Course Descriptions

Comprehensive Academic Transcript

Three-Year Bachelor's Degree in Applied Computer Science and Artificial Intelligence

Sapienza University of Rome

Application to the Master of Science in Informatics

Technical University of Munich (TUM)

Date of Birth: 29 October 2002

Language Note: All courses were conducted fully in English.
Exam Workload: Every taught course (excluding Internship AAF1466) carries 6 ECTS credits, with 60 in-class hours and 90 hours of self-study.

Application of Fausto Zamparelli

Contents

1 First Year

1.1 First Semester

1.1.1 Calculus - Unit 1

Course Code: 1.1.1 — 10595099 Disciplinary Area: MAT/05

The course begins with the definition of real numbers, including proofs by contradiction and by induction. It covers the properties of powers, logarithms, exponentials, and trigonometric functions. The study of number sets includes definitions of maximum, minimum, supremum, and infimum of a set.

Next, it introduces real functions: their definition, domain, symmetry properties, and inverse functions. It then explores sequences, their definition, properties (monotonicity, boundedness), and recursively defined sequences. The concept of limit of a sequence is introduced along with calculation rules (limits of sums, products, quotients, variable change), and limits of functions, emphasizing uniqueness, the squeeze theorem (also known as the two policemen theorem), and the Bolzano–Weierstrass theorem (without proof). It also includes the classification of infinite and infinitesimal quantities.

For continuity, it covers limit properties of continuous functions, the sign permanence theorem, and notable limits. The intermediate value theorem and zero existence theorem are introduced (without proof), along with definitions of local and global maxima and minima, supremum and infimum of functions, and the Weierstrass theorem (without proof).

The derivative is defined and applied to sums, products, quotients, and composite functions (chain rule). Key theorems include: differentiability implies continuity, Fermat's theorem, Rolle's theorem, and the Mean Value Theorem (Lagrange). The first derivative test relates monotonicity to the sign of the derivative. The study extends to concavity, convexity, and Taylor expansions with Peano and Lagrange remainders, including error estimates.

Function analysis includes graphical studies and the Newton method for finding roots. Finally, the course introduces complex numbers: their definition, basic properties, and the nth root of a complex number.

1.1.2 Computer Architecture - Unit 1

Course Code: 1.1.2 — 10595546 Disciplinary Area: INF/01

This unit introduces digital logic and microarchitecture with an emphasis on reasoning about hardware and synthesizable design. It starts from number systems and binary arithmetic, progressing to Boolean algebra, combinational circuits (adders, multiplexers, encoders), and timing and propagation delay analysis.

Sequential circuits (latches, flip-flops) and synchronous design principles are covered, together with Finite State Machine (FSM) design and timing closure. Students use Hardware Description Languages (SystemVerilog, VHDL) to model and simulate both combinational and sequential logic and implement designs on FPGAs.

The unit also examines arithmetic logic units, register files, memory array organization, and programmable logic arrays, and discusses typical design and synthesis workflows.

1.1.3 Linear Algebra

Course Code: 1.1.3 — 10595524 Disciplinary Area: MAT/02

This unit begins with solving systems of linear equations in multiple variables using Gaussian elimination to reduce augmented matrices to row echelon or reduced row echelon form.

It explores the structure of vector spaces, including how to determine if a set with two operations forms a vector space, and whether a subset is a subspace. Key concepts include linear dependence and independence, and spanning sets.

The course covers essential matrix algebra techniques, including finding matrix inverses via Gauss-Jordan elimination, and identifying a matrix's row space, column space, and null space.

It introduces the dimension of subspaces, and the rank and nullity of a matrix. Students also study linear transformations, learning how to compute their kernel and range, and find bases for each.

Finally, the course covers eigenvalues and eigenvectors, how to derive the characteristic equation, and how to diagonalize a matrix when possible. The course concludes with orthogonality, orthonormality, and the Gram–Schmidt process for generating orthonormal sets.

1.1.4 Programming – Unit 1

Course Code: 1.1.4 — 10595102 Disciplinary Area: INF/01

This unit provides a practical, technical introduction to programming in Python with emphasis on algorithmic thinking and performance. Core topics include primitive and composite data types, control flow, functions and modules, scoping, exception handling, and file I/O. A major focus was on implementing and analysing algorithms (sorting, searching, recursion vs iteration, divide-and-conquer, greedy and dynamic programming) in Python and writing solutions that prioritize asymptotic efficiency and constant-factor performance where relevant. Students implemented recursive algorithms and optimized them (tail recursion, memoization), and measured algorithmic behaviour empirically. The course covers fundamental data structures (lists, tuples, dictionaries, sets) and how to select representations for performance.

1.1.5 Programming – Unit 2

Course Code: 1.1.5 — 10595102 Disciplinary Area: INF/01

Building on foundational skills, this unit focuses on improving program robustness through debugging and testing techniques, enabling students to identify and fix errors efficiently. It explores the differences between recursive and iterative algorithms, guiding learners to choose appropriate approaches for different problems and to understand the trade-offs involved. The unit surveys useful Python modules for real-world applications (file and text manipulation, networking, image handling) and introduces concurrency primitives (threading, multiprocessing, async/await) with attention to

synchronization and shared-state hazards. Students complete applied assignments that emphasize robustness, logging, error handling, and automation.

1.2 Second Semester

1.2.1 Calculus - Unit 2

Course Code: 1.2.1 — 10595099 Disciplinary Area: MAT/05

The course covers Riemann integration, including the definition of the integral and basic properties. It also introduces improper integrals, where either the interval is infinite or the function is unbounded.

It then explores numerical series, including convergence tests and examples. Special attention is given to power series, their radius of convergence, and basic operations.

Finally, the course introduces ordinary differential equations (ODEs):

- First-order ODEs: linear equations and equations solvable by separation of variables.
- Second-order linear ODEs with constant coefficients, including both the homogeneous case and non-homogeneous case (with forcing term).

1.2.2 Algorithms

Course Code: 1.2.2 — 1049269 Disciplinary Area: INF/01

The course develops algorithmic thinking with rigorous analysis of correctness and complexity (worst-case, amortized, and average-case). Topics include dynamic programming (e.g., segmentation, optimal substructure examples, Bellman-Ford), graph algorithms (BFS/DFS, Dijkstra, Bellman-Ford, minimum spanning trees), and string and numerical algorithms. Data structures studied include arrays, linked lists, stacks, queues, hash tables, disjoint-set (Union-Find), heaps and priority queues, and balanced search trees; students learn how to select structures for performance and memory constraints. The course treats algorithmic paradigms—greedy algorithms (Huffman coding), divide-and-conquer (fast multiplication, merge sort), and randomized/approximation algorithms—and introduces online algorithms and the experts framework. It concludes with an overview of computability and complexity theory (Turing machines, decidability, reductions, NP-completeness) and practical methods for asymptotic analysis and empirical benchmarking.

1.2.3 Computer Architecture - Unit 2

Course Code: 1.2.3 — 10595546 Disciplinary Area: INF/01

This unit connects ISA concepts to processor microarchitecture and memory systems. It covers instruction encoding, addressing modes, and the semantics of machine-level instructions. The processor section examines datapath design, control logic, pipelined execution, and microarchitectural optimizations including branch prediction and out-of-order execution at a conceptual level.

Pipeline hazards (structural, data, control), forwarding, and stall/flush strategies are analyzed. The memory hierarchy is treated in depth: multi-level caches, cache mapping and associativity, replacement policies, coherence protocols (e.g., MESI), and the performance impact of locality. Virtual memory, page tables, TLBs, and protection mechanisms are explained alongside memory consistency models.

The course also surveys parallelism (instruction-level, thread-level, and multi-core) and exposes students to hands-on RISC-V programming and microbenchmarking to measure latency, throughput, and the effect of microarchitectural decisions.

1.2.4 Physics

Course Code: 1.2.4 — 10595523 Disciplinary Area: FIS/01

Mechanics covers physical quantities and measurements, vectors and their operations, motion in one, two, and three dimensions including projectile and circular motion. It includes Newton's laws, forces, friction, work, energy, power, potential energy, and conservation principles. Topics also include center of mass, momentum and impulse, collisions, rotational motion, moment of inertia, angular momentum, and Newton's laws applied to rotation. The unit ends with Newton's law of gravitation and gravitational fields.

Fluid Mechanics and Thermodynamics address density, pressure, fluid statics, Pascal's and Archimedes' principles, and the continuity equation. Thermodynamics topics include temperature, heat transfer, the first law, ideal gases, entropy, and the second law.

Electricity and Magnetism include Coulomb's law, electric fields from charges and distributions, electric flux and Gauss's law, electric potential and capacitance, current and resistance with Ohm's law, and power. Magnetic fields, forces on charges and currents, Ampere's law, solenoids, Faraday's law, induction, inductance, Maxwell's equations, and electromagnetic waves are also studied.

Modern Physics introduces quantum physics, nuclear physics, particle physics, and radioactivity.

1.2.5 Programming 2

Course Code: 1.2.5 — 10600241 Disciplinary Area: INF/01

The course teaches professional Java development and advanced object-oriented and concurrent programming. Core topics include Java language fundamentals, ADTs, class design and encapsulation, inheritance and polymorphism, interfaces, generics, and the Java Collections Framework. Modern language features are covered such as lambda expressions and the Streams API. Concurrency topics include thread lifecycle, synchronization primitives, concurrent collections, Executors, and common concurrency pitfalls (deadlock, livelock). Practical aspects emphasised include debugging, profiling, and effective use of the Java standard library. Advanced topics touch on reflection, annotations, and design patterns. The capstone project implemented a blockchain prototype in Java, integrating networking, serialization, consensus logic, and persistence; the full codebase is published at BlockChain extunderscore Java.

2 Second Year

2.1 First Semester

2.1.1 Calculus 2

Course Code: 2.1.1 — 10595529 Disciplinary Area: MAT/05

This course aims to provide students with an in-depth understanding of Multivariable Calculus and Vector Analysis, essential for advanced studies in mathematics, physics, and engineering.

It begins with the classification and sketching of surfaces, focusing on identifying the exact types of cylinders and various quadric surfaces such as ellipsoids, hyperboloids, paraboloids, and cones. This geometric foundation helps students visualize complex 3D objects.

Students then study multivariable functions, learning how to rigorously compute and analyze limits and continuity in higher dimensions, which is crucial for understanding behavior near points in multi-dimensional domains.

The course introduces partial derivatives, teaching students how to compute them and apply them to find local extrema of functions of several variables, including techniques for identifying saddle points and applying second derivative tests. The concept of the directional derivative is developed to understand rates of change in arbitrary directions, and tangent planes to surfaces are derived using gradients.

Moving forward, students explore multiple integrals — double and triple integrals — learning how to set up and compute integrals over general regions in two and three dimensions. Applications include calculating areas of closed bounded regions in the plane and volumes of solid bodies, with an emphasis on changing variables and using coordinate transformations such as polar, cylindrical, and spherical coordinates.

The course also covers line integrals along curves in space, enabling students to compute physical quantities like work done by a force field and circulation. This includes understanding parameterization of curves, scalar and vector fields, and fundamental theorems that connect line integrals with surface integrals and gradients.

2.1.2 Probability

Course Code: 2.1.2 — 10595525 Disciplinary Area: MAT/06

The course covers foundational probability theory, starting with probability spaces, events, and probability mass functions. It introduces conditional probability, the law of total probability, compound probability formulas, probability trees, and Bayes' theorem. Key concepts such as independence and conditional probability, alongside basic combinatorics, are also explored.

The language of random variables is developed next, including probability mass functions (PMF), probability density functions (PDF), and cumulative distribution functions (CDF). Students learn about expectation (mean), variance, joint, marginal, and conditional distributions, as well as functions of random variables. Linear combinations of random variables are analyzed, with focus on

means, variances, and covariances. Important inequalities like Markov's, Jensen's, and Chebyshev's are introduced, leading to applications such as the law of large numbers for sample means.

The course studies key probability distributions: binomial, geometric, Poisson (as a limit of binomials), multinomial, uniform, and exponential (linked to the Poisson process). Finally, it covers the normal distribution and its linear properties, the normal approximation to other distributions, and the central limit theorem.

2.1.3 Systems and Networking - Unit 1 (Operating Systems)

Course Code: 2.1.3 — 10595616 Disciplinary Area: INF/01

This unit covers operating-system principles with hands-on systems programming. It introduces kernel architecture, system calls, and the interface between hardware and OS. Process management topics include process lifecycle, context switching, CPU scheduling algorithms (FIFO, SJF, round-robin, multi-level feedback queues), and lightweight threads. Process synchronization is covered with mutexes, semaphores, condition variables, monitors, and deadlock detection/avoidance strategies. Memory management addresses paging, segmentation, virtual memory, page replacement algorithms, and TLB design. I/O and storage management examine device drivers, interrupt handling, DMA, block I/O stacks, and storage subsystems. File systems are studied from metadata structures to journaling and consistency. The course includes lab work in C/UNIX (POSIX APIs) and exercises implementing parts of an OS or simulating scheduling and memory subsystems.

2.1.4 Systems and Networking - Unit 2 (Networking)

Course Code: 2.1.4 — 10595616 Disciplinary Area: INF/01

This networking unit combines theoretical protocols with practical network engineering skills. It covers the Internet protocol stack, performance metrics (latency, throughput, jitter), and protocol design principles. Application-layer topics include HTTP(S), REST, DNS, SMTP, and P2P models. At the transport layer, TCP and UDP are studied with flow and congestion control (TCP NewReno, Cubic), sockets programming, and reliability mechanisms.

The network layer treats IPv4/IPv6 addressing, routing algorithms, and protocols (RIP, OSPF, BGP) as well as NAT, DHCP, and multicast. Data-link topics include MAC protocols and Ethernet/Wi-Fi fundamentals. The course includes packet capture and analysis (Wireshark), socket programming labs, and basic routing/switching configuration; it also introduces security primitives for networking (TLS, IPsec) and threat mitigation strategies.

2.1.5 Data Management and Analysis- Unit 1

Course Code: 2.1.5 — 10595617 Disciplinary Area: INF/01

The course begins with an introduction to relational algebra, covering fundamental definitions and relational operators essential for querying and manipulating data in relational databases.

Next, it delves into relational theory, focusing on relation schemas, functional dependencies, closures, and keys. Students study normalization principles, particularly the Third Normal Form (3NF), and learn how to perform decompositions to reduce redundancy and improve database design.

The course then explores physical database organization, including indexing techniques such as B-trees, the physical implementation of queries, and strategies for query optimization to enhance performance.

Finally, concurrency control is covered with emphasis on transaction management, locking mechanisms, including two-phase locking protocols, and timestamp ordering to ensure data consistency and isolation in multi-user environments.

2.2 Second Semester

2.2.1 Artificial Intelligence and Machine Learning - Unit 1

Course Code: 2.2.1 — 10595618 Disciplinary Area: INF/01

The course starts with an introduction to Artificial Intelligence, providing an overview of its main subfields and tracing its historical development. It then explores intelligent agents, defining rationality and examining different types of environments and agent architectures.

Next, it covers uninformed search strategies such as breadth-first, uniform-cost, depth-first, depth-limited, and iterative deepening search, explaining their mechanisms and use cases. Building on this, informed search strategies like greedy best-first and A* are studied, with a focus on heuristic functions, including the properties of admissibility and consistency.

The course then addresses adversarial search methods applied to games, illustrating how agents compete in strategic environments.

Constraint satisfaction problems are introduced, detailing their definitions, variations, and constraint propagation techniques including node, arc, path, and k-consistency. Techniques like backtracking, backjumping, and local search are covered to solve these problems efficiently.

Markov decision processes are examined to model nondeterministic search problems, along with their solution methods.

Reinforcement learning is introduced with a focus on different variants, both those that model the Markov decision process and those that do not, highlighting the exploration-exploitation trade-off.

Finally, the course explores knowledge representation and reasoning, focusing on propositional logic-based agents and satisfiability (SAT) problems.

2.2.2 Artificial Intelligence and Machine Learning - Unit 2

Course Code: 2.2.2 — 10595618 Disciplinary Area: INF/01

The course begins with an introduction to Machine Learning (ML), exploring what ML is and how it evolved from rule-based systems to data-driven decision making. It covers the types of problems ML can solve and reviews essential foundational concepts in statistics, linear algebra, and probability theory. Key challenges such as the curse of dimensionality and the assumption that data lie on a low-

dimensional manifold are discussed. Concepts like the multivariate normal distribution, Mahalanobis distance, L_p norms, and the difference between correlation and causation are introduced to frame data analysis.

In supervised learning, the course examines the distinction between regression and classification tasks. It covers parametric linear models including linear regression, least squares, logistic regression, the perceptron algorithm, and gradient descent optimization. Model complexity is analyzed through the bias-variance tradeoff, overfitting and underfitting, empirical risk minimization, and regularization techniques. Support Vector Machines are introduced with concepts such as the optimal hyperplane, margins, and kernel methods. The course then delves into deep learning, explaining deep neural networks (DNNs), multilayer perceptrons, backpropagation, computational graphs, automatic differentiation, and stochastic gradient descent on mini-batches. Non-parametric models such as k-nearest neighbors, decision trees, and random forests are also covered.

Unsupervised learning topics include dimensionality reduction methods like Principal Components Analysis (PCA) and t-SNE, clustering algorithms such as k-means and Expectation-Maximization (EM), and Gaussian Mixture Models (GMM).

The course also focuses on practical aspects of ML systems, including data visualization and analysis, feature engineering versus feature learning, model selection, cross-validation, hyperparameter tuning, error analysis, and evaluation metrics such as accuracy, error rate, precision/recall, ROC and CMC curves, and confusion matrices.

Hands-on experience is emphasized using Python and essential libraries like NumPy for matrix manipulation, scikit-learn for basic ML algorithms, and PyTorch for neural networks and automatic differentiation.

2.2.3 AI LAB: Computer Vision and NLP

Course Code: 2.2.3 — 10595610 Disciplinary Area: INF/01

The AI lab combines practical deep-learning workflows with applied projects in computer vision and NLP. Students learn image-processing fundamentals (filters, morphological operations, feature detectors), and then implement end-to-end CV pipelines using libraries such as OpenCV and PyTorch. Topics include convolutional neural networks for classification and object detection (e.g., Faster R-CNN, YOLO), transfer learning, data augmentation and evaluation metrics (mAP). NLP coverage includes tokenization, embeddings (word2vec, GloVe), sequence models (RNNs, Transformers) and common tasks (classification, sequence tagging, text generation). The lab emphasises reproducible experiments, dataset handling, model training/validation, and deployment prototypes. Final projects demonstrate the full ML lifecycle; mine, Aldone (2024), is a React front-end paired with Node.js and Python services that lets users add and manage tasks via natural-language dialogue, auto-split chores into subtasks, estimate effort, and rely on computer-vision grocery detection to reconcile the digital pantry after shopping.

2.2.4 Data Management and Analysis - Unit 2

Course Code: 2.2.4 — 10595617 Disciplinary Area: INF/01

The course begins with an introduction to data analysis challenges, especially focusing on the issues that arise when handling large datasets. It offers a brief overview of Complex Systems to provide context for data relationships and structures.

Next, the course introduces specialized programming languages used for data analysis, primarily R and Python, equipping students with practical tools to manipulate and analyze data.

Key data processing techniques are covered, including segmentation, clustering, and classification, which form the backbone of organizing and interpreting data.

The course explores the fundamentals of network science, starting with random and scale-free networks, and delves into centrality metrics that identify important nodes within networks. It also covers algorithms for community detection, enabling the understanding of network substructures.

Students learn to design efficient pipelines for data analysis, from feature extraction and data gathering to modeling and visualization, ensuring a comprehensive grasp of the full data analysis workflow. The final group project involves performing a comprehensive statistical analysis on a large dataset using the R programming language.

The capstone deliverable, Book Recommendation System (2024), combines a production-grade Next.js 14 + TypeScript + Tailwind + Nivo frontend with a Supabase/PostgreSQL backend and Python/R analytics pipelines. Users can vote on titles (/vote), explore community favourites (/favs), and receive personalised recommendations (/recs) driven by collaborative filtering and statistical modelling.

2.2.5 Statistics

Course Code: 2.2.5 — 1055043 Disciplinary Area: SECS-S/01

The first part focuses on descriptive statistics, covering types of variables, graphical data representations, measures of central tendency and variability, as well as measures of dependence, including the correlation coefficient.

The second part reviews probability and statistical inference. It begins with fundamental probability concepts and discrete and continuous random variables. Key topics include the central limit theorem and sampling distributions for the mean and variance. The course covers maximum likelihood estimation and various point estimators (mean, variance, proportions for single and two populations). Confidence intervals and hypothesis tests for means and proportions are also thoroughly addressed.

The third part introduces statistical models, focusing on regression analysis. This includes the simple linear regression model, its parameter estimation, and inference. It then extends to multiple linear regression, with emphasis on model estimation, statistical inference, and variable selection techniques. The program concludes with logistic regression, covering its definition, parameter estimation, and inference.

3 Third Year

3.1 First Semester

3.1.1 Foundations of Computer Science

Course Code: 3.1.1 — 10595530 Disciplinary Area: INF/01

Automata and Languages: Study of regular languages through finite state automata and regular grammars. Exploration of context-free languages using context-free grammars and pushdown automata. Introduction to the Chomsky hierarchy.

Computability Theory: Turing Machines and the Church-Turing thesis. Analysis of undecidable problems, such as the Halting Problem. Use of reductions as a method to prove (un)decidability, with examples of both decidable and undecidable problems.

Complexity Theory: Time and space complexity of Turing Machines. Introduction to complexity classes, focusing on tractable problems (class P) and problems that are not known to be tractable (class NP). Polynomial-time reductions and the concept of NP-completeness, including the Cook-Levin theorem and examples of NP-complete problems.

3.1.2 Deep Learning

Course Code: 3.1.2 — 10595531 Disciplinary Area: INF/01

This course introduced deep learning theory and practice, with applications to computer vision and natural language processing. It began with fundamentals (relationship between ML and deep learning), digital image processing and discrete convolutions, and methods for handling text and speech inputs. The class reviewed multinomial logistic regression and linear classifiers, optimization with gradient descent and regularization, and introduced computational graphs and backpropagation.

Convolutional Neural Networks and modern convolutional architectures were studied in depth, including practical training concerns: activation functions, data preprocessing, weight initialization, batch normalization, dropout, data augmentation, hyperparameter tuning and transfer learning. The course examined ConvNets applied to time-series data, recurrent models (RNNs, LSTMs) and Transformer architectures (attention, positional encoding, BERT-style encoders). Generative models were introduced (autoregressive generation, autoencoders, GANs), and the ethical implications of deep learning were discussed. Labs involved training, validating and comparing models on benchmark vision and NLP datasets using contemporary frameworks and reproducible experiment practices.

3.1.3 Law and Computer Science

Course Code: 3.1.3 — 10595537 Disciplinary Area: IUS/20

This course examined the legal and regulatory frameworks that govern computing technologies, with a focus on data protection, artificial intelligence, e-commerce and intellectual property. Students learned European and international regulations relating to data processing and AI governance, and analysed protocols and standards for cross-border cooperation in AI deployment. The curriculum

blended doctrine with case studies and practical exercises to develop the ability to identify legal risks and compliance requirements in public-administration and corporate settings.

Topics included privacy and data-protection law (GDPR concepts), liability and accountability for automated decision-making, copyright and software, contracts and legal aspects of cloud services, and regulatory approaches to AI. The course emphasised the intersection between technical design choices and legal obligations, enabling students to participate effectively in multidisciplinary teams. The final project analysed U.S. copyright law and landmark Supreme Court cases regarding software and creative works, and discussed possible future legal frameworks for AI-generated content and authorship.

3.1.4 Web And Software Architecture

Course Code: 3.1.4 — 10595534 Disciplinary Area: INF/01

This course provides a full-stack, pragmatic approach to modern web and software architecture. It covers RESTful API design principles, OpenAPI/Swagger documentation, JSON schema design, and version control workflows (Git). Backend development focuses on Go (Go modules, concurrency via goroutines and channels, dependency management, testing and benchmarking). Frontend topics include JavaScript, component-based frameworks (Vue.js), state management (Vuex), and build tooling. The course covers containerization with Docker; the final project was a full-stack WhatsApp-like clone deployed with Docker following software engineering best practices (modular services, secure configuration, and container-based deployment) and open-sourced as FriendsText. Emphasis is placed on observability (logging, metrics), API versioning, authentication and authorization patterns (JWT, OAuth2), and scalability considerations for production systems.

3.1.5 Business And Computer Science

Course Code: 3.1.5 — 10595536 Disciplinary Area: SECS-P/07

The course covers the functional models and information structures of business processes, focusing on the integration needs and data control requirements within corporate information systems. It addresses the systems and technologies necessary to develop and maintain successful e-businesses, along with the main application areas of information systems across various industries. Students learn about ICT organizational structures and common models for evaluating IT quality. Additionally, the course provides foundational knowledge of key concepts, methodologies, and essential soft skills required for effective management of ICT projects.

In applied terms, students gain the ability to analyze diverse categories of requirements gathered from different stakeholders in information systems. They learn to select appropriate models for evaluating the quality of ICT processes, software products, and services. The curriculum includes developing feasibility studies through choosing suitable application and technological architectures. It also covers preparing development plans for simple ICT projects while managing constraints of time, cost, and quality, and establishing the right organizational setup. Monitoring and controlling ongoing ICT projects through project management techniques is taught, as well as understanding and

evaluating lessons learned from previous projects.

3.1.6 Earthquake Physics and Machine Learning

Course Code: 3.1.6 — 10603321 Disciplinary Area: GEO/10

This course will provide a comprehensive summary of how machine learning (ML) is being used to predict frictional failure events, the lab equivalent of earthquakes, and improve our understanding of the physics of catastrophic earthquakes on tectonic faults. We will review the recent works, over the past few years, that solved a 50+ year old dilemma of how to predict the time to failure of lab earthquakes using laboratory seismic data.

These works show: 1) that ML can predict the timing and magnitude of labquakes using acoustic emissions (AE) that originate in the lab fault zone, 2) that in addition to passive measurements of lab AE, active source measurements of changes in fault zone elastic properties during the lab seismic cycle can be used to predict lab earthquakes, and 3) that the lab-based work can be applied to tectonic faulting in at least special cases if not more generally.

Recent work shows that labquakes are preceded by a cascade of AE events and systematic changes in elastic wave speed and transmitted amplitude that foretell catastrophic failure. As in all areas of AI, the methods being used for this problem evolve rapidly. We will discuss current techniques and traditional ML techniques based on regression.

A primary goal of the course will be to provide the scientific background needed to understand tenets of earthquake physics while introducing students to ML/DL methods to predict failure time and magnitudes. We will also study DL-based methods to autoregressively forecast labquakes and fault zone shear stress. Students will learn how to use lab seismic data and also how to access earthquake data from regional networks in Italy and worldwide.

Our studies of earthquake physics will include the evolution of frequency magnitude statistics during the lab seismic cycle, which provides an opportunity to use ML to interrogate the physics of impending failure. We will also see how precursors to lab earthquakes, that can be identified by AI, provide a sensible connection between the ML-based predictions, based on AE, and the physics of failure. In the lab, AE events represent a form of foreshock and, not surprisingly, the rate of foreshock activity correlates with fault slip rate and its acceleration toward failure. A central theme of the course will be to learn how lab earthquake prediction can improve forecasts of earthquake precursors and tectonic faulting.

As a research-style final project, I built Hypocenter Prediction Using PINN, FNO, and PINO, a comparative study that enforces the Eikonal equation inside physics-informed neural networks while also training Fourier and physics-informed neural operators. The models ingest receiver travel-time fields to localise 3D hypocenters, integrate domain constraints for data efficiency, and illustrate how neural-operator architectures accelerate seismic inversion across heterogeneous velocity models.

3.2 Second Semester

3.2.1 Human And Computer Interaction

Course Code: 3.2.1 — 10595535 Disciplinary Area: INF/01

The course explores both the theoretical foundations and practical methodologies of Human-Computer Interaction (HCI). It begins by introducing key models and theories, including cognitive models, models of communication and collaboration, task analysis, dialogue notation and design, system modeling, advanced interaction paradigms, and the social and collaborative aspects of interaction.

A major focus is placed on the integration of HCI within the software development lifecycle, emphasizing user-centered design, iterative development, scenario-based design, evaluation techniques, and the synergy with agile methodologies.

Topics covered include need finding, interviews and questionnaires, storyboarding, expert-based and user-based interface evaluation techniques, paper prototyping, and the full development process of a user interface. The course also addresses agile user-centered design, with iterative cycles of interaction design, software development, and evaluation. Specific attention is given to mobile interaction, with design criteria for interfaces on iOS and Android platforms. Toward the end of the course, students are assigned and discuss their final project topics.

My final project, FITM8, followed the full human-centered design loop (needfinding, synthesis, prototyping, user evaluation, deployment) to deliver a React Native + Expo fitness companion backed by Supabase (auth, data, realtime chat). The app lets runners plan workouts, discover community events, and coordinate via group or 1:1 messaging while maintaining typed navigation, shared state (hooks/Context), themed UI components, and automated Jest tests.

3.3 Internship

3.3.1 Assistant Researcher Internship

Course Code: AAF1466 — 12 ECTS Disciplinary Area: INF/01

This internal internship at Sapienza University of Rome spanned 105 days and focused on assistant research within the Department of Computer Science. Acting as the lead engineer on the project, I conducted applied research in explainable AI for financial markets, covering credit-risk scoring and algorithmic-trading strategies. The work combined data engineering, model governance, and stakeholder reporting with rigorous experimentation.

The outcome is the paper and reproducible artifact *Bridging Black Boxes*, which benchmarks model-agnostic and model-specific XAI techniques: SHAP variants (TreeSHAP, KernelSHAP, LinearSHAP, DeepSHAP), LIME, global/local surrogates, Anchors, counterfactual explanations, saliency and attention maps, and concept-based approaches such as TCAV and ProtoPNet. The study contrasts fidelity, stability, and regulatory suitability, providing guidance for risk officers and quants evaluating explainability stacks in production.

3.4 Final Examination

3.4.1 Final Exam

Course Code: AAF2011 — 6 ECTS Disciplinary Area: INF/01

The final examination consisted of the presentation and discussion of the research project *Bridging Black Boxes* before an academic commission.

The evaluation focused on the ability to clearly communicate the methodology, experimental design, and results of the work, as well as to justify technical choices and critically discuss the strengths and limitations of the implemented explainable AI techniques in financial applications.