COMPUTER ENGINEERING (LM75)

(Lecce - Università degli Studi)

Teaching DATA MINING & MACHINE LEARNING

GenCod A006449

Owner professor Massimo CAFARO

Teaching in italian DATA MINING & MACHINE LEARNING

Teaching 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 2022/2023

Taught in 2023/2024

Course year 2

Language ENGLISH

Curriculum Intelligenza artificiale

Location Lecce

Semester First Semester

Exam type Oral

Assessment Final grade

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

BRIEF COURSE DESCRIPTION

Il corso fornisce un'introduzione moderna al data mining ed al machine learning, che comprendono tecniche, algoritmi e metodologie per scoprire strutture, schemi e relazioni in insiemi di dati (tipicamente di grandi dimensioni) e fare previsioni. Le applicazioni del data mining e del machine learning sono già presenti intorno a noi e, quando sono fatte bene, a volte passano addirittura inosservate. Per esempio, come funziona la ricerca web di Google? Come fa Netflix a consigliare i film ai suoi utenti? I principi del data mining e del machine learning forniscono risposte a queste e altre domande. Il data mining ed il machine learning intersecano i campi dell'informatica, dell'apprendimento statistico e delle basi di dati. Il corso mira a fornire agli studenti le conoscenze necessarie per esplorare, analizzare e sfruttare i dati disponibili al fine di trasformarli in informazioni preziose e utilizzabili per un'azienda, ad esempio per facilitare il processo decisionale.

REQUIREMENTS

Analisi Matematica I e II. Teoria della probabilità. Algebra lineare. Capacità di programmazione in linguaggio C/C++.

COURSE AIMS

Knowledge and understanding. Il corso descrive metodi e modelli per l'analisi di grandi quantità di dati. Gli studenti devono avere un solido background con un ampio spettro di conoscenze di base relative al data mining:

- gli studenti devono possedere gli strumenti cognitivi di base per pensare in modo analitico, creativo, critico e curioso, e possedere le capacità di astrazione e di risoluzione dei problemi necessarie per affrontare sistemi complessi;
- devono avere una solida conoscenza dei modelli e delle metodologie di data mining;
- devono essere in grado di lavorare su grandi raccolte di dati, anche eterogenei e prodotti ad alta velocità, al fine di integrarli - in particolare sapendo gestirne l'origine e la qualità - e di effettuare analisi tematiche approfondite, attingendo a queste conoscenze per migliorare il processo decisionale.

Applying knowledge and understanding. Al termine del corso lo studente dovrà essere in grado di:

- descrivere e utilizzare le principali tecniche di data mining;
- comprendere le differenze tra diversi algoritmi che risolvono lo stesso problema e riconoscere quale sia il migliore in diverse condizioni;
- affrontare nuovi problemi di data mining selezionando i metodi appropriati e giustificando le proprie scelte;
- affrontare nuovi problemi di data mining progettando algoritmi adeguati e valutando i risultati;
- spiegare i risultati sperimentali a persone estranee al machine learning o all'informatica.

Making judgements. Lo studente deve avere la capacità di elaborare dati complessi e/o frammentari e deve giungere a idee e giudizi originali e autonomi, nonché a scelte coerenti nel contesto del proprio lavoro, particolarmente delicate nella professione del data scientist. Il corso promuove lo sviluppo dell'autonomia di giudizio nella scelta appropriata di tecniche/modelli per l'elaborazione dei dati e la capacità critica di interpretare la bontà dei risultati dei modelli/metodi applicati ai dataset in esame.

Communication. È fondamentale che gli studenti siano in grado di comunicare con un pubblico vario e composito, non culturalmente omogeneo, in modo chiaro, logico ed efficace, utilizzando gli strumenti metodologici acquisiti e le loro conoscenze scientifiche e, in particolare, il lessico specialistico. Gli studenti devono essere in grado di organizzare materiale divulgativo e di studio efficacemente, attraverso i più comuni strumenti di presentazione anche informatici, per comunicare i risultati dei processi di analisi dei dati, ad esempio utilizzando strumenti di visualizzazione e reportistica rivolti a diversi tipi di pubblico.

Learning skills. Gli studenti devono acquisire la capacità critica di rapportarsi, con originalità e autonomia, alle problematiche tipiche del data mining e, in generale, alle questioni culturali legate ad altri ambiti simili. Dovranno essere in grado di sviluppare e applicare autonomamente le conoscenze e i metodi appresi in vista di un eventuale proseguimento degli studi a livello superiore (dottorato) o nella più ampia prospettiva di auto-miglioramento culturale e professionale dell'apprendimento permanente. Pertanto, gli studenti devono essere in grado di passare a forme espositive diverse dai testi di partenza per memorizzare, riassumere per sé e per gli altri e diffondere le conoscenze scientifiche.

TEACHING METHODOLOGY

Il corso si propone di fornire agli studenti strumenti avanzati per l'analisi dei dati, attraverso i quali estrapolare informazioni rilevanti da grandi insiemi di dati e guidare i relativi processi decisionali. Il corso si articola in lezioni frontali, con l'ausilio di slide messe a disposizione degli studenti tramite la piattaforma e-learning di UniSalento, ed esercitazioni in aula. Le lezioni frontali mirano a migliorare la conoscenza e la comprensione degli studenti attraverso la presentazione di teorie, modelli e metodi; gli studenti sono invitati a partecipare alla lezione con autonomia di giudizio, ponendo domande e presentando esempi. Le esercitazioni sono finalizzate alla comprensione degli algoritmi e dei modelli presentati.

ASSESSMENT TYPE

Progetto software ed esame orale. Durante l'esame viene chiesto allo studente di illustrare argomenti teorici per verificare la sua conoscenza e comprensione degli argomenti selezionati. Lo studente deve dimostrare un'adeguata conoscenza e comprensione degli argomenti presentati o indicati, applicando in modo pertinente le teorie e i modelli concettuali oggetto del programma di studio. L'esame orale è valutato su una scala da 18 a 30 punti. Il progetto software viene assegnato su richiesta dello studente e deve essere obbligatoriamente discusso nella stessa prova in cui si svolge l'esame orale. A tal fine, il progetto (codice e relativa documentazione) deve essere inviato al docente via email almeno tre giorni prima dell'esame. Non saranno accettati progetti inviati oltre il termine indicato. Il progetto software è valutato su una scala da 18 a 30 punti. Il voto finale è dato dalla media dei voti ottenuti, e l'esame è superato se il voto conseguito è almeno pari a 18.

La relazione relativa al progetto assegnato deve essere strutturata come segue.

1. Introduzione. Lo studente deve fornire una descrizione accurata del progetto assegnato, comprensiva dell'analisi dell'algoritmo sequenziale che risolve il problema affrontato nel progetto. Qualora lo studente lo ritenga importante ai fini della comprensione, possono essere presenti pseudo-codice, esempi, grafici, figure, istanze applicative etc;

2. Implementazione. Il software deve essere implementato utilizzando il linguaggio di programmazione C o, eventualmente, C++ (in modo da usare le strutture dati presenti nella STL del C++). Il codice deve essere necessariamente commentato in modo adeguato. Nel caso in cui lo studente verifichi l'esistenza di strategie multiple per lo stesso problema assegnato, è possibile fornire una implementazione per ciascuna strategia, discutendo vantaggi e svantaggi di ogni strategia. Si noti che il codice non deve essere riportato integralmente nel testo della relazione, in quanto il codice del progetto - sorgenti e Makefile per il build (in alternativa è possibile usare CMake) - deve in ogni caso essere consegnato a parte. Tuttavia, la relazione può includere alcuni snippet di codice relativi alla parti più critiche ed interessanti.

3. Debug e test. Si raccomanda di effettuare un debug e test del codice che consenta di verificare, ragionevolmente, l'assenza di bugs e la correttezza dell'algoritmo in relazione all'output prodotto. Si invita lo studente a progettare e verificare gli unit tests ritenuti idonei a valutare la bontà del codice. Lo studente è esplicitamente avvisato che l'implementazione di una strategia risolutiva non corretta e/o la presenza di bugs che mandano in crash l'applicazione a runtime comportano il non superamento dell'esame, mentre la presenza di bugs che inficiano la correttezza dell'algoritmo comporta una penalizzazione relativa al punteggio che sarà attribuito.

4. Analisi delle prestazioni e della scalabilità. Lo studente deve analizzare le prestazioni dell'implementazione sviluppata in termini di tempo di esecuzione ed occupazione di memoria se necessario. Deve essere valutata la scalabilità dell'algoritmo al variare della problem size.

5. Sintesi. La relazione deve essere quanto più possibile sintetica, ma senza che ciò vada a discapito della chiarezza espositiva.

6. Valutazione del progetto. Il progetto sarà valutato con un punteggio relativo ad una scala che va da 1 a 30 punti. Il voto sarà definito solo al termine della discussione del progetto. La complessità del problema assegnato sarà presa in considerazione per la valutazione; gli studenti che si occupano di problemi più semplici devono quindi aspettarsi una minore indulgenza nella valutazione rispetto agli studenti che si occupano di problemi più difficili.

La valutazione terrà inoltre conto di:

- Chiarezza ed efficacia della presentazione;
- Capacità tecniche e documentazione dell'implementazione;
- Pensiero critico nella valutazione e nell'analisi delle prestazioni;

- Significatività dei risultati: dato che le prestazioni sono lo scopo principale, un buon progetto deve anche fornire prestazioni significative.

7. Plagio. Lo studente è avvisato esplicitamente che il plagio è gravissimo, ed è considerato molto seriamente. L'uso di fonti esterne di qualsiasi genere (internet, libri, dispense, lavori pregressi di altri studenti etc) è consentito solo per contributi marginali nel progetto (ad esempio, per la descrizione iniziale del progetto assegnato), ed ogni singola occorrenza deve essere esplicitamente citata e riportata nella relazione. La violazione di questo codice di comportamento non sarà in alcun modo tollerata, e comporta l'immediato annullamento dell'esame ed il deferimento dello studente alla commissione disciplinare di ateneo, con avvio del relativo procedimento disciplinare secondo quanto riportato del Titolo V (PROCEDIMENTI DISCIPLINARI E SANZIONI) del Regolamento di Ateneo per gli studenti (D.R. 672 del 5/12/2017, entrato in vigore in data 8/12/2017).

OTHER USEFUL INFORMATION

Ricevimento studenti
Su appuntamento; contattare il docente via e-mail o al termine degli incontri di classe.

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 a fixed-size sample: varying the sample size. Reservoir 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.

REFERENCE TEXT BOOKS

Mining of Massive Datasets, 3rd edition
J. Leskovec, A. Rajaraman and J. Ullman
Freely available online: http://www.mmds.org

Data Mining and Analysis, 2nd edition
M. J. Zaki and W. Meira
Freely available online: http://dataminingbook.info