COMPUTER ENGINEERING (LM55)

(Lecce - Università degli Studi)

Teaching DATA MINING & MACHINE LEARNING

GenCod A006449

Owner professor Massimo CAFARO

<b>Teaching in italian</b> DATA MINING & MACHINE LEARNING

<b>Teaching</b> DATA MINING & MACHINE LEARNING

SSD code ING-INF/05

Reference course COMPUTER ENGINEERING

Course type Laurea Magistrale

Credits 9.0

Teaching hours Front activity hours: 81.0

For enrolled in 2021/2022

Taught in 2022/2023

Course year 2

Language ENGLISH

Curriculum ARTIFICIAL INTELLIGENCE

Location Lecce

Semester First Semester

Exam type Oral

Assessment Final grade

Course timetable https://easyroom.unisalento.it/Orario

BRIEF COURSE DESCRIPTION

The course provides a modern introduction to data mining, which spans techniques, algorithms and methodologies for discovering structure, patterns and relationships in data sets (typically, large ones) and making predictions. Applications of data mining are already happening all around us, and, when they are done well, sometimes they even go unnoticed. For instance, how does the Google web search work? How does Netflix recommend movies to its users? The principles of data mining provide answers to these and others questions. Data mining overlaps the fields of computer science, statistical machine learning and data bases. The course aims at providing the students with the knowldedge required to explore, analyze and leverage available data in order to turn the data into valuable and actionable information for a company, for instance, in order to facilitate a decision-making process.

REQUIREMENTS

Calculus. Probability theory. Linear Algebra. Programming skills.

COURSE AIMS

Knowledge and understanding. The course describes methods and models for the analysis of large amounts of data. Students must have a solid background with a broad spectrum of basic knowledge related to data mining:

- the students must have the basic cognitive tools to think analytically, creatively, critically and in an inquiring way, and have the abstraction and problem-solving skills needed to cope with complex systems;
- they must have solid knowledge of data mining models and methodologies;
- they must be able to work on large data collections, including heterogeneous and produced at high speed data, in order to integrate them - in particular by knowing how to manage their origin and quality - and to carry out in-depth thematic analyses, drawing on this knowledge to improve the decision-making process.

Applying knowledge and understanding. After the course the student should be able to:

- describe and use the main data mining techniques;
- understand the differences among several algorithms solving the same problem and recognize which one is better under different conditions;
- tackle new data mining problems by selecting the appropriate methods and justifying his/her choices;
- tackle new data mining problems by designing suitable algorithms and evaluating the results;
- explain experimental results to people outside of statistical machine learning or computer science.

Making judgements. Students must have the ability to process complex and/or fragmentary data and must arrive at original and autonomous ideas and judgments, and consistent choices in the context of their work, which are particularly delicate in the profession of data scientist. The course promotes the development of independent judgment in the appropriate choice of technique/model for data processing and the critical ability to interpret the goodness of the results of the models/methods applied to the datasets under examination.

Communication. It is essential that students are able to communicate with a varied and composite audience, not culturally homogeneous, in a clear, logical and effective way, using the methodological tools acquired and their scientific knowledge and, in particular, the specialty vocabulary. Students should be able to organize effective dissemination and study material through the most common presentation tools, including computer-based ones, to communicate the results of data analysis processes, for example by using visualization and reporting tools aimed at different types of audiences.

Learning skills. Students must acquire the critical ability to relate, with originality and autonomy, to the typical problems of data mining and, in general, cultural issues related to other similar areas. They should be able to develop and apply independently the knowledge and methods learnt with a view to possible continuation of studies at higher (doctoral) level or in the broader perspective of cultural and professional self-improvement of lifelong learning. Therefore, students should be able to switch to exhibition forms other than the source texts in order to memorize, summarize for themselves and for others, and disseminate scientific knowledge.

TEACHING METHODOLOGY

The course aims to provide students with advanced tools for data analysis, through which to extrapolate relevant information from large datasets and guide the related decision-making processes. The course consists of frontal lessons using slides made available to students via the UniSalento e-learning platform, and classroom exercises. The frontal lessons are aimed at improving students' knowledge and understanding through the presentation of theories, models and methods; students are invited to participate in the lesson with autonomy of judgement, by asking questions and presenting examples. The exercises are aimed at understanding the algorithms and models presented.

ASSESSMENT TYPE

Software project and oral exam. During the exam the student is asked to illustrate theoretical topics in order to verify his/her knowledge and understanding of the selected topics. The student must demonstrate adequate knowledge and understanding of the issues presented or indicated, applying in a relevant manner the theories and conceptual models covered by the study programme. The oral exam is evaluated on a scale from 18 to 30 points. The software project is assigned upon request to the student, and must be mandatorily discussed into the same trial in which the oral test is performed. The software project is evaluated on a scale from 18 to 30 points. The final grade is given by the average of the grades obtained, and the exam is passed if the grade achieved is at least 18.

The report on the assigned project must be structured as follows.

1. Introduction. The student must provide an accurate description of the assigned project, including an analysis of the sequential algorithm that solves the problem addressed in the project. If the student deems it important for understanding, then pseudo-code, examples, graphs, figures, application instances etc. may be included;

2. Implementation. The software must be implemented using the C or, possibly, C++ programming language (so as to use the data structures present in the C++ STL). The code must necessarily be appropriately commented. In the event that the student verifies the existence of multiple solution strategies for the same assigned problem, an implementation can be provided for each strategy, discussing the advantages and disadvantages of each strategy. Note that the code does not have to be given in full in the text of the report, as the project code - source and Makefile for the build (alternatively CMake can be used) - must in any case be delivered separately. However, the report may include some code snippets relating to the most critical and interesting parts.

3. Debugging and testing. It is recommended to debug and test the code in order to reasonably verify the absence of bugs and the correctness of the algorithm in relation to the output produced. The student is urged to design and test unit tests deemed suitable for assessing the goodness of the code. The student is explicitly warned that the implementation of an incorrect solving strategy and/or the presence of bugs that crash the application at runtime will result in failing the exam, while the presence of bugs that affect the correctness of the algorithm will result in a penalty relative to the mark that will be awarded.

4. Performance and scalability analysis. The student must analyse the performance of the implementation developed in terms of execution time and memory occupation if necessary. The scalability of the algorithm as the problem size changes must be assessed.

5. Synthesis. The report must be as concise as possible, but without detriment to clarity of presentation.

6. Project evaluation. The project will be assessed on a scale of 1 to 30 points. The grade will only be determined at the end of the project discussion. The complexity of the problem assigned will be taken into account in the assessment; students dealing with simpler problems should therefore expect less leniency in the assessment than students dealing with more difficult problems.

Assessment will also take into account:

- Clarity and effectiveness of presentation;
- Technical skills and documentation of implementation;
- Critical thinking in evaluation and performance analysis;

- Meaningfulness of results: since performance is the main purpose, a good project must also deliver meaningful performance.

7. Plagiarism. The student is explicitly warned that plagiarism is very serious, and is taken very seriously. The use of external sources of any kind (internet, books, handouts, previous work by other students etc.) is only allowed for marginal contributions in the project (e.g. for the initial description of the assigned project), and each individual occurrence must be explicitly cited and reported in the report. Violation of this code of conduct will not be tolerated in any way, and will result in the immediate cancellation of the examination and the referral of the student to the University Disciplinary Committee, with the commencement of the relevant disciplinary proceedings in accordance with the provisions of Title V (DISCIPLINARY PROCEDIMENTS AND SANCTIONS) of the University Regulations for Students (D.R. 672 of 5/12/2017, which came into force on 8/12/2017).

OTHER USEFUL INFORMATION

Office Hours
By appointment; contact the instructor by email or at the end of class meetings.

FULL SYLLABUS

Introduction to the course.

Introduction to the Map-Reduce parallel programming model. Open-source Hadoop implementation. Pros and cons of Hadoop and Map-Reduce. Distributed File System. Chunk servers, Master node. Map Function. Sort and Shuffle. Reduce Function. Map Tasks. Reduce Tasks. Word counting. Handling failures. Number of Map and Reduce jobs. Granularity of tasks and pipelining. Strugglers tasks: mitigating the problem by spawning backup tasks. Combiners. Partition (hash) function. Additional examples of Map-Reduce: natural join, two-pass matrix multiply, single pass matrix multiply. Cost measures for Map-Reduce algorithms.

Frequent Pattern Mining. Discovery of association rules. Market-basket model. Examples of possible applications. Frequent itemsets. Support of an itemset. Association rules. Confidence and Interest. Association rules with highly positive or negative interest. Mining association rules. Maximal and closed frequent itemsets. Lattice of the itemsets. Naive approach to counting frequent pairs. A-priori. Monotonicity. PCY. PCY refinements: multistage and multihash. Frequent itemsets in 2 passes: random sampling and choice of the threshold, SON, monotonicity, parallel SON parallelo with Map-Reduce in 2 passes, Toivonen and the negative border. ECLAT. DECLAT. FPGrowth. Sequence mining. Frequent sequences. Mining frequent sequences. GSP. SPADE. PREFIXSPAN.

Streams. Uniform, 2-universal and pairwise independent hash function. Streaming: turnstile, strict turnstile and cash register models. Frequency estimation. Sketches. Count-Sketch. Count-Min. Comparing Count-Sketch and Count-Min. Frequent items. Phi-frequent items. The majority problem. Boyer-Moore. Misra-Gries. Frequent. Space Saving. Space Saving properties. Comparing Frequent and Space Saving. Sampling from a Data Stream: Sampling a fixed proportion. Sampling from a Data Stream: Sampling a fixed-size sample (Reserved Sampling). Queries over a sliding windows: counting the number of bits equal to 1 with DGIM. Filtering data streams: Bloom FIlters. Counting distinct elements: Flajolet-Martin. Estimating moments with AMS.

Scene completion problem. Near neighbors in high dimensionality spaces. Document similarity. Pairs of candidate documents. Near neighbor search. Jaccard similarity and distance. Shingling: how to convert documents, emails etc in sets. k-shingles. Compression through hashing of k-shingles. Min-Hashing: converting big sets in short signatures preserving the similarity. Similarity and Jaccard distance for boolean vectors. Boolean matrices. Min-hash signatures. Implementation. Locality-Sensitive Hashing: determining pairs of candidate documents. Matrix partitioning in b bands of r rows: analysis of accuracy with regard to false positives and false negatives.

Link analysis. PageRank. Dead ends. Spider traps. Flow formulation. Matrix formulation. Random walk interpretation. Stationary distribution of a Discrete-Time Markov Chain. Perron-Frobenius Theorem. Google matrix and teleportation. Sparse matrix encoding. Block update algorithm. Topic-specific PageRank. Matrix formulation. Topic vector. Web Spam. Term spam. Spam farms. PageRank value obtained through a Spam Farm. TrustRank. Trust propagation. Spam Mass estimation.

Recommender systems. Recommendations. The long tail phenomenon. Content-based systems. Utility function and matrix. Ratings. Extrapolation of ratings (utilities). Item profiles. User profiles. Collaborative filtering. k-NN. Similarity metrics. User-user and item-item collaborative filtering. Evaluation of systems. Error metrics. RMSE, precision, rank correlation. Complexity of collaborative filtering. The Netflix challenge. Bellkor recommender system. Modeling local and global effects. Learning the optimal weights: optimization problem and gradient descent. Latent factor models. SVD decomposition. Learning the P and Q matrices. Preventing overfitting: regularization. Stochastic Gradient Descent. Biases and interactions. Temporal biases and factors.

Artificial Intelligence. Different definitions over time: thinking like a human, acting like humans and the Turing test, thinking rationally, acting rationally. Rational agents, decisions and behaviour. Game playing: chess, go, etc. The human intelligence. Human versus artificial intelligence. Benefits of AI. Ai, machine learning and deep learning. AI applications and use-cases. Human learning process. Features, instances, labels, classes, classification function. Training set, validation and test set. Supervised learning: regression, binary and multiclass classification. Unsupervised learning. Semi-supervised and weakly-supervised learning. Reinforcement learning: agent, environment and its state that changes depending on the agent's actions, reward and penalty. Loss functions. Zero-one and squared loss functions. Bias and variance. Deep Learning. Machine Learning vs Deep Learning. Limitations of Deep Learning. Artificial versus biological neural networks. AI Myths.

The clustering problem. Curse of dimensionality. Clustering in euclidean and non euclidean spaces. Distances. Hierarchical clustering: agglomerative and divisive algorithms. Clustering by point assignment. Centroid and clustroid. K-means and K-means++. Choosing k: elbow criterion. BFR. Discard, Compression and Retained sets. Summarizing points. Mahalanobis distance. CURE. Representative points and their choice. Input space and feature space. Kernel methods. Kernel matrix. Linear kernel. Kernel trick. Kernel operations in feature space. Represenative clustering: K-means and Kernel K-means. Expectation-Maximization clustering. Hierarchical clustering. Density-based clustering. DBSCAN. Clustering validation: external, internal and relative measures.

Machine learning. Supervised and unsupervised approaches. Numerical and categorical attributes. Categorical attributes: nominal and ordinal attributes. Probabilistic classifiers. Parametric approach: Bayes and naive Bayes classifiers. Data centering. Non parametric approach (density based): K-nearest neighbors classifier. Decision Trees. Hyperplans. Split points. Data partion and purity. Split Point Evaluation Measures: entropy, split entropy, information gain, Gini index, CART. Evaluation of numerical and categorical split points. Support Vector machines. Hyperplanes. Support Vectors and Margins. Linear and Separable Case. Soft Margin SVM: Linear and Nonseparable Case. Kernel SVM: Nonlinear Case. SVM Training Algorithms. Multiclass SVM. Assessing the performances of a classifier. Evaluation metrics. ROC curve and AUC. K-fold cross-validation. Bootstrapping. Confidence intervals. Paired t-Test. Bias and variance decomposition. Ensemble classifiers. Bagging. Random Forest. Boosting. Stacking.

Introduction to neural networks. History. The Biological Inspiration. Learning in Biological vs Artificial Networks. An Alternative View: The Computational Graph Extension of Traditional Machine Learning. Machine Learning versus Deep Learning. Single Layer Networks: the Perceptron. Bias neurons. Training a Perceptron. Perceptron vs Linear SVMs. Activation and Loss Functions. Multilayer Neural Networks. Reasons for nonlinearity of hidden layers. Role of Hidden Layers. The Feature Engineering View of Hidden Layers. Multilayer Networks as Computational Graphs. Connecting Machine Learning with Shallow Neural Networks: Perceptron versus Linear SVM, RBF Network versus kernel SVM. Neural models for linear regression, classification and the Fisher discriminant. Widrow-Hoff learning. Neural model for linear regression. Comparison of Widrow-Hoff with Perceptron and SVM. Connections with Fisher Discriminant. Neural Models for Logistic Regression. The Softmax Activation Function and Multinomial Logistic Regression. The Autoencoder for Unsupervised Representation Learning. Singular Value Decomposition with Autoencoders. Row-Index to Row-Value Autoencoders: Incomplete Matrix Factorization for Recommender Systems. Model Generalization and the Bias-Variance Trade-Off in the context of neural networks.
Penalty-Based Regularization. Economy in Parameters. Soft vs hard economy. L1 vs L2 regularization. L2 regularization and noise injection. Weight sharing. Dropout. Feature co-adaptation. Why feature co-adaptation is bad. Dropout training. Dropout testing: repeated sampling and geometric averaging vs weight scaling inference rule. Why dropout helps. Connection of

REFERENCE TEXT BOOKS	Mining of Massive Datasets J. Leskovec, A. Rajaraman and J. Ullman Freely availableonline: http://www.mmds.org Data Mining and Analysis M. J. Zaki and W. Meira Freely available online: http://dataminingbook.info Neural Networks and Deep Learning Charu C. Aggarwal Springer