoperable on mobile browsers, apps and desktop apps. We would introduce students to building desktop apps using HTML and CSS using toolkits like Electron. We would also study popular frameworks for front end development like Bootstrap 4 which can speed up UI development significantly.

Further Studies in Data Science and Data Analytics

Course Description

This advanced graduate class addresses a unique topic on a rotating basis in order to keep the program at the forefront of scholarly research and industry practice. Every year the academic staff member will approve of a new topic to be covered. The bibliography will contain not less than 8 peer-reviewed articles or scholarly publications reflecting the current topic.

Though the exact topic will vary, the emphasis of this course is practical, domain-specific issues in data science. Topics might include data handling, big data management systems, optimization, sparse signal recovery, principal component analysis, or deeper explorations of text mining, natural language processing, computer vision, or other topics introduced in other courses.

Often, Further Studies in Data Science and Data Analytics will extend, complicate, or otherwise deepen the topic taken on in its predecessor course, Studies in Data Science and Data Analytics, giving students who elect this sequence to develop genuine expertise in a specific domain.

High Dimensional Data Analysis

Course Description

This course is aimed to help learners understand various techniques and algorithms to visualize, analyze and understand high dimensional data which is very common in Data Science and ML. The course starts with linear algebraic methods like Principal Component Analysis (PCA) and SVD (Singular Value Decomposition) for obtaining linear projection of high dimensional data. This is followed by more advanced nonlinear and state of the art techniques like t-SNE and UMAP for visualizing high dimensional data. Each of these

techniques would be covered in full mathematical detail from first principles along with applying them to real world datasets in NLP, Genomics and internet-datasets. Students will also study how PCA and SVD are related to general Matrix Factorization techniques. To analyze and understand high dimensional un-labelled data, students learn clustering techniques like K-Means, Gaussian Mixture models, Hierarchical Clustering and DBSCAN. The course shows how some of the techniques are mathematically related to Matrix Factorization. Students study various outlier detection techniques based on density, proximity, factorization and cluster analysis.

Introduction to Computer Programming: Part 1

Course Description

This course helps students translate advanced mathematical/statistical/scientific concepts into code. This is a course for writing code to solve real-world problems. It introduces programming concepts (such as control structures, recursion, classes and objects) assuming no prior programming knowledge, to make this course accessible to advanced professionals from scientific fields like Biology, Physics, Medicine, Chemistry, Civil & Mechanical Engineering etc. After building a strong foundation for converting scientific knowledge into programming concepts, the course advances to dive deeply into Object-Oriented Programming and its methodologies. It also covers when and how to use inbuilt-data structures like 1-Dimensional and 2-Dimensional Arrays before introducing the concepts of computational complexity to help students write optimized code using appropriate data structures and algorithmic design methods.

The course can be taught to allow students to learn these concepts using a modern programming language such as Java or Python. The course offers students the ability to identify and solve computer programming problems in scientific fields at a graduate level.

The course prepares students to handle advanced data structures and algorithm design methods in the separate course, ‘Data Structures’.

Introduction to Computer Programming: Part 2

Course Description

This course provides a practical and detailed understanding of popular programming paradigms and data storage types. Students learning this will be able to write and solve programming problems. The course starts from the basics about functions, various built in functions and how to code user defined functions. Then students will learn about various data type storages and learn about lists and how various manipulations can be done lists like list slicing and also go through examples of 2D Lists.

While learning how to create functions students have to learn how various results and inputs can be stored using different data types after the introduction and discussion on Lists, students will go through sets, tuples, Dictionaries and Strings.

The student should be well prepared to apply these concepts and build algorithms and software using what they learnt in this course.

Introduction to Deep Learning

Course Description

This course provides a strong mathematical and applicative introduction to Deep Learning. The course starts with the perceptron model as an over simplified approximation to a biological neuron. We motivate the need for a network of neurons and how they can be connected to form a Multi Layered Perceptron (MLPs). This is followed by a rigorous understanding of back-propagation algorithms and its limitations from the 1980s. Students study how modern deep learning took off with improved computational tools and data sets. We teach more modern activation units (like ReLU and SeLU) and how they overcome problems with the more classical Sigmoid and Tanh units. Students learn weight initialization methods, regularization by dropouts, batch normalization etc., to ensure that deep MLPs can be successfully trained. The course teaches variants of Gradient Descent that have been specifically designed to work well for deep learning systems like ADAM, AdaGrad, RMSProp etc. Students also learn AutoEncoders, VAEs and Word2Vec as unsupervised, encoding deep-learning architectures. We apply all of the foundational theory learned to various real world problems using TensorFlow 2 and Keras. Students also understand how TensorFlow 2 works internally with specific focus on computational graph processing.

Introduction to Machine Learning

Course Description

This course focuses on building basic classification and regression models and understanding these models rigorously both with a mathematical and an applicative focus. The course starts with a basic introduction to high dimensional geometry of points, distance-metrics, hyperplanes and hyperspheres. We build on top this to introduce the mathematical formulation of logistic regression to find a separating hyperplane. Students learn to solve the optimization problem using vector calculus and gradient descent (GD) based algorithms. The course introduces computational variations of GD like mini-batch and stochastic gradient descent. Students also learn other popular classification and regression methods like k-Nearest Neighbours, Naive Bayes, Decision Trees, Linear Regression etc. Students also learn how each of these techniques under various real world situations like the presence of outliers, imbalanced data, multi class classification etc. Students learn bias and variance trade-off and various techniques to avoid overfitting and underfitting. Students also study these algorithms from a Bayesian viewpoint along with geometric intuition. This course is hands-on and students apply all these classical techniques to real world problems.

Introduction to Problem-Solving Techniques: Part 1

Course Description

The ability to solve problems is a skill, and just like any other skill, the more one practices, the better one gets. So how exactly does one practice problem solving? Learning about different problem-solving strategies and when to use them will give a good start. Problem solving is a process. Most strategies provide steps that help you identify the problem and choose the best solution.

Building a toolbox of problem-solving strategies will improve problem solving skills. With practice, students will be able to recognize and choose among multiple strategies to find the most appropriate one to solve complex problems. The course will focus on developing problem-solving strategies such as abstraction, modularity, recursion, iteration, bisection, and exhaustive enumeration.

The course will also introduce arrays and some of their real-world applications, such as prefix sum, carry forward, subarrays, and 2-dimensional matrices. Examples will include industry-relevant problems and dive deeply into building their solutions with various approaches, recognizing each's limitations (i.e when to use a data structure and when not to use a data structure).

By the end of this course a student can come up with the best strategy which can optimize both time and space complexities by choosing the best data structure suitable for a given problem.

Introduction to Problem-Solving Techniques: Part 2

Course Description

This course is a follow-up to Introduction to Problem-Solving Techniques: Part 1, and as part of their academic planning process with Woolf staff, students will ordinarily take that course first.

Part 2 deepens the approach to data structures by including such topics as stacks, queues, linked lists, and trees, and discussing in detail real world applications of each approach and their comparative strengths and limitations (i.e when to use a data structure and when not to use a data structure). This course will also include hashing techniques along with recursion and subset problems. This course will have rigorous homework and assignments to support the introduction of more than 4 data structures.

By the end of this course a student can come up with the best strategy which can optimize both time and space complexities by choosing the best data structure suitable for a given problem.

JavaScript

Course Description

This course is a hands-on course covering JavaScript from basics to advanced concepts in detail using multiple examples. We start with basic programming concepts like variables, control statements, loops, classes and objects. Students also learn basic data-structures like Strings, Arrays and dates. Students also learn to debug our code and handle errors gracefully in code. We learn popular style guides and good coding practices to build readable and reusable code which is also highly performant. We then learn how web browsers execute JavaScript code using V8 engine as an example. We also cover concepts like JIT-compiling which helps JS code to run faster. This is followed by slightly advanced concepts like DOM, Async-functions, Web APIs and AJAX which are very popularly used in modern front end development. We learn how to optimize JavaScript code to run on both mobile apps and mobile browsers along with Desktop browsers and as desktop apps via ElectronJS. Most of this course would be covered via real world examples and by learning from JS code of popular open-source websites and libraries.

Low-Level design & Design Patterns

Course Description

Low-Level Design & Design Patterns focuses on modularity and reusability in software design, common design vocabularies, refactoring and how to reduce it, and how to incorporate design patterns into iterative development processes. The course pays significant attention to the interaction between system architecture and components, including data organization.

The course begins with Object-Oriented Analysis (OOA), which is a problems-solving technique that includes: modeling an information design; representing behavior; describing functions; dividing data, functional, and behavioral models to uncover detail; moving from abstraction to implementation details. The course then turns to Object-Oriented Design (OOD), which reduces the analysis model into a modular design for software creation, with subsystems, components, and objects.

The iteration of analysis and implementation will be covered in detail with real-world industry examples.

Mathematics for Computer Science

Course Description

Mathematics and computer science are closely related fields. Problems in computer science are often formalized and solved with mathematical methods. It is likely that many important problems currently facing computer scientists will be solved by researchers skilled in algebra, analysis, combinatorics, logic and/or probability theory, as well as computer science.

This course covers discrete mathematics for computer science and engineering. Topics may include asymptotic notation and growth of functions; permutations and combinations; counting principles; discrete probability. Further selected topics may also be covered, such as recursive definition and structural induction; state machines and invariants; recurrences; generating functions.

Students will be able to explain and apply the basic methods of discrete (noncontinuous) mathematics in computer science. They will be able to use these methods in subsequent courses in the design and analysis of algorithms, computability theory, software engineering, and computer systems. The focus of the course is real-world problems and applications often found in business and industry.

NoSQL Cloud Computing

Course Description

This course provides a comprehensive overview and practical knowledge of various NoSQL data stores and how they can be used on the Cloud (AWS). We focus on three NoSQL datastores in this course: MongoDB, DynamoDB and Redis. For each of them, we first understand the design and architecture in depth. We compare and contrast each of them with traditional relational databases and other NoSQL databases so that students understand the engineering trade-offs when using them. We take multiple real-world case-studies from various companies and organizations to discuss which datastore is more apt in each situation to help students better appreciate the differences and use-cases. We dive into the technical details from setting up and deploying each of these datastores on the cloud (AWS) with latency and scalability in mind. We also discuss various datastore specific optimizations and good practices to follow. The course also teaches students how to stress test each of these datastores under differing loads to compare and contrast which would be a better fit in a real world scenario. At the end of this course, students would be able to choose an optimal data store for their engineering needs to build websites or build data pipelines or deploy machine learning applications.

Numerical Programming in Python

Course Description

This course helps students translate mathematical/statistical/scientific concepts into code. This is a foundational course for writing code to solve Data Science ML & AI problems. It introduces basic programming concepts (like control structures, recursion, classes and objects) from scratch, assuming no prerequisites, to make this course accessible to students from non-computational scientific fields like Biology, Physics, Medicine, Chemistry, Civil & Mechanical Engineering etc. After building a strong foundation, the course advances to dive deep into core Mathematical libraries like NumPy, Scipy and Pandas. Students also learn when and how to use inbuilt-data structures like Lists, Dicts, Sets and Tuples. The course introduces the concepts of computational complexity to help students write optimized code using appropriate data structures and algorithmic design methods. The course does not dive

deep into the data structures and algorithm design methods in this course - that is available in the ‘Data Structures and Algorithms’ course. This course is valuable for all students specializing in mathematical sub-areas of CS like ML, Data Science, Scientific Computing etc.

Power BI for Data Analysis and Exploration

Course Description

Power BI is a Microsoft tool that works on turning unrelated sources of data into coherent, visually immersive, and interactive insights. Input data can be of various formats ranging from spreadsheets to JSON.

Learning this tool will help in working on real time data creating solutions for business with interactive solutions from very unstructured data and reporting with business insights. Students will learn how to handle data sources in Power BI, connecting to various data sources using Power BI, query editors, managing data relationships, and cross filter direction. In Power BI students will learn how to visualize data like map visualizations, funnel charts, waterfall charts. For Data Analysis students will learn different data types in DAX, Syntax used, DAX functions, operators, tables and filters used. Parameter Naming. Power BI is used extensively for report making so students will learn how to report basic servers, web portal, paginated reports, schedule refresh and how to configure schedule refresh, publish to web embedded code. Students will also learn how to use R language and Python on Power BI. Hands-on training with a project will also be dealt with where students will work on real-time industry data and provide reports.

Practical Software Engineering

Course Description

This course gives the detailed overview on how to approach Low Level Design problems with real-world case studies discussed such as Designing a Pen (Mac/Windows), TicTacToe, BookMyShow (most used event booking app, manages millions of users), Email campaign Management System and detailed design of Splitwise.

Product Analytics

Course Description

This course teaches students how to analyze the ways users engage with a service. This method, called product analytics, helps businesses track and analyze user data. Students will learn more deeply what is required to move a product from idea to implementation, through to launch, and then on to iterative improvements. The course teaches how to measure progress, validate or update product hypotheses, and present product learnings.

Also, students will gain experience in making informed decisions, as well as how to present findings and make an analytics-informed business case to win support for a product.

Productization of Machine Learning (ML) Systems

Course Description

This course aims to build the core competency of building real world end-to-end ML systems and deploy them into production for a variety of problems and scenarios. Students would learn a variety of ML systems ranging from high throughput and low latency internet scale systems to low compute power and energy constrained IoT devices like smart watches. Students will study the ML lifecycle and various components in detail. We also use real world ML platforms like Google's KubeFlow, TensorFlow Lite, and Amazon's SageMaker to implement real world systems and understand the engineering trade-offs and challenges. Students also learn relevant technologies and tools like Containerization (Docker) and Container Orchestration (Kubernetes) and Git which are often used extensively in real world scalable ML systems. This course is a hands-on course where we solve multiple real world cases and discuss solutions built by various companies and organizations to provide the students a comprehensive understanding of varied systems and design choices.

Product Management for Software Engineers

Course Description

Every organization is building products to solve the pain points of its customers. Product managers are a critical part of an organization, who make sure that evolving customer needs, and market trends are observed and converted into delightful solutions which help businesses get their outcomes.

In this course, students will get a fundamental understanding of product management practices.

This will give them a comprehensive view of the complete product management life cycle.

Relational Databases

Course Description

This is a core and foundational course which aims to equip the student with the ability to model, design, implement and query relational database systems for real-world data storage & processing needs. Students would start with diagrammatic tools (ER-diagram) to map a real world data storage problem into entities, relationships and keys. Then, they learn to translate the ER-diagram into a relational model with tables. SQL is then introduced as a de facto tool to create, modify, append, delete, query and manipulate data in a relational database. Due to SQL's popularity, the course spends considerable time building the ability

to write optimized and complex queries for various data manipulation tasks. The course exposes students to various real world SQL examples to build solid practical knowledge. Students then move on to understanding various trade-offs in modern relational databases like the ones between storage space and latency. Designing a database would need a solid understanding of normal forms to minimize data duplication, indexing for speedup and flattening tables to avoid complex joins in low-latency environments. These real-world database design strategies are discussed with practical examples from various domains. Most of this course uses the open source MySQL database and cloud-hosted relational databases (like Amazon RDS) to help students apply the concepts learned on real databases via assignments.

Spreadsheets for Data Understanding

Course Description

Spreadsheets for Data Understanding introduces students to the principles and techniques of data cleaning, handling data sets of varying sizes, and visualizing data/data storytelling. Students will also learn the basics of predictive modeling from data sets. These are all introduced through the means of Microsoft Excel, the industry-standard spreadsheet program. Students will learn how to use inbuilt functions, as well as techniques such as creating and modifying pivot tables.

SQL for Data Analytics

Course Description

Structured Query Language (SQL) is key to working with data in relational databases, a task at the core of data science and analytics. In this course, students will learn all the major keywords and clauses used to extract data, best practices for formatting SQL queries, and how to generate meaningful insights from the results.

The focus is at all times on real-world uses of SQL queries, syntax, and expression, to allow students to begin professional-level work as quickly as possible.

Statistical Programming

Course Description

This course focuses on representing statistical techniques in code, and may be conducted in Python, R, or another relevant language. Such languages provide libraries that can handle a wide variety of statistical techniques like linear and nonlinear modeling, classical statistical tests, time-series analysis, classification, clustering and graphical techniques, and are highly extensible.

Learning to work in statistically-oriented programming language environments can equip you with the following skills among many others:

1. An effective way of data handling (using arrays for example) and storing data in a structured manner.
2. Expertise in diverse tools and libraries for Data Analysis
3. Ability to present complex data in a graphical and visual format for easy understanding of the data and further solutions.

Studies in Data Science and Data Analytics

Course Description

This advanced graduate class addresses a unique topic on a rotating basis in order to keep the program at the forefront of scholarly research and industry practice. Every year the academic staff member will approve of a new topic to be covered. The bibliography will contain not less than 8 peer-reviewed articles or scholarly publications reflecting the current topic.

Though the exact topic will vary, the emphasis of this course is practical, domain-specific issues in data science. Topics might include data handling, big data management systems, optimization, sparse signal recovery, principal component analysis, or deeper explorations of text mining, natural language processing, computer vision, or other topics introduced in other courses.

System Design

Course Description

This course is aimed at equipping students with skills to architect the high level design (a.k.a. system design) of software and data systems. We start with some of the good to have properties of large complex software systems like scalability, reliability, availability, consistency etc. The course teaches various patterns and design choices we have to satisfy each of these good to have properties. We then go on to understand key components of system design like load-balancers, microservices, reverse-proxies, content-delivery networks etc. Students learn how each of them work internally along with real world implementations of each. We study various NoSQL data stores, their internal architectures and where to use which one with real-world examples. Students also learn popular data encoding schemes like XML and JSON. We learn how to build data pipelines using batch and stream processing systems. We also work on multiple real world cases on architecting on the cloud using popular open-source libraries and tools. Students will study design documents and high-level-design of popular internet applications and services like video-conferencing, recommender-systems, peer-to-peer chat, voice-assistants etc.