POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Matematica

Tesi di Laurea Magistrale

Caratterizzazione e analisi di dati di performance in un'ottica di genere: caso di studio nel contesto delle discipline STEM

Relatore/i

prof. Tania Cerquitelli

Candidato

Maura Podda

Anno Accademico 2020-2021

Ai miei genitori

Sommario

Nonostante studi, risorse e impegni presi per incoraggiare le donne ad entrare nel mondo scientifico, ancora persistono numerose problematiche sia salariali sia di carriera. Concentrandosi sulla carriera universitaria, questo studio si propone di analizzare quali variabili influenzano maggiormente il voto finale delle persone laureate, e se l'identità di genere ha delle influenze o meno. Nell'attività descritta in questa tesi di laurea magistrale si analizzando un set di dati relativi ai risultati di diploma, test di ammissione e alcune informazioni relative alla carriera di un gruppo di persone laureate nel periodo 2015-2019 in una università nelle discipline STEM. Il processo di analisi comprende la pulizia dei dati, una prima fase di esplorazione e caratterizzazione delle distribuzioni in termini di metriche statistiche (e.g., medie, varianze) nonchè l'utilizzo di vari algoritmi di data mining (e.g., alberi di decisione, random forest, SVM e regressione lineare) per la modellazione della relazioni principali tra le variabili di input (quali ad esempio il percorso di formazione, i risultati di performance, e alcune caratteristiche della popolazione sotto osservazione) e il voto conseguito alla laurea triennale. Le performance dei modelli predittivi, stimati valutando un indice statistico (e.g. R2), non sono ottimali tuttavia evidenziano che la variabile genere è tra quelle che influenzano la predizione. Per migliorare le performance degli algoritmi testati si suggerisce di estendere lo studio con un set di variabili aggiuntivi in grado di rappresentare correttamente le relazioni principali che esistono tra percorsi di studi e performance in un'ottica di genere.

Indice

Introduzione generale

Le donne nel mondo del lavoro tendono a lavorare meno ore, lavorano in settori scarsamente retribuiti e occupano posizioni di rango inferiore rispetto agli uomini, il che determina considerevoli divari retributivi tra i generi. Tali differenze sono dovute, in una certa misura, alla radicazione profonda dei ruoli di genere tradizionali, ma anche agli incentivi economici.

Nonostante studi, risorse e impegni presi per incoraggiare le donne ad entrare nel mondo scientifico, ancora persistono numerose problematiche sia nell'ingresso nel mondo del lavoro sia nel fare carriera. Secondo [1], solo il 30% dei ricercatori sono donne e la percentuale di autrici di articoli scientifici nella maggior parte dei campi è ancora minore, e, esaminando solo quelle che continuano a pubblicare per molti anni il numero decresce molto più rapidamente dei colleghi uomini.

Questi problemi non si manifestano solo a livello lavorativo ma cominciano generalmente a livello formativo. Mentre molti studi su questo argomento si concentrano sull'attività delle scuole primarie e secondarie, le università non sono sicuramente esenti da criticismo.

Questa tesi si propone di esaminare i risultati conseguiti dalle persone che si sono laureate in una università italiana e verificare se il risultato finale è influenzato dal genere. I dati sono stati prima selezionati e resi omogenei, e una prima analisi è stata fatta confrontando le statistiche sui dati relativi alla votazione conseguita nell'esame di diploma e nei test di ammissione in Ateneo.

In seguito, abbiamo scelto alcuni modelli sia categorici sia continui per tentare di prevedere i risultati finali utilizzando oltre ai risultati precedenti anche i dati anagrafici come la regione di appartenenza e informazioni sulla carriera universitaria come il tempo necessario per laurearsi e il collegio di laurea.

Nel primo e secondo capitolo vengono presentati i risultati ottenuti da tre studi precedenti e la data science che si utilizzerà nella nostra analisi.

Nel terzo capitolo verranno introdotte le librerie di python utili all'analisi dei dati e alla creazione di modelli.

Nel quarto capitolo viene analizzato il procedimento che abbiamo utilizzato per estrarre le informazioni, la pipeline che permette di ottenere modelli. Vengono quindi esaminati i risultati, commentando questi ultimi e suggerendo possibili ulteriori studi per la elaborazione di modelli più accurati.

1 Letteratura

Ci sono molte differenze di genere nel campo scientifico: il più discusso è nel mondo del lavoro: “A Global Approach to the Gender Gap in Mathematical, Computing, and Natural Science How to Measure It, How to Reduce It?” [1] lo dimostra utilizzando sia dati ottenuti mediante un questionario sia attraverso l’analisi degli articoli di tre diversi archivi.

Nel primo caso, sono stati raccolti dati da partecipanti in 157 paesi e diverse discipline, selezionati usano un metodo a “palla di neve” (I partecipanti sono incoraggiati a contattare i loro conoscenti per l’adesione al sondaggio). Questo metodo non ha prodotto un campione statisticamente rappresentativo, causando limitazioni sull’analisi dati ottenute e all’informazione ottenuta.

Sono state utilizzate le tecniche della statistica multivariata per estrarre informazioni, nel particolare la regressione logistica e la regressione logistica ordinale per approssimare il p-value e la correzione di Bonferroni per escludere errori del primo tipo.

In aggiunta ai dati anagrafici e dell’ambito di studi e lavoro, nel questionario vi sono domande su come e quando si è scelto di studiare nel campo STEM, livello di studi, incoraggiamento agli studi ed esperienza universitaria e lavorativa (sia con risposta multipla sia con domande aperte).

Oltre ad avere uno stipendio medio minore, le donne dello studio hanno generalmente riscontrato un’esperienza peggiore rispetto ai loro colleghi: questa differenza si mostra specialmente per quanto riguarda l’essere stata soggetta a molestie sessuali. Si è riscontrato che le donne in ogni campo di studi o lavorativo e in ogni area geografica - eccetto l’asia occidentale - hanno una probabilità di subirlo tra cinque e venticinque volte superiore, anche escludendo il campo della storia della scienza a causa del basso numero di dati.

È stata rilevata anche un’alta percentuale di abbandono degli studi e generalmente un peggiore trattamento da parte di superiori e colleghi. Gli unici campi in cui le donne

hanno registrato una situazione migliore rispetto agli uomini sono l'incoraggiamento da parte della famiglia e la loro stessa determinazione. Le differenze aumentano dopo la nascita di un figlio: le donne subiscono più discriminazioni, la loro carriera viene influenzata maggiormente sia dalle relazioni sia dalle nascite. Questo viene riscontrato nella maggior parte dei campi e in tutte le aree geografiche, mentre l'inverso mai.

Nonostante il campione non statistico, le informazioni ottenute sembrano essere in linea con le scoperte di studi precedenti.

Nella seconda parte gli autori dello studio hanno analizzato le pubblicazioni di tre database e le tre branche di cui si occupano: SAO/ADS (astronomia e astrofisica), zbMATH (matematica pura e applicata) e arXiv (argomenti vari, ma in questo caso principalmente fisica). Altri database sono stati considerati per ulteriori informazioni. Multipli autori per lo stesso articolo vengono considerati in ordine di importanza a meno che non siano elencati ordine alfabetico (comune principalmente in matematica).

Nella maggior parte dei casi i database stessi contengono le informazioni bibliografiche della pubblicazione, ma spesso è necessario disambiguare il genere dell'autore: molto spesso le informazioni sono abbreviate ed è necessario interpolare i dati da altre fonti. Sono stati utilizzati tre metodi diversi per ciascun database: zbMATH contiene profili degli autori sufficienti alla nostra analisi, per ADS è stato creato un modello di machine learning seguito da euristiche, per arXiv si era già sviluppato un metodo basato sulle iniziali che era abbastanza accurato per gli scopi. Dopo questo processo, il genere dell'autore è stato identificato principalmente a partire dal nome tramite una serie di algoritmi sviluppati in precedenza. Le informazioni sull'area geografica sono state ottenute principalmente tramite le affiliazioni.

Vengono considerati vari aspetti delle pubblicazioni, ad esempio: percentuale di autori donna, "dropout rates" (stima della lunghezza della carriera), differenza di produttività e autori donna in pubblicazioni rinomate.

In generale, la percentuale di donne aumenta col tempo in tutti i tre gli argomenti, ma raggiunge al massimo 25-30% per matematica e astronomia, 16% in astrofisica e al massimo 20% in altri campi della fisica.

Per la lunghezza della carriera, è stata analizzata la percentuale di uomini e donne attivi da uno 1 a 10 anni dopo aver pubblicato articoli per 5 anni, raggruppati per la data della loro prima pubblicazione. In tutte le categorie e genere, questa diminuisce, illustrando una maggiore difficoltà nell'avere una carriera a lungo termine. Osservando per ogni materia entrambi i grafi, si mostra che le donne hanno percentuali maggiori di abbandono, le linee sono meno dritte e non formano un chiaro ordine cronologico pari a quelle maschili.

La differenza di produttività (numero di publicazioni), rappresentata come il rapporto tra le produttività delle donne rispetto agli uomini sembra essersi avvicinata a uno nelle fasi iniziali della carriera, ma continua a diminuisce nel corso degli anni in matematica e fisica ma in minor misura in astronomia/astrofisica. Questo è spiegato mostrando la produttività mediana, che evidenzia come la maggior parte degli "overperformer" sono uomini e la causa del divario e suggerisce che la maggiora collaborazione in astronomia (indicato dalla percentuale superiore di articoli con 4+ autori) che permette alle donne di forgiare le connessioni necessarie per competere.

Un'altra analisi importante è quella degli articoli in pubblicazioni considerate rinomate, stimato in due modi:

- è stato sia chiesto direttamente nel questionario esaminato nella sezione precedente se si possedeva un master o un dottorato in una delle discipline con più risposte quanti articoli e in caso positivo, quanti articoli hanno pubblicato in esse negli ultimi cinque anni (senza chiedere dove),
- si è scelta una serie di pubblicazioni nelle varie discipline per esaminare il loro contenuto.

Nel primo caso, donne e uomini hanno riportato una media simile, invece usando il test di Mann-Whitney U su tutte le risposte e sui gruppi si sono trovate differenze significative ma l'effetto è poco significativo. Inoltre, osservando le pubblicazioni, sia attraverso la regressione delle percentuali per ogni rivista, sia attraverso la differenza fra la media di donne autori e la percentuale di donne nella specializzazione lavorativa,

si vede che in astronomia e astrofisica procedono come la media totale, mentre in matematica e fisica teorica è bassa.

Queste differenze però tendono a mostrarsi anche prima, nell' esame di ammissione all'università. "Gender Bias in Standardized Tests: Evidence from a Centralized College Admissions System" [2] mostra che usare test standardizzati causa la selezione di studenti rispetto a studentesse con voti medi equivalenti o superiori.

I testi standardizzati o SAT sono usati in molti paesi e idealmente rappresentano la performance dello studente e permettono di trattare tutti come uguali, ma sono stati criticati da varie fonti sia come incoraggiamento verso un insegnamento ottimizzato sullo specifico test invece che sullo sviluppo delle capacità dello studente, sia come una metrica inferiore alla media scolastica o GPA.

Questo studio è stato sviluppato su un campione di dati delle applicazioni a studi universitari nel 2008 in Turchia contenente i risultati dei test standardizzati, il GPA, le università scelte e il risultato dell'esame di ammissione. Alcuni studenti hanno anche partecipato a un sondaggio a proposito della situazione socioeconomica della famiglia, i loro risultati nelle superiori, la loro opinione sulla loro educazione e l'uso di tutor privati.

L'ammissione all'università dipende principalmente dal test standardizzato: questo è composta in una parte qualitativa e una quantitativa, ciascuna divisa a sua volta in due: la seconda contiene domande più complesse ed è richiesta solo per alcune specializzazioni. Il GPA è calcolato con tre pesi diversi e aggiunti al risultato del test standardizzato per il percorso scolastico corrispondente, comunque ha un'influenza minore.

Osservando le funzioni cumulative, le donne si diplomavano con voti superiori e avevano un tasso di ammissione al primo tentativo maggiore. Anche i loro voti nel test standardizzato tendevano ad essere più alti - ma meno chiaramente del GPA - e questo vantaggio scompare condizionando su quest'ultimo. Questo non cambia osservando solo i risultati dei primi tentativi.

Per capire se il test e GPA vengono influenzati differentemente dal genere è stata fatta una regressione lineare rispetto a genere e altre variabili di controllo come tipo di scuola, carriera scolastica e città, e si rileva la differenza fra i coefficienti. Per quanto riguarda il GPA, le donne sembrano essere avvantaggiate come per ogni tipo di scuola e percorso scolastico. Usando una stima del quantile per stimare la differenza: in ogni percentile osserviamo che vi è un vantaggio nelle per le donne, specialmente selezionando solo i primi tentativi per l'esame.

Per quanto riguarda i test standardizzati, la maggior parte dei voti sembra avvantaggiare le donne, specialmente concentrandoci sui soli primi tentativi, ma condizionando rispetto al GPA questo risultato viene invertito, specialmente considerando i risultati che vengono considerati per università che portano a lavori meglio retribuiti.

Questo studio mostra un collegamento, ma non spiega perché vi è questa differenza. "Why Do Standardized Tests Underpredict Women's Academic Performance? The Role of Conscientiousness" [3] oltre a confermare questa differenza, tenta di spiegarla teorizzando che sia dovuta a una maggiore coscienziosità delle donne.

Tre gruppi di studenti che hanno frequentato tre diverse università americane (non necessariamente nello STEM) in tre diversi anni hanno compilato test di personalità contenenti sezioni sulla coscienziosità. In seguito, sono stati ottenuti i voti degli studenti nel test standardizzato e il GPA alla fine del primo anno accademico, tramite uffici amministrativi o tramite autocertificazione.

In tutte e tre i gruppi, le donne avevano SAT più bassi di quelli degli uomini, ma GPA più alti o equivalenti, e un livello più alto di coscienziosità. Per quantificare la tendenza a sottovalutare i voti ottenuti dalle studentesse si calcola la regressione lineare del GPA rispetto al SAT, e quindi la differenza fra i voti reali e quelli predetti come *GPA Underprediction*. Questa è positiva quando i voti reali sono superiori, negativa i voti predetti.

In tutte le sezioni, le donne avevano questa metrica in media positiva (indicando voti superiori alla predizione) e gli uomini negativa. Usando il test T si mostra che il risultato è rilevante.

È stato utilizzato il test di Sobel per calcolare la correlazione tra il genere, coscienziosità e sottovalutazione. Per far ciò si sono prodotte tre regressioni lineari:

- Sottovalutazione rispetto al genere
- Coscienziosità rispetto al genere
- Sottovalutazione rispetto al genere e coscienziosità.

Dai coefficienti ottenuti è stato osservato che in tutti i gruppi la coscienziosità spiega almeno in parte la correlazione e in uno la spiega completamente.

Uno dei gruppi conteneva studenti appartenenti a diverse discipline. Raggruppandoli nei campi scientifici, umanistici, e scienze sociali ed eseguendo lo stesso processo si è mostrato che questa tendenza a sottovalutare i risultati delle studentesse non dipende dalla disciplina studiata e non si concentra esclusivamente nello STEM.

Anche se le differenze ottenute sono relativamente piccole, possono avere risultati significativi sia nel le ammissioni al college sia nel selezionare i riceventi di borse di studio.

2 Data science

Negli ultimi decenni, la nostra capacità di raccogliere e memorizzare grandi quantità di dati si è espansa esponenzialmente. Questo ha fatto nascere un desiderio di utilizzare le informazioni in esso contenute per ottenere vantaggi.

La Data Science è l'insieme di tecniche statistiche che ci permettono di organizzare ed estrarre informazioni dai dati in modo da ottenere intuizioni per il futuro.

Una buona analisi permette di migliorare i risultati ottenuti in ogni situazione: in campo clinico sintomi e risultati di esami possono essere utilizzati per diagnosticare pazienti nelle fasi iniziali e più gestibili della malattia; aziende possono identificare prodotti che non rendono abbastanza e quale strategia promette una maggiore crescita nel futuro; mentre serie temporali di visite permettono di identificare fattori di stress nel sistema (sia esso un sito web che non ha retto, un negozio a corto di personale in certi periodi) e sviluppare contromisure.

Questo capitolo si propone di descrivere le tre discipline principali di data science.

- **DATA MODELLING** si occupa di identificare gli scopi della ricerca e della creazione di un database efficace per mantenere una collezione di dati rilevanti ed essa.
- **DATA EXPLORATION** si occupa della ricerca e visualizzazione di pattern e tendenze intrinseche nei dati.
- **PREDICTIVE ANALYTICS** si occupa di identificare quali fattori sono importanti per il conseguimento di un certo risultato e creare un modello che ci permetta di prevedere come dati futuri si comporteranno e come possono essere influenzati per ottenere risultati più favorevoli.

Come parte di questo processo descriveremo anche i passi che compongono la pipeline KDD – il processo che ci permette di estrarre informazioni dai dati puri.
In seguito, esamineremo un linguaggio di programmazione utilizzato nella data science

– python – e in particolare le librerie utilizzate per manipolare database, ed estrarre e visualizzare informazioni.

2.1 Data Modelling

Per riuscire ad avere informazioni dai dati è necessario conoscere le necessità del sistema e ottenere i dati organizzati in modo che possano essere interpretati.

Data Modelling è l'insieme di tecniche per creare e visualizzare il sistema di informazioni che sono richieste dall'organizzazione che organizza la creazione del database.

Il primo passo è ovviamente raccogliere informazioni sulle necessità e obbiettivi che il dataset verrà utilizzato: il dataset verrà idealmente aggiornato e usato per anni; quindi, inconsistenze e mancanze possono creare problemi anche dopo che il creatore originale è uscito dal progetto o possono rendere difficile l'espansione o adeguamento con progetti più ampi. Solo poi si creeranno mappe per concettualizzare la struttura.

I modelli possono essere creati su vari livelli di astrazione, ma sono diagrammi formali che servono a capire meglio quello che i dati contengono. A questo scopo si usano tecniche e schemi standardizzati in modo che il dataset sia concettualizzato e mantenuto in modo riconoscibile e consistente non solo per l'organizzazione originaria ma anche al di fuori, nel caso sia ritenuto necessario condividerlo con altri utilizzatori.

Tre modelli vengono prodotti durante il processo, cominciando sempre da quello più astratto per poi passare ai livelli più concettuali. In ordine, sono:

1. **MODELLI CONCETTUALI**: creati come parte del processo di esplorazione delle esigenze del dataset, offrono un'immagine generale delle entità (oggetti, persone, concetti) che devono essere rappresentati nel dataset e come si collegano fra loro e con che regole (devi mantenere solo la filiale in cui l'impiegato lavora in quel momento o devi mantenere lo storico di tutti i trasferimenti?). Definiscono categorie di informazioni importanti con le loro caratteristiche e i requisiti.

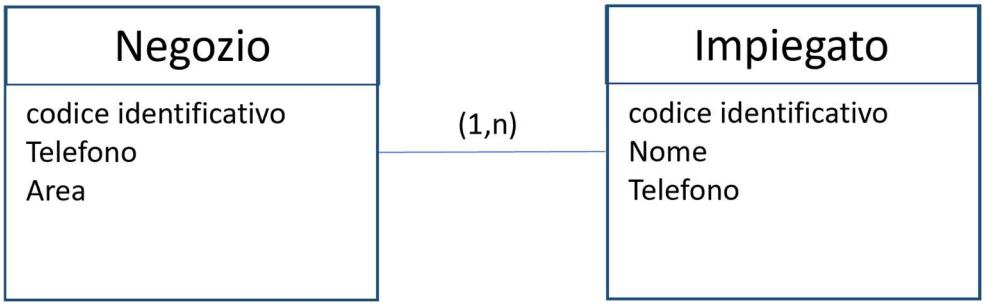


Figura 1

2. **MODELLI LOGICI**: Descrivono con più dettagli i concetti e le relazioni che devono essere inseriti nel database indipendentemente da dove quest'ultimo debba essere implementato. In questo passo indichiamo quali attributi devono essere inseriti, una loro descrizione e che tipo di dati siano.

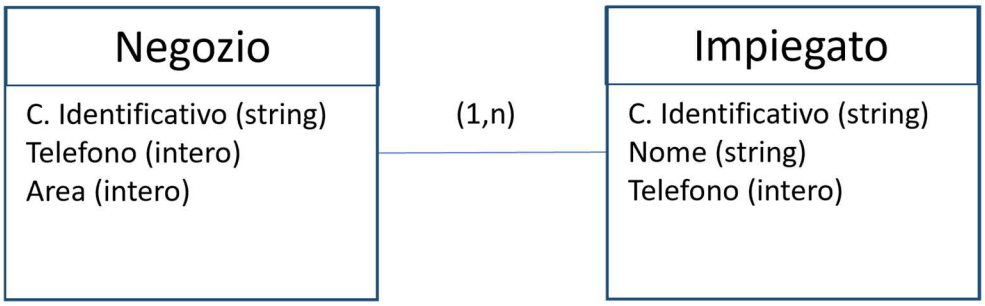


Figura 2

3. **MODELLI FISICI**: forniscono uno schema di come il database deve essere fisicamente creato. Questo è il modello che deve essere implementato come database relazionali e contiene sia le tabelle che rappresentano le associazioni fra i dati sia quali sono le chiavi primarie e quali valori devono essere unici.

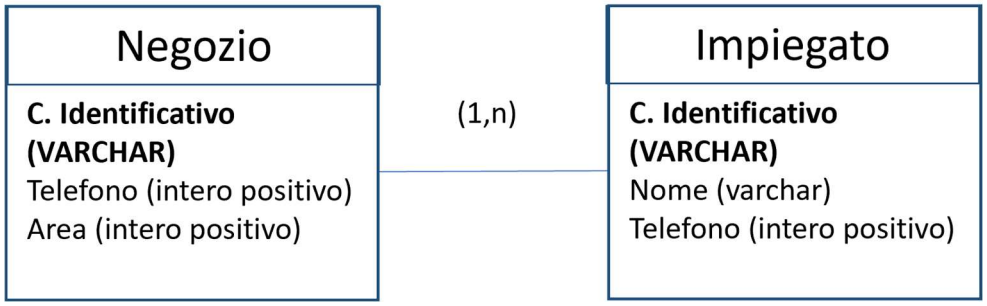


Figura 3

Produrre modelli chiari ha vari vantaggi, facilitando l'implementazione del database, aumentando la sua performance, e chiarendo per chiunque ci interagisca cosa esso contenga e come questa informazione è organizzata: permette sia di evitare errori sia nello sviluppo di programmi che utilizzano il database, sia in studi successivi; inoltre permette a sviluppatori successivi di ampliarlo con facilità mantenendo consistenza.

Il processo di modellazione può essere diviso in sei passi:

1. Identifica le entità che devono essere inseriti nel dataset. Questi devono essere separati fra loro e coesi con loro stessi.

2. Identifica le proprietà di questi entità che possono essere interessanti al dataset. Alcune proprietà possono ripetersi (clienti e negozi hanno numeri di telefono).

3. Identifica le relazioni fra le entità, e la natura di queste: un ordine è connesso a un singolo indirizzo, ma il cliente può essere connesso a più indirizzi, e gli indirizzi possono essere connessi a multipli clienti e ordini.

4. Assicura che le proprietà vengano assegnate alle corrette in modo da ottimizzare il modello.

5. Assegna chiavi identificative dove è necessario e tenta di ridurre la ripetizione di informazioni. Gli ordini sono collegati a un cliente e a un indirizzo. Utilizzando chiavi identificative uniche per tutti e tre possiamo evitare di ripetere

informazioni e permettere di ottenere comunque informazioni come lo storico degli ordini.

6. Finalizzazione e verifica l'efficacia del modello.

Idealmente, questo processo va ripetuto e raffinato qualora emergessero necessità nel corso dell'utilizzo.

Il modello prodotto da questo processo può avere varie forme a seconda dello scopo:

- I più comuni sono i **MODELLI RELAZIONALI**, rappresentati in Figura 4 che collegano le tabelle fra loro nel caso molti a molti usando altre tabelle per ridurre la complessità del database. Per questo modello non è necessaria una conoscenza approfondita delle strutture fisiche usate, e permette di ridurre la ripetizione di informazioni e garantire l'integrità dei dati.

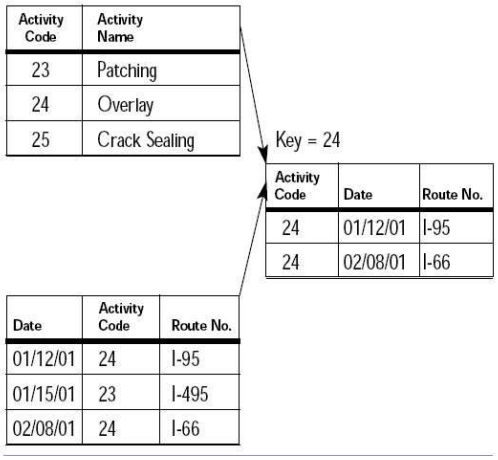


Figura 4. Modello gerarchico. Da U.S. Department of Transportation, Public domain, via Wikimedia Commons

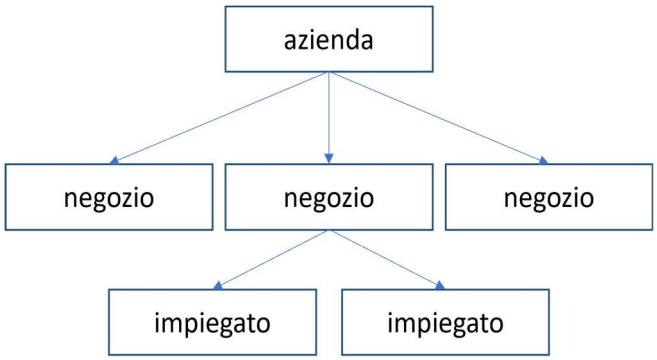


Figura 5. Modello gerarchico.

- Leggermente più vecchi sono i **MODELLI GERARCHICI**, che rappresentano relazioni uno-a-molti come alberi: ogni record ha una singola radice che si collega a una o più record figli. Per rappresentare relazioni molti-a-molti, è necessario replicare i dati. Anche se poco efficiente, è ancora utilizzata nel sistema informativo geografico (GIS) e nel linguaggio di markup (XML).

- I **MODELLI E-R** (entity-relationship) modellano formalmente nel diagramma non solo le entità ma anche le relazioni e le caratteristiche. Può essere tradotto in uno schema gerarchico.

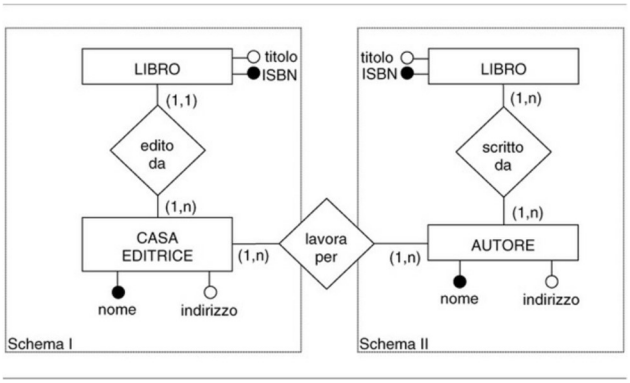


Figura 6: schema ER. Tratto da Tratto da Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

- Sviluppato quando la programmazione a oggetti ha acquistato popolarità, i **MODELLI A OGGETTI** si trattano sia le entità sia le relazioni come oggetti organizzati in gerarchie e permettono l'utilizzo non solo di tabelle ma anche di relazione più complessa. Grazie a ciò è utilizzato per ipertesti e progetti multimediali.

- I modelli sviluppati più di recente sono i **MODELLI DIMENSIONALI**, che per ottimizzare l'accesso alle informazioni contenute nei big data invece di essere ottimizzate per ridurre lo spazio necessario incrementano le ridondanze e facilitano l'accesso alle informazioni. Per modello viene generalmente utilizzato con uno schema a stella dove al centro vi è la tabella principale con i suoi attributi, ed attorno ad essa sono rappresentate le tabelle dimensionali ciascuna contenete come attributi i diversi livelli di granularità che ci interessano. Se delle tabelle di dimensione

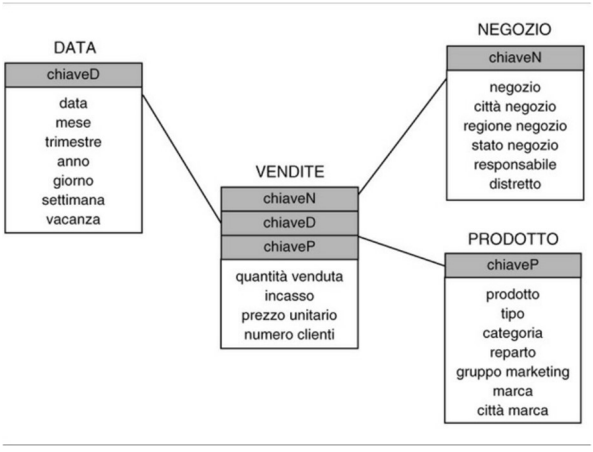


Figura 7: Schema a stella. Tratto da Tratto da Golfarelli, Rizzi,"Data warehouse, teoria e pratica della progettazione", McGraw Hill 2006

vengono decomposte in multiple tabelle subordinate l'una all'altra si parla di schemi snowflake.

Il modello relazionale è fondato sull'evitare ripetizioni di dati tramite l'uso di tabelle per rappresentare sia le entità sia le relazioni fra esse.

Le tabelle delle entità sono caratterizzate da una chiave primaria, un valore unico che caratterizza ogni record, e una serie di altre colonne che esprimono gli attributi. Ogni data deva essere accessibile senza alcuna ambiguità. Ogni colonna è caratterizzata dal nome che deve essere unico all'interno della tabella e dai valori ammessi in essa che può o meno comprendere NULL. Riempito completamente o in parte, ogni record occupa lo stesso spazio nel server di implementazione.

Quando un'entità deve essere connessa a un singolo record di un'altra entità, si può usare la chiave primaria di quest'ultima come chiave esterna o straniera nel primo (per esempio, per ogni tabella di libri bisogna indicare l'unica casa editrice).

Ovviamente questo non è possibile quando i record di una prima entità possono essere collegati a multipli altri record della seconda che possono essere collegati a multipli membri della prima a meno di ripetere dati: ad esempio, un singolo autore può scrivere più di un libro e ciascuno di essi può essere in collaborazione con uno o più altri autori. Per risolvere vengono create nuove tabelle che contengono come chiavi straniere le chiavi primarie delle entità che vengono connesse.

Questa struttura ci permette sia di mantenere la consistenza dei dati sia di ridurre il più possibile le ripetizioni. Se vogliamo accedere a informazioni contenute su multiple tabelle, possiamo unirle per ottenere dati completi.

Quando i dati storici si accumulano la preoccupazione principale è quella di accedere alle informazioni in essi contenuti rapidamente invece che evitare le ripetizioni. Per questo sono state sviluppati i Data Warehouse. Per di creare questo database per ogni campo di interesse si crea un modello dimensionale dove al centro

elencheremo le informazioni che vogliamo aggregare e attorno elenchiamo le dimensioni lungo la quale vogliamo suddividere i dati e le precisioni richieste. Se una dimensione ha una singola precisione, può essere aggiunta alla lista degli attributi nella tabella centrale.

Questo modello viene tradotto in un database costruendo per primo una tabella per ogni dimensione dove vi è un record per ogni unità più piccola di quella dimensione dove sono stati prodotti i dati. In questa tabella le altre colonne si riferiscono alle altre precisioni di quella dimensione (es. il mese e l'anno di quel giorno, la regione e stato della città). Infine, creeremo la tabella centrale che ha come colonne le chiavi primarie delle dimensioni e gli attributi da aggregare. In questa tabella verrà creato un record per ogni combinazione di dimensioni per cui vi erano dati nel database originale.

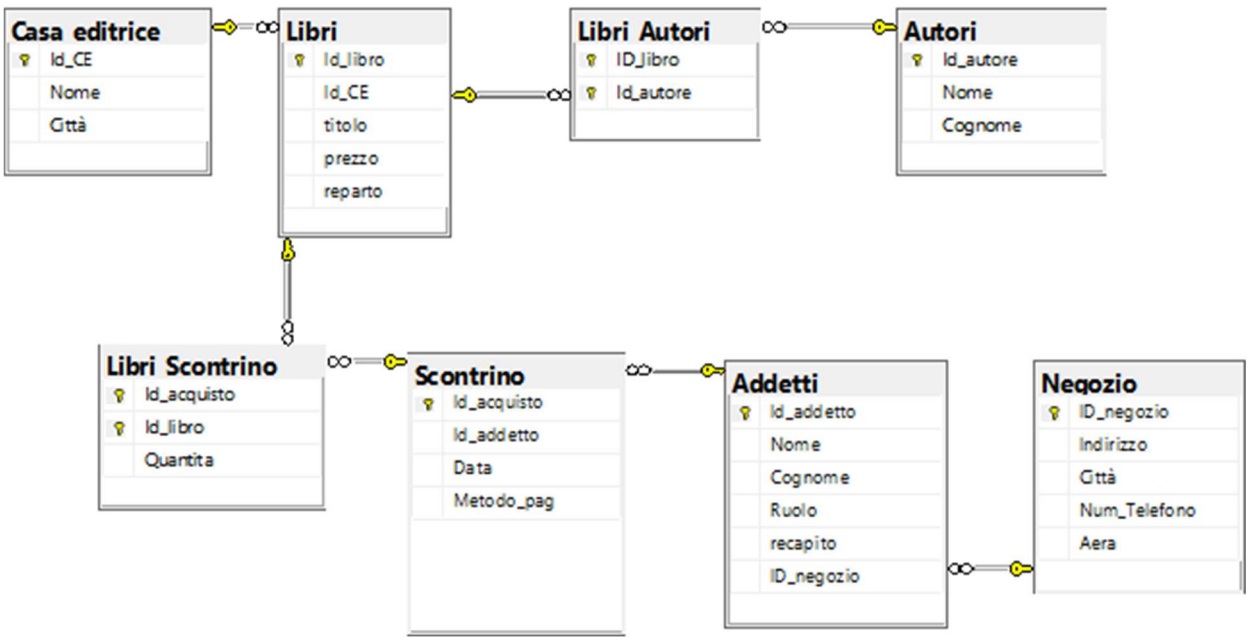


Figura 8: Esempio di modello relazionale

Questo permette di aggregare velocemente le informazioni che vogliamo per qualsiasi precisione richiesta. Se sappiamo che qualche combinazione di precisioni verrà visualizzata spesso può essere utile anche creare ulteriori aggregazioni più generali per ridurre i tempi di accesso.

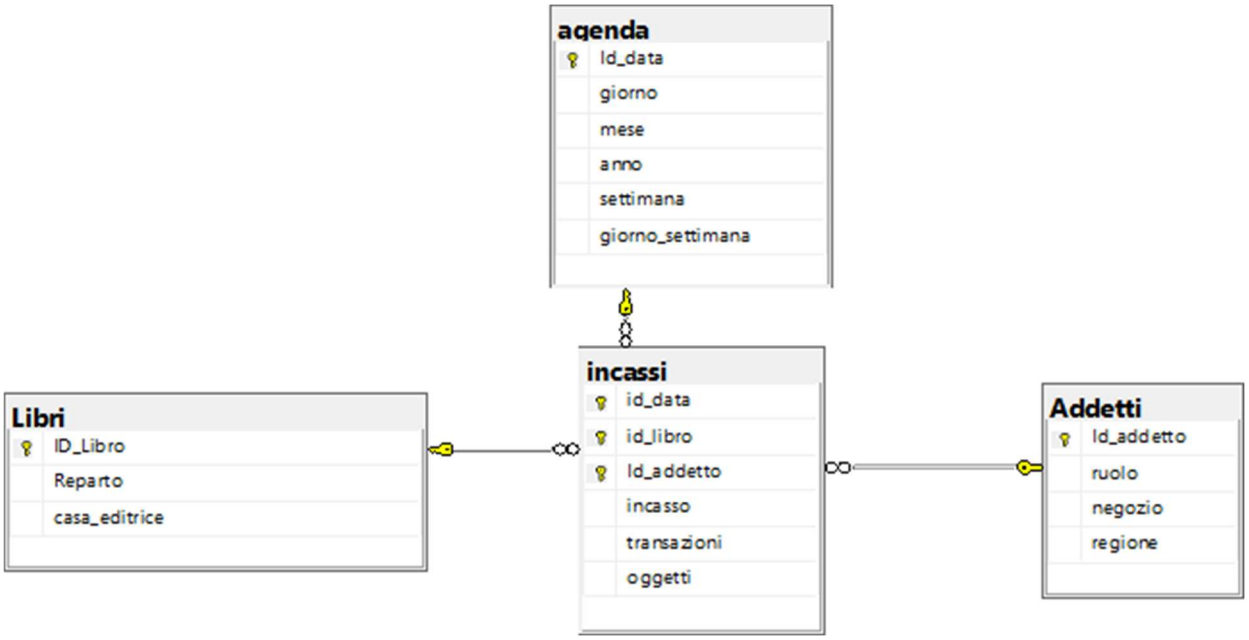


Figura 9: Esempio di modello a stella, basato sul modello gerarchico precedente

2.2 Data exploration

Conoscere il contenuto del dataset è fondamentale per qualsiasi analisi su esso. A causa delle dimensioni, una completa analisi diretta è impossibile quindi dobbiamo utilizzare varie tecniche che permettano di visualizzare i dati, descriverne le caratteristiche e le relazioni.

A seconda che il nostro obbiettivo sia di esporre e visualizzare pattern nei dati senza conoscere necessariamente cosa stiamo cercando o sia di predire il comportamento di una variabile che consideriamo dipendente dal resto, si parla rispettivamente di Data Exploration o di Predictive Analytics. In questa sezione ci concentreremo sul primo dei due.

Il processo di Data Exploration è generalmente schematizzato utilizzando la pipeline KDD mostrata nella Figura 10:

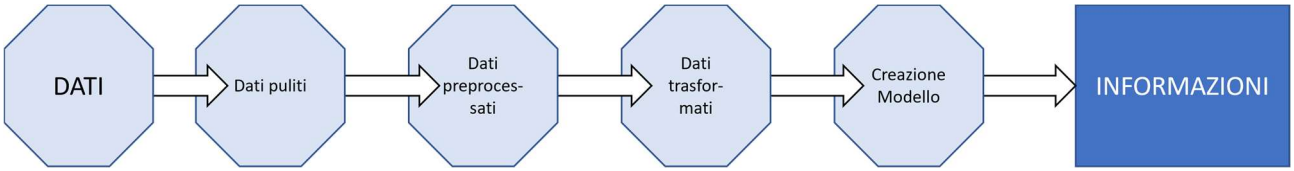


Figura 10: processo KDD

Il primo passo è esaminare i dati a disposizione per identificare il contenuto di ciascuna colonna e selezionare le informazioni significative: capire se i valori sono continui o categorici è particolarmente importante. A questo punto possiamo rimuovere colonne e record che consideriamo inutili.

A questo punto possiamo trovare caratteristiche dei valori contenuti in qualsiasi colonna, che dipenderanno dalla precedente analisi:

- per valori numeri si possono sia valori come il minimo, il massimo, le medie e deviazioni standard, sia i quartili, cioè i valori Q1, Q2, Q3 tali che k% dei dati è minore o uguale ad esso, dove k è rispettivamente 25, 50 e 75. notiamo che il secondo quartile è la mediana;

- mentre per stringhe ed altri valori categorici possiamo scoprire il numero di valori unici e la rispettiva frequenza, e la presenza e significanza di valori nulli. Molto spesso è utile trattare gli elementi categorici come variabili numeriche trasformando la colonna in una serie di colonne di valori binari, una per ciascun valore unico, in modo che sia zero in ciascuna eccetto in quella che identifica il valore.

Molto utili in questo stadio sono in particolare alcuni grafici, come:

- istogrammi che permettono suddividono il range di dati in una regione e mostrano la numerosità degli elementi in quella sezione con l'altezza della barra;

- matrice di correlazione fra colonne;

- funzioni di ripartizione e approssimazioni della funzione di densità;

- boxplot, che stilizzano la distribuzione dei dati attraverso i quartili e la mediana e mostrando gli “outlier”, valori al di fuori di un range sempre dipendente dai quantili e dalla distanza fra essi;
- scatterplots per ogni coppia di variabili che ci permettono di visualizzare la relazione fra variabili e possibili regressioni fra i due;
- grafico dei “residui”, cioè gli errori fra le suddette regressioni e i valori effettivi. Anche questo ci permette di identificare gli outlier.

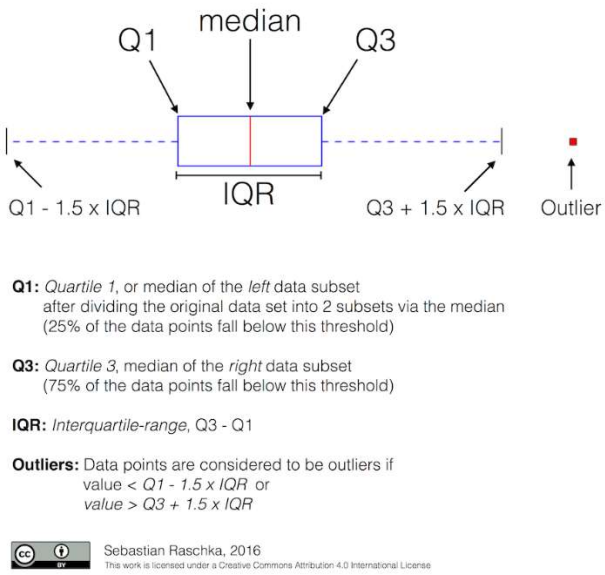


Figura 11: spiegazione di come i boxplot rappresentano le informazioni. da https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.boxplot.html?highlight=boxplot#matplotlib.axes.Axes.boxplot

In aggiunta ad illustrare alcune proprietà base, le informazioni ottenute in questa fase sono utili per effettuare la pulizia e il preprocessing dei dati: possiamo eliminare o trasformare dati che per qualche ragione non si comportano come indica l’analisi (valori fuori dal range previsto, errori di battitura nelle categorie, stringhe in colonne numeriche). Per le colonne categoriche bisogna anche scegliere se è utile lasciale così come sono dividendo i dati in sottogruppi o se è preferibile trasformale in serie di colonne binarie.

Oltre a ciò, possiamo cominciare a identificare ed eventualmente eliminare gli outlier, valori anomali molto distanti dal resto dei dati di cui abbiamo già parlato nel caso dei boxplot.

Due ulteriori analisi per trovare dati che si comportano in modo differente dal resto sono:

- si può utilizzare l'analisi della varianza o ANOVA su ogni variabile quando i dati hanno una specifica classe che li identifica in gruppi separati per identificare quelle che hanno un'alta probabilità di essere importanti per identificare la classe. Si fa quindi il boxplot su queste in modo da identificare gli outlier.

- DBSCAN viene utilizzato quando la classe non è disponibile: si utilizza questo algoritmo di clustering (descritti in seguito) per identificare sfere che contengono raggruppamenti di dati ad alta densità, identificando come outlier tutti gli elementi che non appartengono né all'interno né al bordo di queste.

Prima di fare altre analisi è importante trasformare i dati in modo, per esempio normalizzando: molto spesso alcune variabili hanno un range molto più amplio rispetto al resto e per questo possono avere troppa importanza in vari modelli

Per facilità di rappresentazione dei dati è utile identificare ed eliminare quali "direzioni" nelle quali i dati tendono a variare meno. A questo scopo utilizziamo l'analisi a componenti principali (PCA).

2.2.1 l'analisi a componenti principali (PCA)

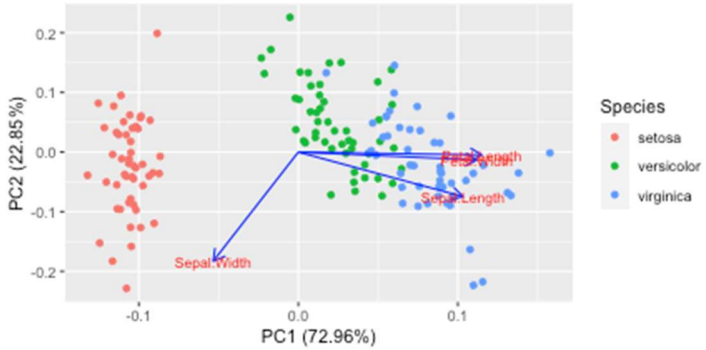


Figura 12: rappresentazione di un dataset lungo le sue prime due componenti principali. da https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html

Questo algoritmo applica una trasformazione ortogonale sulla matrice dei dati per creare un nuovo sistema di coordinate dove i dati sono proiettati con la maggiore varianza sulla prima coordinata, la maggiore varianza ortogonale a quest'ultima sulla seconda coordinata e così via.

Idealmente si calcola la matrice di covarianze dei dati e i suoi autovalori in ordine decrescente. I corrispondenti autovettori sono le colonne della matrice delle componenti principali P tale che, se $\bar{S_1}$ è la matrice dei dati centrata, $X^T = P^T \bar{S_1}$ è la matrice dei dati nelle coordinate richieste. Nella pratica, gli autovettori vengono calcolate iterativamente risolvendo

$$w_i = \text{argmax} \frac{w^T \bar{S}_i^T \bar{S}_i w}{w^T w} \forall i$$

Dove $\overline{S_{i+1}} = \bar{S_i} - \bar{S}_1 w_i w_i^T$.

Tagliare la matrice alle prime m colonne ci permette di ridurre il numero di dimensioni del grafico mantenendo il più informazioni possibili. La scelta di m può essere arbitraria (per esempio, se vogliamo rappresentare i dati in due o tre dimensioni) o può essere scelta basandosi sulla porzione di varianza spiegata da quelle m componenti.

Il grafico in Figura 12 mostra un esempio di dataset con quattro dimensioni rappresentato lungo le sue prime due componenti principali. Le percentuale di varianza per ciascuna componente è indicata negli assi e mostra che insieme le due componenti hanno oltre il 90% dell'informazione. Le frecce indicano la proiezione delle dimensioni sulle nuove direzioni e mostrano che tre di esse sono altamente correlate fra loro mentre la quarta comporta diversamente.

Dopo queste trasformazioni è possibile utilizzare vari metodi che ci permettono di creare modelli che identificano pattern nascosti in nei dati. Esamineremo sia le regole di associazione sia alcuni metodi di clustering.

2.2.2 Regole di associazione

Le regole di associazione identificano frequenti correlazioni fra combinazioni variabili.

Data una serie di dati non ordinati, una regola di associazione $A \Rightarrow B$ indica che se una certa combinazione di variabili A è vera, allora B è anch'essa probabilmente vera. Questa regola non implica che A causa B, ma solo che le due ricorrono insieme.

A è chiamato corpo della regola, B la testa.

Per il momento assumiamo che le variabili siano esclusivamente binarie o oggetti. L'esempio più utilizzato per illustrare questo modello è una lista di scontrini dove ogni colonna indica se è stato acquistato o meno un particolare oggetto senza considerarne la quantità, per cui ogni record viene anche chiamato transazione.

Due metriche per misurare la bontà di queste regole sono:

- **SUPPORTO**: la frazione degli elementi del dataset D che contengono entrambe le componenti della regola $sup(A, B) = \frac{\#\{A,B\}}{|D|}$. In alcuni casi viene considerato solo il numero di elementi.

- **CONFIDENZA**: la frazione degli elementi che contengono A in cui appare anche B $conf(\mathrm{A} \Rightarrow \mathrm{B}) = \frac{sup(A,B)}{sup(A)} = \frac{\#\{A,B\}}{\#\{A\}}$. Questa misura è la "forza" della regola di associazione.

Possiamo estrarre le regole migliori dal dataset selezionando tutte ed esclusivamente quelle per cui sia il supporto sia la confidenza superano delle soglie fissate a priori. Verificare ogni possibile combinazione non è sostenibile al crescere del numero delle variabili e della lunghezza del dataset: se ci sono d variabili, vi sono 2^d possibili combinazioni di cui verificare le regole.

Un principio che ci permette di escludere combinazioni di variabili basandosi sul loro supporto è quello dell'Apriori: il supporto di un gruppo di oggetti è ovviamente maggiore di quello di tutti gruppi che lo contengono, per cui se il primo è infrequente, i secondi

possono essere esclusi permettendoci di ridurre il numero di candidati. Ovviamente, il punto più cruciale per questo algoritmo è quello dove si esaminano le combinazioni di uno o due elementi, dato che sono quelle che ci permettono di rimuovere più candidati.

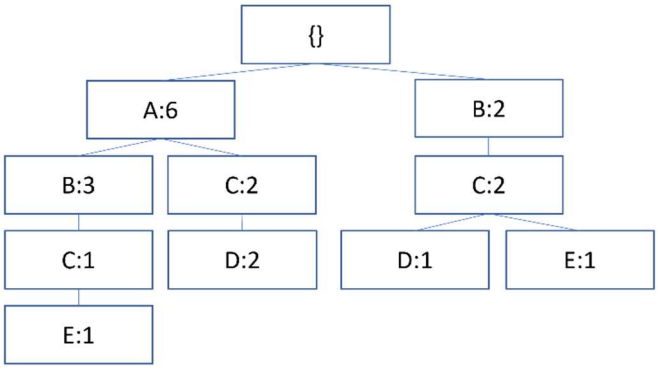
Questo metodo viene implementato producendo la lista di tutti gli oggetti, calcolando il loro supporto ed escludendo gli oggetti infrequenti; per poi produrre tutti i gruppi che contengono solo i rimanenti. Anche per questi è calcolato il supporto per poi escludere le combinazioni rare. Si continuano quindi a creare gruppi aumentando progressivamente la lunghezza, verificando sempre che non contengano combinazioni escluse, calcolando i supporti ed escludendo le combinazioni infrequenti sino a quando non esistono più gruppi di quella lunghezza nella lista dei candidati.

Solo dopo aver ottenuto tutti i gruppi possibili il loro supporto, vengono create le regole di quel gruppo e la loro confidenza.

Questo algoritmo ha la problematica di richiedere ad ogni livello l'esame dell'intero database, aumentando i tempi di elaborazione specialmente quando viene implementata la massima lunghezza dei gruppi di oggetti o nella gestione di BigData. Per ottimizzarlo, si sono sviluppate varie tecniche: per esempio escludere dal dataset elementi che non contengono gruppi rilevanti di una certa lunghezza, utilizzare funzioni hash per ridurre il numero di candidati nei gruppi più piccoli, o aggiungere un candidato solo dopo aver verificato che tutti i suoi sottogruppi siano rilevanti.

Un altro metodo è la costruzione dell'FP-Tree: il primo passo è costruire una tabella degli oggetti e frequenze ordinata in ordine decrescente eliminando eventuali oggetti poco frequenti. Ogni transazione viene ordinata usando questo stesso ordine. L'albero è costituito da nodi contenenti l'oggetto referenziato e il numero di volte che quel nodo viene attraversato durante la costruzione. Questa avviene a partire dalla radice vuota aggiungendo le transazioni all'albero, ogni elemento su un nuovo livello, attraversando dove possibile rami e nodi già creati altrimenti creando un nuovo ramo al livello della differenza.

Transazioni ordinate
A
A, B
A, B
A, B, C, E
A, C, D
A, C, D
B, C, D
B, C, E



Transazioni condizionate a C
A, B
A
A
B
B

Figura 13: esempio di FP-Tree

Per ottenere informazioni si creano, partendo dagli elementi meno frequenti, gli FP-tree condizionali a quell'elemento iniziando dall'albero appena creato.

Questi alberi si creano iterando sui rami dell'albero originale registrando per tutte le transazioni condizionali ad *i* esaminato registrando solo gli oggetti che lo precedono. Queste nuove transazioni vanno utilizzate per formare il nuovo albero *i*-condizionale. A sua volta quest'ultimo può essere utilizzato iterativamente per creare alberi condizionali a tutti i gruppi contenenti i e gli altri elementi presenti nell'albero.

L'insieme di alberi creati ci permette di estrarre facilmente le regole con supporto e confidenza superiore ai limiti fissati.

Entrambi i metodi utilizzati hanno utilizzato le due metriche sopra indicate per fare una decisione, ma queste possono essere ingannevoli, specialmente la confidenza quando la testa ha un ampio supporto. Una metrica utile e comune è quella della correlazione $corr(A,B) = \frac{conf(\mathrm{A}\Rightarrow\mathrm{B})}{sup(B)}$: un valore troppo vicino a o minore di uno indica che quella regola è fuorviante.

2.2.3 metodi di clustering

Questi metodi ci permettono di dividere i dati in gruppi che evidenziano caratteristiche comuni o permettono di astrarre i dati.

Per ciascun gruppo i dati devono essere il più possibile simili fra loro e differenti dagli elementi degli altri gruppi. Sia queste misure sia il concetto stesso dei cluster non sono ben definiti nell'astratto e devono essere fissati per ogni modello.

Questi metodi si occupano di separare e assegnare una classe ai dati utilizzando esclusivamente le informazioni dei dati, al contrario dei metodi di classificazione discussi in seguito.

2.2.3.1 DBscan

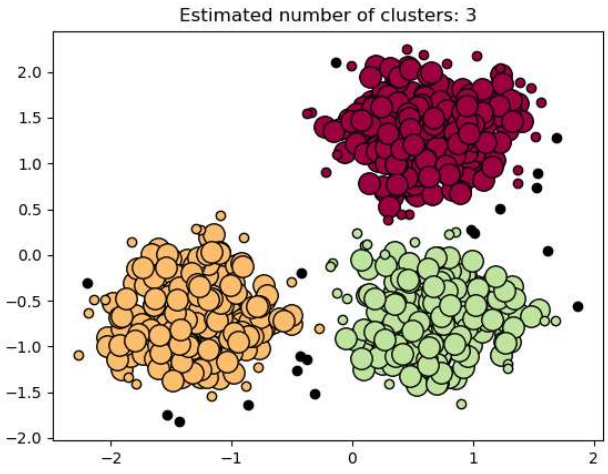


Figura 14: DBSCAN. Da https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html#sphx-glr-auto-examples-cluster-plot-dbscan-py

Il DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) è un metodo di clustering basato sulla densità che permette di classificare tutti i dati come appartenenti a un cluster o come rumore a seconda del numero di elementi nelle loro vicinanze e

della loro classificazione. Come menzionato in precedenza può essere utilizzato per selezionare outlier e dati rumorosi.

In questo caso, la densità di un elemento è stimata come il numero dei punti, compreso sé stesso, in una sfera centrata in esso di un certo raggio. La scelta del raggio varia notevolmente il livello di informazioni ottenute: troppo largo, e molti punti hanno come densità la lunghezza del dataset, troppo piccola e tutti hanno densità 1.

Gli elementi verranno inseriti nello stesso cluster dei loro vicini se appartengono a un'area abbastanza densa, altrimenti vengono considerati come rumore.

L'algoritmo itera sui punti e per ciascuno cerca i dati appartenenti alla sfera centrata in esso di raggio ϵ. Se questi sono più dell'iper-parametro k, il punto originario e tutti i suoi vicini vengono classificati come appartenenti allo stesso cluster. Altrimenti, il punto viene classificato come rumore o *noise point*, almeno sino a che un suo vicino venga classificato come elemento di un cluster.

È utile anche un'ulteriore classificazione degli elementi del cluster come core points e border points a seconda che rispettino o meno la condizione di avere molti vicini.

Dato che i border points sono vicini di punti appartenenti a cluster diversi, è necessario sviluppare una strategia che permetta di scegliere a quale assegnarli.

Il valore k è scelto in proporzione al numero di variabili e dei dati disponibili (nella maggior parte dei casi, il doppio delle variabili è sufficiente), mentre ϵ viene scelto facendo il grafico delle distanze di tutti i punti rispetto al loro k-esimo vicino: questa crescerà relativamente costantemente per la maggior parte del dataset, ma per valori superiori a una certa soglia accelererà di colpo. Questa sarà scelta come raggio.

Al contrario di altri metodi, questo algoritmo non ha problemi a distinguere cluster di dimensioni e forme arbitrarie. Tende invece ad avere problemi quando i cluster hanno densità diverse: fissando numero minimo di punti, utilizzando un raggio leggermente superiore si raggruppano cluster separati di densità maggiore nello stesso gruppo insieme anche al rumore che li circonda, mentre un raggio immediatamente inferiore fa

escludere come rumore il contenuto di cluster di densità minore. Questi problemi vengono evidenziati nel grafico delle distanze.

Il metodo è anche molto caro per dataset di grandi dimensioni, dove non esistono metodi per ridurre il numero di elementi da considerare come vicini.

2.2.3.2 k-means

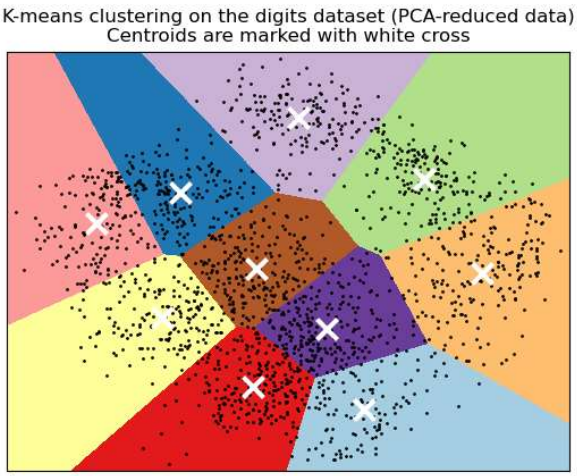


Figura 15: K means su un set di lettere scritte a mano, k=9. da https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html#sphx-glr-auto-examples-cluster-plot-kmeans-digits-py

Una classe di metodi di cluster è quella basata dalla definizione di una serie di *prototipi* che definiscono il cluster assegnando ogni elemento al prototipo più vicino. Il più comune e quello che esamineremo qui è il K-means, il cui prototipo è il valore medio degli elementi del cluster (*centroide*), ma esiste anche il K-medoid, il cui prototipo chiamato *medoid* è il dato con distanza minore a tutti gli altri dello stesso cluster.

L'algoritmo comincia scegliendo casualmente la posizione dei centroidi per poi identificare ciascun dato come appartenente al cluster C_i col centro più vicino. Nel caso di cluster vuoti, il centroide corrispondente è abbandonato e il cluster più largo viene separato in due.

Il centro m_i è quindi ricalcolato come media degli elementi del cluster *i* e dopo i dati vengono nuovamente riassegnati al centro più vicino. Il processo continua sino a che il risultato converge.

Generalmente è usata la distanza euclidea(L^2), ma a seconda dei dati può essere utile utilizzare altre misure come la distanza di Manhattan (L^1) o di Jaccard.

Una misura della bontà del modello nello spazio euclideo è la somma dell'errore quadratico medio $\mathrm{SSE} = \sum_{i=1}^{k} \sum_{x \in C_i} dist(m_i, x)^2$. Questo valore tende a scendere all'aumento di K, ma possiamo utilizzarlo per scegliere il numero ideale di cluster allenando il modello per tutti i valori in un intervallo e facendo il grafico di valori ottenuti. L'errore in un primo momento scenderà rapidamente, ma ad un certo punto rallenterà e formerà un gomito dopo il quale la diminuzione non compete con l'aumento di complessità.

Il valore in questo gomito è quello che viene scelto.

Il metodo è relativamente veloce, specialmente se si accetta una convergenza non perfetta, ma è molto dipendente dall'inizializzazione: anche ripetendo l'algoritmo multiple volte, può risultare improbabile produrre i cluster che minimizzano l'errore scelto. Si sono sviluppate varie strategie per ridurre questa incertezza, fra cui produrre un clustering gerarchico (vedi la sezione successiva) su un campione del dataset e ottenere i centroidi iniziali dai raggruppamenti ottenuti in esso, o più comunemente si sceglie il primo centroide a caso, e per ciascuno dei centroidi successivi scegliamo il dato più distante da quelli già selezionati. Questo metodo è costoso ed è suscettibile agli outliers, ma entrambe queste problematiche possono essere mitigate operando anche questa volta su un campione.

Vi sono anche difficoltà quando i cluster "naturali" non hanno una forma sferica o hanno densità e dimensioni diverse. Mentre ci sono metodi migliori per gestire queste situazioni, possiamo mitigare le limitazioni accettando un numero maggiore di cluster in modo da ottenere che ogni cluster finale contenga elementi di un solo cluster "naturale"

2.2.3.3 Custering Gerarchico

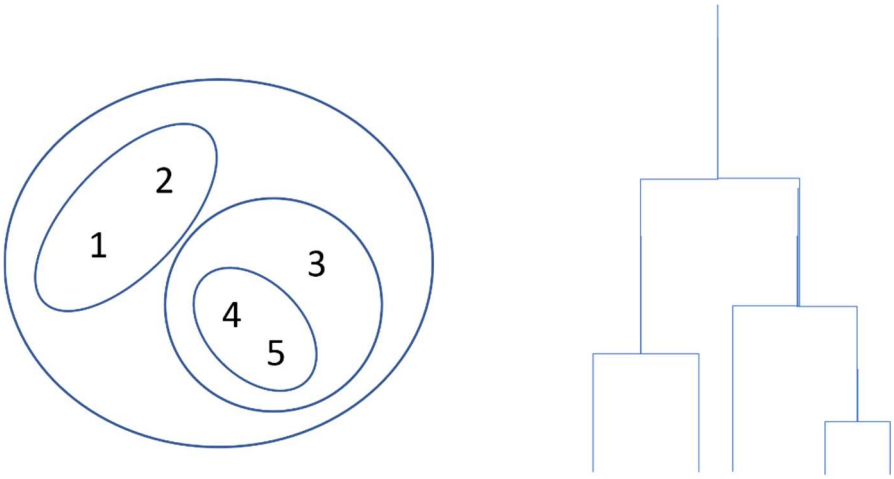


Figura 16:Esempio di clustering gerarchico e relativo dendogramma

Sinora abbiamo esaminato metodi che creano cluster di simile densità e chiaramente separati, ma i dati possono creare gruppi più densi all'interno di cluster relativamente meno densi ma sempre rilevanti.

Il clustering gerarchico è una categoria di algoritmi che ci permette di creare cluster interconnessi utilizzando le distanze fra i dati.

Gli algoritmi possono essere separati in due gruppi a seconda dell'inizializzazione: aggregativo dove tutti i punti vengono considerati un cluster separato e divisivo dove ogni punto è considerato parte dello stesso cluster. Il primo è tradizionalmente più popolare ed è quello che esamineremo ora.

Con il metodo aggregativo, inizialmente si crea la matrice delle distanze fra tutti i punti\cluster. Ad ogni turno vengono uniti i due cluster più vicini e si calcolano le distanze fra questo nuovo cluster e gli altri aggiornando le matrice.

La distanza fra cluster può essere calcolata con varie metriche. Le più comuni sono:

- minima distanza in una coppia di punti, ciascuno appartenente a un diverso cluster;

- massima distanza in una coppia di punti, ciascuno appartenente a un diverso cluster;
- media di tutte le distanze fra coppie di punti, ciascuno appartenente a un diverso cluster;
- distanza fra i centroidi dei cluster (Ward).

Le diverse distanze possono produrre cluster molto diversi.

I cluster possono essere rappresentati utilizzando un *dendogramma,* un grafico ad albero che rappresenta sia la distanza sia i raggruppamenti avendo l'indice dei dati come foglie e indicando la formazione di nuovi cluster nei nodi che hanno come altezza la distanza fra i due che lo formano.

Utilizziamo il dendogramma anche per scegliere quali dei cluster più piccoli e vicini è meglio eliminare per evidenziare pattern e ridurre la complessità. In generale si sceglie un valore e si mantengono solo i cluster con una distanza di formazione superiore. Idealmente i cluster eliminati praticamente la stessa distanza di formazione in confronto alle distanze dei cluster rimanenti.

Questo metodo è molto costoso in termini di spazio e tempo, il primo $O(m^2)$ dove m è il numero di punti del dataset e il secondo, anche utilizzando liste ordinate per la matrice delle distanze, è $O\left(m^2 log(m)\right)$. Gli outlier sono un problema anche in questo caso, specialmente utilizzando distanza di Ward, tendendo a far formare piccoli cluster che non vengono unificati al resto sino alle fasi finali. Un modo di ottimizzazione è quindi quello di identificare ed eliminare questi piccoli cluster.

2.3 Predictive Analytics

In quasi tutte le aziende e campi di studio è importante identificare modelli su dati passati utili per ottenere predizioni su risultati futuri.

Questi modelli identificano strutture e pattern intrinsechi ai dati, identificano le variabili più importanti nel variare del risultato e generano una probabilità che un certo evento avvenga avendo certe condizioni.

Anche in questo caso per creare e utilizzare il modello utilizziamo una serie di passaggi precisi chiamato knowledge Discovery in databases (KDD) rappresentato nella Figura 10.

Come nel Data exploration, il primo passaggio è la selezione dei dati. Questi devono essere un campione significativo della situazione: dati troppo vecchi o troppo sparsi possono causare più inconvenienti che aiutare. Bisogna selezionare anche quali variabili possono essere rilevanti alla nostra analisi: è inutile inserire informazioni che non si avevano prima della realizzazione del risultato cercato.

Anche il processo di pulizia e di pre-processing procede come mostrato nel Data Exploration: si decide se i dati incompleti vanno eliminati o riempiti, gestire inconsistenze di formato e processi simili, si identificano ed eliminano anche outlier, colonne categoriche diventano binarie, e, se si ritiene che possa essere utile, si identificano cluster.

A questo punto si sceglie se applicare trasformazioni come la normalizzazione e il PCA.

Utilizzando i dati prodotti da questo processo si creano modelli che prevedono il comportamento del target. La differenza principale rispetto ai modelli di clustering è la presenza di una variabile risultato che consideriamo dipendente dal resto e che vogliamo prevedere in nuovi dati.

2.3.1 Modelli Di previsione

I dati sono composti da un vettore di variabili x e da un singolo target y. Il target divide ulteriormente i modelli in quelli che possono essere utilizzati esclusivamente in caso y sia una classe, un valore continuo o in entrambi i casi.

I dati a disposizione vengono generalmente suddivise in due sezioni create a caso: la prima, più ampia, viene utilizzata per allenare il modello, mentre la seconda viene tenuta da parte per testare il modello e stimarne l'efficacia, osservando la percentuale di classificazioni corrette o nel caso continuo l'R^2-score

$$R^2 = 1 - \frac{RSS}{TSS} = 1 - \frac{\sum(\hat{y}_i - y_i)^2}{\sum(y_i - \bar{y})^2}$$

Dove y_i sono i valori osservati, $\bar{y}$ la loro media e $\hat{y}_i$ la stima prodotta dal modello

2.3.1.1 Alberi e foreste

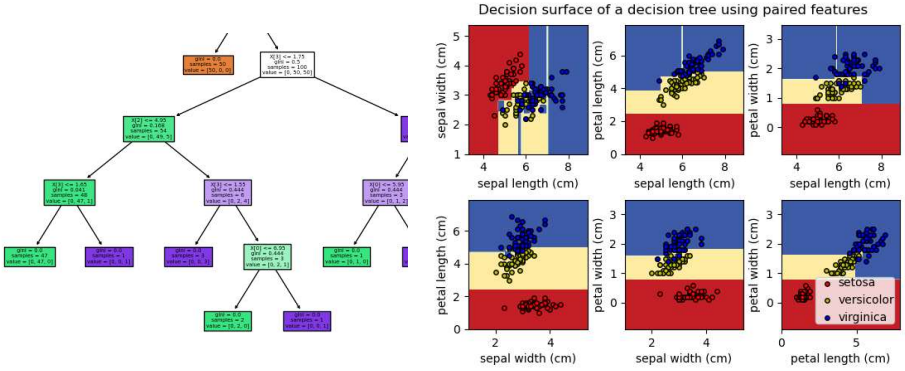


Figura 17: porzione di albero decisionale e rappresentazione della suddivisione dei dati prodotta. Entrambi da https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html#sphx-glr-auto-examples-tree-plot-iris-dtc-py

Il primo metodo esaminato è l'albero di decisione: questo modello imita il processo decisionale umano ponendo una serie di domande e utilizzando le risposte per prevedere il risultato.

Nella maggior parte delle implementazioni queste domande devono essere condizioni con risposta positiva o negativa e quindi possono essere organizzate lungo un albero binario che viene creato ricorsivamente a partire dalla radice.

In ogni nodo viene selezionata una variabile per creare una condizione che separa i dati in due sottoinsiemi: questo processo avviene in ogni nodo successivo sino a che non si raggiungono i nodi finali (foglie) dove invece avviene una classificazione, minimizzando l'errore quadratico dei risultati nella regressione e scegliendo la classe più rappresentata nella classificazione.

Questo processo può andare avanti sino a che ogni elemento di ciascuna foglia è identico – che nel caso continuo può significare una foglia per dato – che oltre ad essere lento e dispendioso causa errori di overfitting. È utile quindi fissare dei parametri quali la lunghezza massima dell'albero o il numero minimo di elementi per foglia in modo da fermare l'algoritmo al momento ideale.

Per determinare la condizione in ogni nodo si minimizza nel caso continuo l'errore quadratico con la media degli elementi che finiscono nel sottospazio creato, mentre nel caso categorico si l'impurità di Gini dei due sottoinsiemi: questo si calcola come

$$\mathrm{GINI(D)} = 1 - \sum_{\mathrm{i}} (\mathrm{p_i})^2$$

Dove D sono i dati nel nodo e p_i sono la percentuale di elementi di classe i presenti in D. In entrambi i casi il risultato è 0 nel caso ideale quando tutti i dati hanno lo stesso risultato e più questo varia più l'indice aumenta.

Le condizioni sono ottimizzate in ogni nodo e non a livello globale, per cui la divisione nel primo nodo è particolarmente importante per garantire che i dati separati nel modo più efficace.

Un singolo albero è spesso influenzato da errori di over fitting; un altro metodo che può essere utilizzato è quello delle random forest: vengono allenati non uno ma una serie di alberi su vari sottoinsiemi dei dati forniti selezionati a campione con il bootstrap (ciascun dato può essere selezionato più volte). Nel momento della classificazione si sceglie la media dei risultati ottenuti o la classe più scelta.

2.3.1.2 SVM

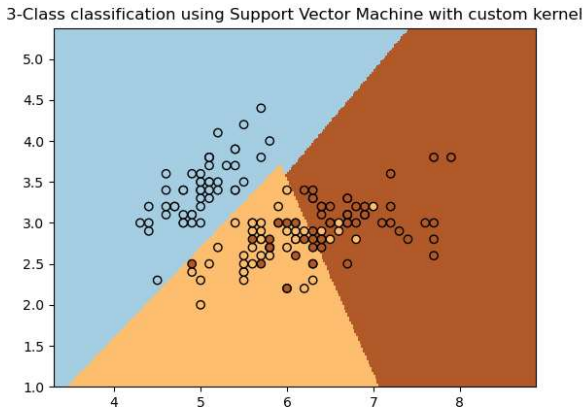


Figura 18: SVM su tre classi. da https://scikit-learn.org/stable/auto_examples/svm/plot_custom_kernel.html#sphx-glr-auto-examples-svm-plot-custom-kernel-py

Un metodo usato esclusivamente per separare dati in categorie sono le support vector machines: nella versione più semplice, con due classi linearmente separabili (identificate da $y_i = \pm 1$), si tratta di identificare l'iperpiano che produrrà l'errore quadratico medio minore nell'utilizzo. A questo fine, si sceglie quello che ha il più ampio "margine" o distanza dai dati in entrambi i lati.

Nella pratica, si tratta di risolvere il sistema:

$$\max_{\alpha} -\frac{1}{2}\sum_{ij} y_i y_j \, \alpha_i \alpha_j (x_i, x_j) + \sum_i \alpha_i$$

$$dove \sum_i \alpha_i y_i = 0 \; e \; \alpha_i > 0 \; \forall i$$

Dove il vettore $w = \sum_i y_i \, \alpha_i x_i$ è il vettore normale all'iperpiano scelto

Per espandere l'uso a dati non separabili viene introdotta una funzione $\Phi(\cdot)$ che trasforma lo spazio dei dati in modo che il kernel $\mathrm{k}(\mathrm{x_i}, \mathrm{x_j}) = (\Phi(x\_i), \Phi(x\_j))$ sostituisca il prodotto scalare nel caso precedente e uno slack che permetta ai dati di essere incorrettamente classificati con una tolleranza C. Si risolve quindi il problema:

$$\max_{\alpha} -\frac{1}{2}\sum_{\mathrm{ij}} \alpha_\mathrm{i}\alpha_\mathrm{j} k(\mathrm{x_i x_j})\mathrm{y_i y_j} + \sum_i \alpha_i$$

$$dove \sum_i \alpha_i y_i = 0 \; e \; \alpha_i \in [0, C] \; \forall i$$

i kernel nativamente disponibili in python sono quattro: lineare, polinomiale, a base radiale (RBF), e sigmoide.

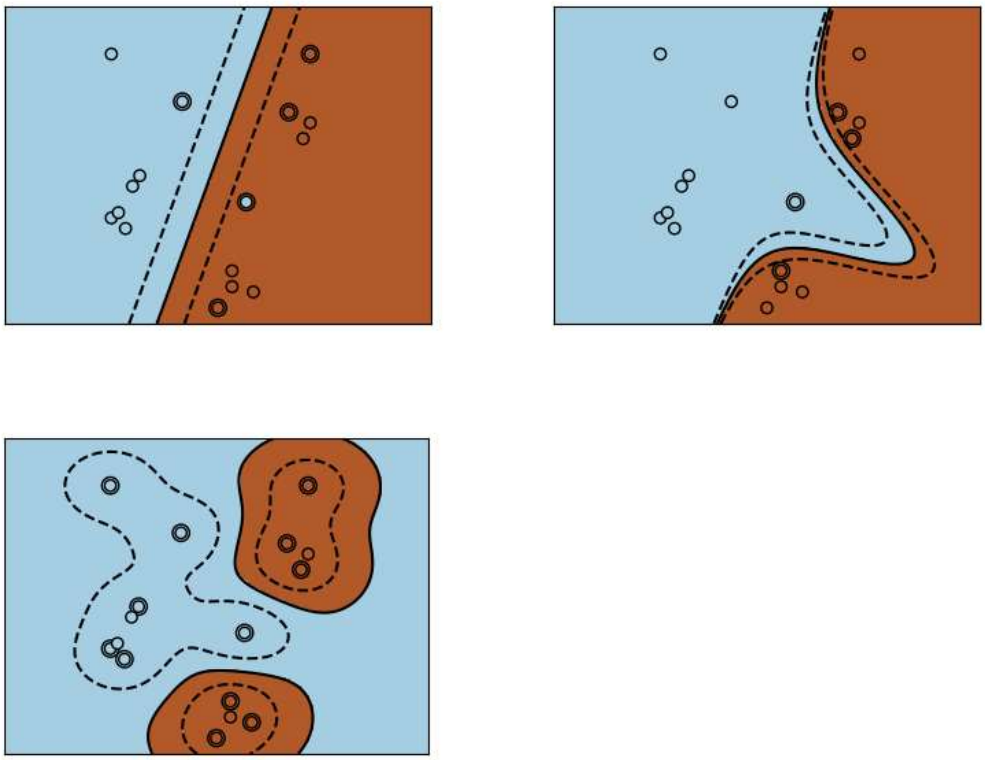


Figura 19: Stesso dataset, differenti modelli SVM con kernel lineare, polinomiale, RBF. Da https://scikit-learn.org/stable/auto_examples/svm/plot_svm_kernels.html#sphx-glr-auto-examples-svm-plot-svm-kernels-py

Per espandersi al caso con più classi si utilizzano due approcci: one-versus-one (OVO) quando si produce una classificazione per ogni coppia di classi e one-versus-all (OVA) quando ogni classe è confrontata con il resto dei dati. In entrambi i casi si classifica nella classe più scelta.

2.3.1.3 Regressione lineare

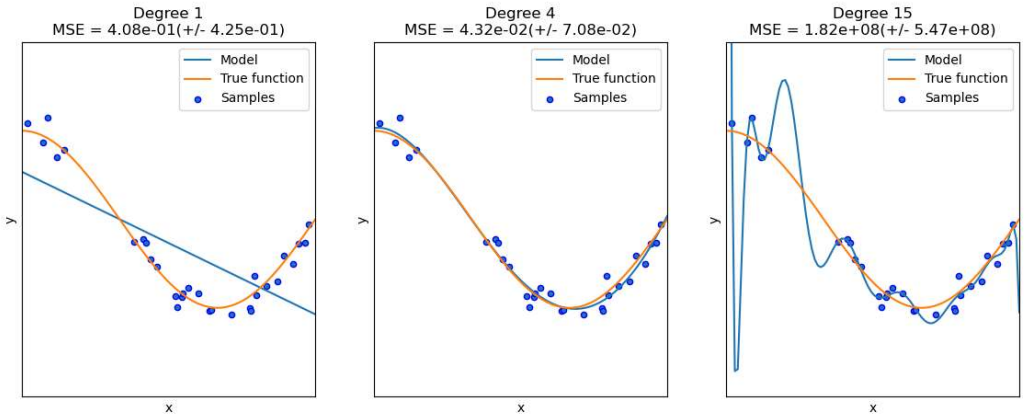


Figura 20: rappresentazione della regressione lineare e con variabili esponenziali. da https://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html#sphx-glr-auto-examples-model-selection-plot-underfitting-overfitting-py

La regressione lineare assume che il risultato y_i sia una combinazione lineare delle altre variabili e quindi sia possibile ottenere una serie di parametri α tali che $\alpha_0 + \alpha^{\mathrm{T}}\, \mathrm{x_i} = \overline{y_i}$ ha il minimo errore quadrato con y_i.

Il risultato non ha iper-parametri da ottimizzare, ma un processo di selezione sulle variabili è spesso utile per aumentare ridurre la complessità e il rischio di over fitting causato da variabili altamente correlate.

Il metodo può essere espanso per ottenere approssimazioni polinomiali aggiungendo termini di grado superiore. In questo caso la "potatura" delle variabili è ancora più importante, dato che il numero di termini aumenta in modo esponenziale.

2.3.1.4 k-Nearest neighbour

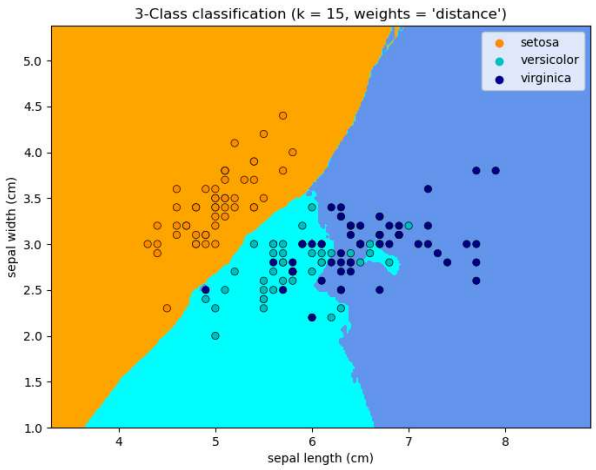


Figura 21: Classificazione di un dataset in due dimensioni con k=15. Da https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py

Inizialmente sviluppato esclusivamente per il caso categorico, per ogni punto dello spazio dei dati si trovano i k dati usati per allenare il modello, si esaminano le loro classi e si sceglie la classe più "votata". Questo metodo è stato espanso anche al caso continuo scegliendo come risultato la media, uniforme o pesata, rispetto alla distanza del target dei vicini.

Il valore di k è il fattore più importante: troppo basso e il rumore ha un'influenza eccessiva, troppo ampio e il risultato ottenuto è influenzato da valori troppo separati oltre ad aumentare la complessità di calcolo.

Questo metodo, anche se semplice e interpretabile, ha severi svantaggi con l'aumento sia del numero delle variabili sia dei dati che aumentano la complessità del metodo esponenzialmente.

2.3.1.5 Neural network

L'ultima classe di modelli creati per la classificazione e regressione è quella delle reti neurali. Queste sono costituite da una serie di strati o livelli di neuroni, a partire da un livello di input, passando per molteplici livelli nascosti e arrivando in fine a un livello di output.

Nonostante l'apparentemente complessità di questa struttura, al suo cuore in ogni neurone vengono presi i valori nel livello precedente, si applica una funzione su essi, e si produce un novo valore da utilizzare al livello successivo; sino ad arrivare all'output. Questo processo deve essere ottimizzato in modo che il valore maggiore nei neuroni al livello di output appartenga a quello corrispondente alla classe giusta.

La funzione del neurone è generalmente la composizione di una funzione lineare e una non lineare chiamata activation function.

Nel caso della classificazione di immagini, la funzione lineare è nei primi strati una convoluzione con una matrice, negli ultimi una combinazione lineare. Rispettivamente, gli livelli vengono chiamati *convoluzional e fully connected*.

Vogliamo minimizzare la funzione di costo o loss che stimi il costo delle misclassificazioni. A causa della non concavità della funzione e del grande numero sia di parametri da ottimizzare, sia dei dati necessari all'allenamento non è possibile utilizzare i metodi classici per trovare l'ottimo locale per cui utilizzeremo il metodo del *stochastic gradient descent*.

A questo scopo separemo i dati in gruppi più piccoli che utilizzeremo a turno durante l'allenamento. I valori ottenuti verranno utilizzati per stimare il gradiente della funzione rispetto ai parametri in quel punto, che a sua volta ci permette di produrre i nuovi parametri come $x^{(t+1)} = x^{(t)} + \alpha \nabla f\left(x^{(t)}\right)$. α è un iper-parametro che può mutare durante il processo di allenamento, e il suo comportamento deve essere stabilito prima di cominciare.

Per ottimizzare ulteriormente questo gradiente viene creato non attraverso metodi tradizionali ma attraverso la *backpropagation*, partendo dal valore in uscita invece e usando le regole di derivazione per ottenere il risultato utilizzando meno memoria possibile.

Questo metodo è difficilmente interpretabile, ma ciò non è necessariamente uno svantaggio: in molte applicazioni si rischierebbero più errori se l'utente finale fosse in grado di capire come ingannare la rete.

2.3.1.6 Scegliere gli iper-parametri: Cross-Validation

Quasi tutti i modelli presentati hanno iper-parametri che devono essere scelti prima di allenare il modello: utilizzare semplicemente tutte le possibilità e scegliere il modello allenato con maggiore precisione ci rende vulnerabili a casi di over fitting e ad ottenere risultati meno effettivi.

Un modo per prevenire questo risultato è l'utilizzo della cross validation: i dati che devono essere utilizzati per l'allenamento vengono divisi casualmente in k gruppi e a turno ciascuno è selezionato per testare il modello, il resto per allenarlo. Si fa la media dei k risultati ottenuti e questo è il risultato che viene utilizzato per ottimizzare gli iper-parametri.

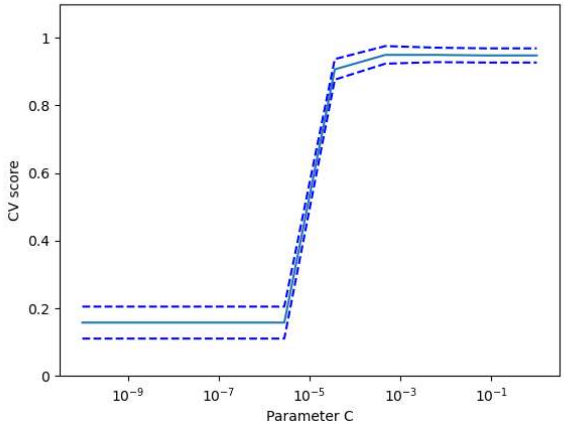


Figura 22: esempio di cross validation per SVM su 10 valori di C

3 Funzioni di python

Per suddividere e organizzare meglio il nostro progetto allo scopo di creare modelli di predizione e grafici abbiamo scelto di utilizzare il linguaggio di programmazione Python. Abbiamo quindi utilizzato una serie di funzioni e script ricreando la pipeline KDD. Quattro librerie in particolare sono state utilizzate in diversi step:

1. Numpy, contenente array più robusti e funzioni matematiche base.
2. Pandas, per le operazioni specifiche relative alle fasi iniziali del processo;
3. Sklearn, per la creazione di modelli sia di clustering sia di predizione;
4. Matplotlib, per la creazione dei grafici sia per la rappresentazione delle informazioni sia estratte dai modelli sia nella fase iniziale di esplorazione.

L'uso combinato delle funzionalità di queste librerie ci permette di estrarre informazioni sia attraverso la Data Exploration sia attraverso la Predictive Analytics.

3.1 Numpy

Numpy è una delle librerie fondamentali per qualsiasi processo matematico in python, e permette di effettuare velocemente operazioni matematiche.

Il cuore di questa libreria è l'oggetto **ndarray**: questo è un array n- dimensionale di lunghezza fissa e con dati di tipo omogeneo (al contrario delle liste native di pyton) su cui le funzioni di questa libreria verranno definite. Un nuovo array viene creato tramite la funzione **array(data)**, dove forniamo i dati attraverso una lista creata in precedenza o inserendo direttamente i valori. L'array identificherà da solo la forma e il formato dei dati che possono essere ottenuti richiedendo i valori **.shape** e **.dtype**. Possiamo usare **.T** per ottenere la trasposta.

Per la creazione di array specifici possiamo utilizzare funzioni come **arange()**, che crea un vettore di valori compresi in un range separati dallo stesso intervallo, entrambi definiti da noi; **linspace()** che crea un vettore di lunghezza specifica di valori in un certo range; o **ones()**, **zeros()** e **fill()** che crea vettori rispettivamente di uni, zeri o un valore scelto; oppure **eye()** che produce matrici d'identità.

Se vogliamo cambiare la forma dei dati, per esempio trasformando un array bidimensionale in uno monodimensionale possiamo utilizzare **.reshape()**, fornendogli le nuove dimensioni. Se vogliamo ottenere un vettore da un array multidimensionale, possiamo usare **.reshape(-1)** oppure **.flatten()**.

Gli operatori come **+** e **-** quando applicati su due array della stessa forma, producono le operazioni corrispondenti fra gli elementi. Uno scalare viene interpretato come un array delle stesse dimensione riempito da quello scalare. Operazioni su ogni elemento di un vettore possono anche essere fatte utilizzando operazioni come **.exp()** o **.sin()**, senza bisogno di iterare sul'array.

Funzioni come **.sum()**, **.mean()**, **.max()**, **.min()**, **argmax()** o .**argmin()** consentono il calcolo dei risultati lungo l'asse specificato (se non indicato, asse 0). Esistono anche funzioni che permettono di operare su due array, per esempio **.dot()** e .**matmul()** che producono rispettivamente il prodotto scalare e matriciale quando sono entrambi unidimensionali e il prodotto matriciale in presenza di array bidimensionali.

Funzioni più complesse di algebra lineare sono incluse in **.linalg.** , tra cui **.norm()** che produce la norma del vettore o matrice, **.eig()** gli autovalori e autovettori di un array quadrato, **.svd()** la scomposizione in valori singolari su cui l'analisi a componenti principali si basa. È anche possibile tentare di risolvere un problema lineare tramite **.solve()**.

3.2 Panda

Libreria che fornisce gli strumenti per l'analisi dei dati introducendo un due nuovi tipi di array di dati: **Series** (monodimensionale, non possiamo aggiungere nuovi record dopo la creazione) e **Dataframe** (multidimensionale, record e colonne possono essere aggiunti dopo la creazione).

Il Dataframe, oltre ai dati, ha vari attributi come il numero di assi, gli indici dei record, i nomi e tipi di dato delle colonne. Questi possono essere ottenuti e in qualche caso modificati facilmente referenziando alla fine del nome del dataframe l'attributo. Fondamentali per l'esplorazione dei dati sono i metodi **.loc()** e .**iloc()**, che permettono di accedere alle informazioni nel dataset rispettivamente utilizzando gli indici per tutte le dimensioni e il valore posizionale.

Il dataframe può essere creato ad hoc o utilizzando altri oggetti già creati nel programma, ma può anche essere importato utilizzando metodi del tipo **.read_f(path\url)**, sostituendo a f il termine adatto per il formato da importare. Fra i formati disponibili ci sono CSV, excel, HTML, SQL e JSON. Corrispondenti a questi metodi di input c'è anche la famiglia di .**to_f(path\url)** per l'output.

Una delle abilità più importanti di questa libreria è la sua capacità di gestire il database in modo simile al linguaggio SQL: l'operazione di SELECT può essere ottenuta ponendo un array contenete i nomi delle colonne mentre si applicano le condizioni del WHERE con una combinazione di **.loc()** e funzioni che applicate sulle colonne producono array binari. È possibile effettuare un JOIN fra tabella attraverso la funzione **.merge(…)** indicando i due oggetti da unire, il tipo di join e le colonne (compresi gli indici) che vengono come chiavi.

Funzioni aggregate sono applicate creando un nuovo oggetto mediante la funzione **.groupby([columns])** e applicando la singola funzione se vogliamo la stessa statistica per ogni colonna, **.agg({})** se vogliamo moltiple funzioni: all'interno delle parentesi grafe inseriamo le colonne e le funzioni da applicare in forma 'nome colonna': [lista di

funzioni] separate da virgole. È anche possibile creare finestre utilizzando le funzioni nella sub-libreria **.rolling**.

La libreria è utile ma non sostituisce il linguaggio SQL: funzioni come CUBE, ROLLUP e GROUPING SETS non sono nativamente implementati e quindi per ottenere questi risultati è necessario ripetere il raggruppamento multiple volte o sdoppiare i dati.

Queste funzioni possono essere utilizzate principalmente per la selezione, esplorazione, pulizia e del database. Oltre a questo pseudo-SQL, sono utili funzioni generiche che permettono di trasformare il colonne come per esempio **.get_dummies(data)** permette di trasformare ogni colonna che contiene valori categorici in una serie di colonne binarie, oppure .cut(data, bin) che invece suddivide valori continui in categorie.

Questa libreria fornisce anche delle semplici funzioni grafiche derivate da matplotlib. Queste sono prodotte applicando **.plot.** e la funzione interessata al dataframe o serie da rappresentare. Un esempio è **.plot.kde()** che permette di approssimare la funzione di densità dei dati.

3.3 Sklearn

Scikit-learn (generalmente riferita come sklearn) è definita dai suoi creatori come una collezione di “strumenti semplici e efficienti per la analisi predittiva dei dati”.

Basandosi sulla loro documentazione, gli algoritmi contenuti nella libreria possono essere suddivisi in sei gruppi basati sul loro obbiettivo:

- Classificazione;
- Regressione;
- Clustering;
- Riduzione delle dimensioni;
- Selezione del modello;

- Preprocessing.

Le funzioni contenute nelle prime due sezioni ci permettono di creare i modelli che abbiamo elencato nel capitolo 3, divise a seconda che la variabile da predire sia categorica o continua, mentre nel clustering troviamo quelli elencati nel capitolo 2.

Questi modelli vengono definiti come oggetti fornendogli gli iper-parametri e solo dopo vengono allenati con il metodo **.fit()**. Nuovi dati possono essere assegnati una previsione o un cluster attraverso **.predict()**.

I modelli di classificazione e regressione permettono di creare semplici modelli di neural network con **.MLPClassifier()** e **.MLPRegressor()**. Questi non sono ottimizzati per modelli di larga scala, specialmente dato che non permettono di utilizzare la GPU, quindi per motivi non didattici è meglio utilizzare altre librerie specifiche.

La selezione del modello si occupa invece della validazione del modello prodotto attraverso selezione degli iper-parametri. La maggior parte di queste funzioni vengono trovate nella sublibreria **.model_selection**.

Questa comprende sia metodi per la cross validation:

- **.KFold()** (divide i dati in k gruppi delle stesse dimensioni),
- **.StratifiedKFold()** (separa i dati mantenendo approssimativamente le stesse percentuale di classi in ogni sezione)
- **.GridSearchCV()**, che prende il modello e i valori degli iperparametri da testare, sceglie la combinazione migliore e automaticamente produce il modello allenato su tutti i dati.

Oltre alla creazione del modello, questa libreria è utilizzata anche nelle fasi iniziali della pipeline KDD: nonostante il nome, la sezione di preprocessing si occupa principalmente di trasformare i dati. Per esempio si normalizzano utilizzando l'oggetto **preprocessing.StandardScaler()** oppure le variabili categoriche vengono trasformate in binarie con **.feature_extraction.DictVectorizer()**. In entrambi i casi questi oggetti vengono prima allenati tramite **.fit()** e producono i dati trasformati via **.transform()**.

La riduzione delle dimensioni comprende tutti i metodi che ci permettono di ridurre la complessità del modello riducendo il numero delle variabili, partendo dalla selezione delle più importanti tramite l'utilizzo di metodi compresi in **.feature_selection** come **.RFE()** (recursive feature elimination) e **.RFECV()**, passando alla PCA con **.decomposition.PCA()**.

Molti di questi metodi sono anche progettati per funzionare con matrici sparse.

3.4 Matplotlib

Matpotlib è una libreria che permette di visualizzare qualsiasi tipo di dati in figure, non solo dataset. Queste figure sono altamente personalizzabili in base alle nostre esigenze potendo includere vari tipi di grafici e permettendoci di modificarne la rappresentazione. La sua più importante sottolibreria è **matplotlib.pyplot**, generalmente importata come **plt**.

I due metodi più semplici di rappresentare le informazioni in un piano bidimensionale sono **.Axes.scatter()** e **.Axes.plot()**: il primo crea un grafico di dispersione, il secondo connette i dati con una linea. I due grafici possono essere modificati in vari modi, per esempio alterando la scala e i nomi degli assi o colore e simbolo dei segni utilizzati. Possono anche essere combinati fra loro o con altri grafici: se vogliamo rappresentare la regressione lineare di una serie e avere un'idea della sua precisione, possiamo combinare la linea dei valori regressi con i veri.

Una simile rappresentazione esiste anche per le tre dimensioni con i metodi di **.Axes3D.** Questo è basato sulla sottolibreria **.Axes.**, ma è meno potente.

Possiamo produrre altre rappresentazioni grafiche con altri metodi come **.hist(),** che produce a l'istogramma partendo dai dati e separandoli in raggruppamenti che possono essere anche definiti da noi. Un'altra rappresentazione di dati continui è il **.boxplot()**, il cui prodotto è ovvio.

Questi metodi non funzionano nativamente con i Dataframe quindi è importante utilizzate i metodi di pandas. Questi vengono comunque gestiti dall'infrastruttura di Matpotlib: per mostrare la figura prodotta e eventualmente iniziarne un'altra, è necessario inserire il comando **plt.show().** Per produrre più di un grafico nella stessa figura è invece necessario definire la figura e tutti i suoi sottospazi come variabili tramite un'assegnazione **fig, {lista di sottospazi}= plt.subplots()** o **axes =fig.subplots()**, indicando nella funzione quanti sottospazi vi sono lungo ogni asse. I grafici saranno allora assegnati a qualsiasi sottofigura accostando la funzione corretta al nome corrispondente.

4 Metodologie

4.1 Descrizione del problema

Nelle Università italiane, l'ammissione è basata su test attitudinali invece che sulla carriera scolastica; per determinati corsi questo test viene adottato a livello nazionale, ma in molti casi è invece la stessa università a crearlo e distribuirlo.

Due dei tre articoli discussi in precedenza hanno evidenziato che questa procedura può non essere un indice ideale della corretta valutazione degli studenti, specialmente per quanto riguarda l'identità di genere.

Come base di partenza consideriamo, oltre al voto di ammissione in Ateneo anche altre variabili, sia anagrafiche sia della carriera scolastica per costruire un modello dei dati che ci consenta di una previsione del risultato dell'esame finale.

Particolare attenzione verrà riservata ad osservare se l'identità di genere influisca nel processo che porta all'esito dell'esame finale.

4.2 Dati utilizzati

L'Università italiana considerata, opera nel campo delle materie STEM; fornisce corsi a livello triennale e magistrale su due indirizzi principali, a cui appartengono diversi collegi suddivisi a loro volta in vari corsi. Per questo studio, prendiamo in considerazione esclusivamente le informazioni sulle lauree triennali.

Per avviare questo studio, l'Ateneo ha messo a disposizione due dataset.

Il primo descrive i risultati ottenuti degli aspiranti che hanno sostenuto l'esame di ammissione. Ogni persona è identificata da una matricola e vengono forniti i seguenti dati:

- informazioni sull'esame di ammissione come data di immatricolazione, tipo e risultati (sia totale sia per argomento);
- dati anagrafici come sesso, cittadinanza e regione italiana di residenza;
- informazioni sulla carriera scolastica precedente (indirizzo di scuola superiore e voto di diploma).

Si procede ad effettuare una prima scrematura e alla normalizzazione del dataset.

Vengono escluse tutte le matricole di cui non possono essere reperiti o il voto di ammissione o quello di maturità. Successivamente, dopo la trasformazione di quest'ultimo in base 100, vengono esclusi anche quelli che contengono valori apparentemente anomali.

Dall'anno accademico in formato XXXX/YYYY selezioniamo solo il secondo anno solare in modo da poterlo utilizzare come numero intero: in caso di esami multipli per la stessa matricola, vengono selezionati solo quelli per l'anno accademico più recente.

Successivamente abbiamo deciso di raggruppare i dati categorici in modo da ottenere una minore disgregazione di dati: per gli studenti residenti in Italia, la regione è stata fatta confluire nella rispettiva macroregione come da definizione dell'Unione Europea.

Un ulteriore scorporo è stato fatto anche per l'istituto di provenienza:

- Istituti tecnici
- Istituti professionali
- Licei classici
- Licei scientifici
- Altri licei

- Non identificati

Il secondo dataset contiene informazioni relative alla carriera universitaria. Vengono registrati il collegio di ammissione e corso di studi oltre, per chi si è già laureato, data, voto e collegio di laurea. Prima ancora di effettuare un join con il dataset di ammissione, si procede alla selezione degli studenti che si sono effettivamente laureati ed esclusivamente le variabili relative alla laurea triennale.

Unificando – attraverso la matricola - il risultato con i dati del primo dataset, possiamo ottenere altre informazioni come quella sulla durata degli studi.

Per avere un campione consistente e significativo si selezionano esclusivamente le matricole collegate alle lauree del periodo intercorrente dal 2015 al 2019 che sono state ammesse attraverso un test di ammissione specifico (quello più numeroso). Ciò permette esclusivamente l'accesso a uno dei due indirizzi, consentendo quindi di risolvere alcune problematiche sugli studenti che hanno sostenuto più esami di ammissione.

Questo processo ha prodotto una selezione di 9448 studenti che può essere utilizzato per costruire i modelli.

Per poter utilizzare metodi categorici, si tenta di suddividere i voti di laurea sia in intervalli basati su feedback ricevuti da esperti di dominio sia sui quartili.

4.3 Esplorazione

Le variabili su cui viene eseguita l'analisi sono:

- 'Anno laurea'
- 'anno imma': anno di immatricolazione
- 'F': indice binario di genere, vero se donna e falso altrimenti

- ‘durata’: Durata del corso di laurea in mesi
- ‘TS’: tipo di scuola superiore frequentata
- ‘Voto ammissione tot.’: voto totale di ammissione
- ‘MATEMATICAFISICA’, ‘LOGICACULTURA’, ‘STORIA’, ‘DISEGNO’: voto nelle quattro sezioni del test di ammissione
- ‘collegio laurea’:
- ‘Area’ descrive la macro regione di residenza
- ‘voto diploma (100)’: voto di diploma in base 100
- ‘stesso collegio’: variabile binaria che indica se il collegio di iscrizione è lo stesso di quello di laurea.

L’evoluzione del voto di ammissione e diploma, rispetto all’anno di immatricolazione e genere, può essere osservata attraverso dei boxplot.

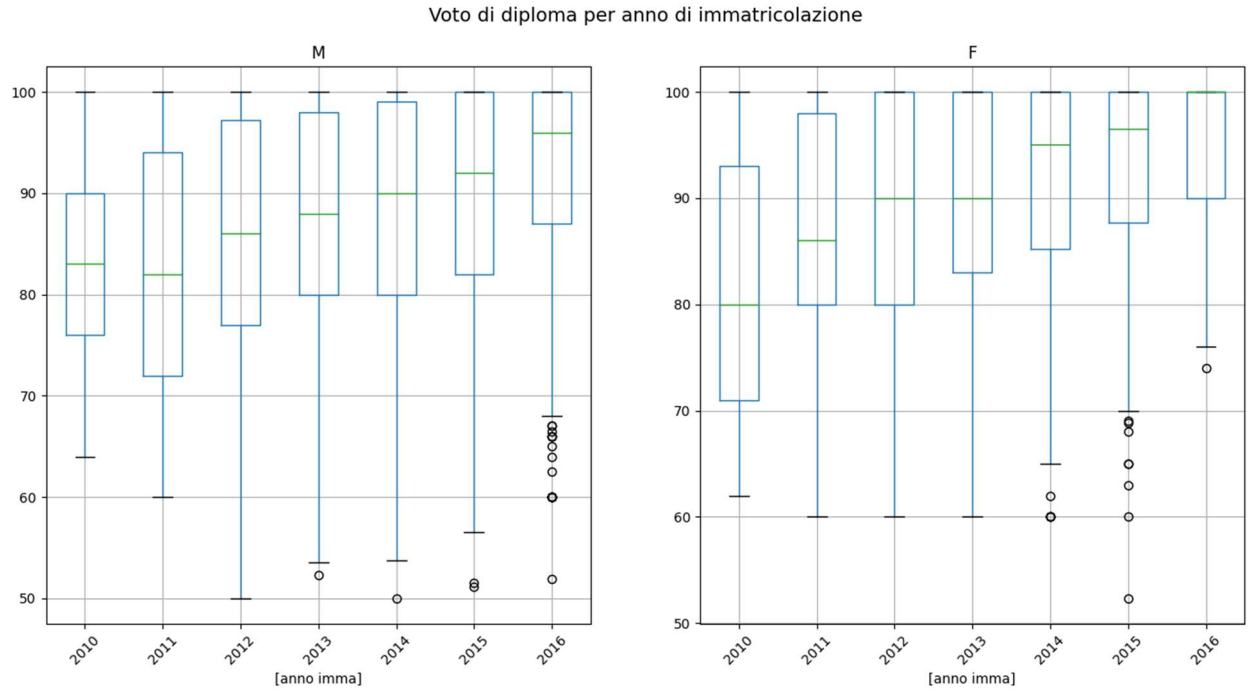


Figura 23 Boxplot del voto di diploma

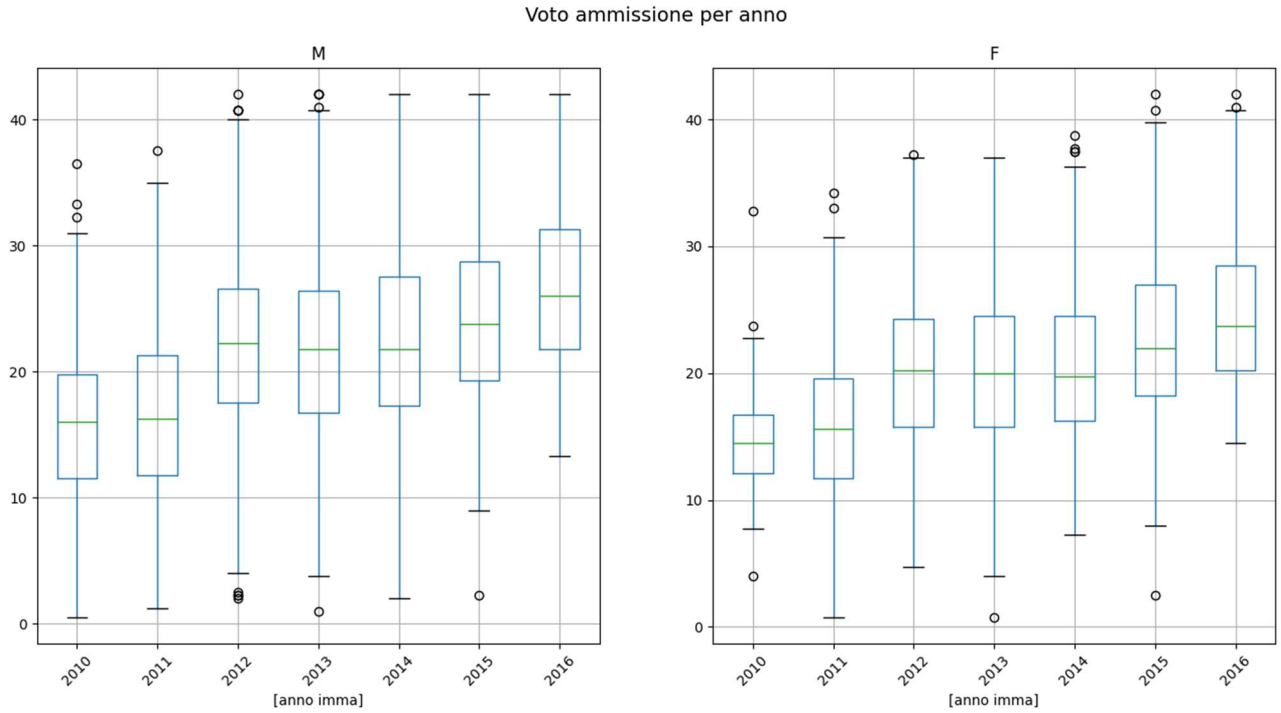


Figura 24 Boxplot del voto di ammissione

Entrambi questi voti sembrano crescere con il passare degli anni, ciò parrebbe indicare che probabilmente studenti con voti migliori a inizio carriera tenderanno a laurearsi in un lasso di tempo minore.

È anche evidente da Figura 23 che le donne sembrano avere voti di diploma superiori, ad eccezione del 2010 dove vi sono meno dati, ma questo vantaggio non viene trasportato nel test di ammissione come mostrato in Figura 24, dove invece si ha apparentemente una inversione.

Al contrario, il voto di laurea viene mostrato rispetto all'anno in cui viene conseguito. Come visto in Figura 25 i risultati in questo caso sono comparabili.

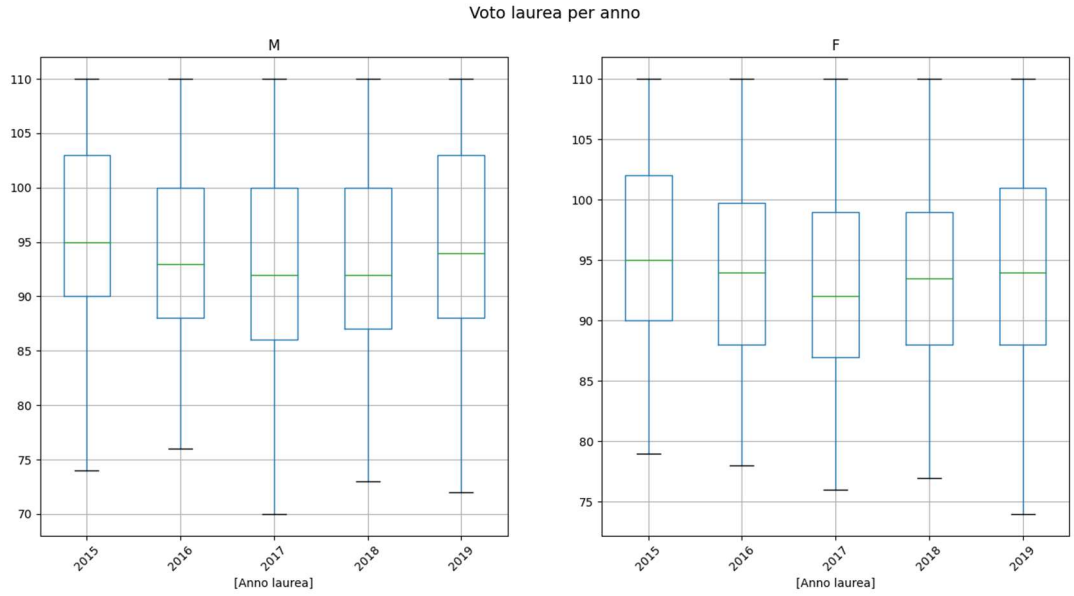


Figura 25 Boxplot del voto di laurea

Per le variabili categoriche non è possibile creare un boxplot; quindi, produciamo grafici a linea indicando per ogni classe quanti studenti vi appartengano per anno di immatricolazione.

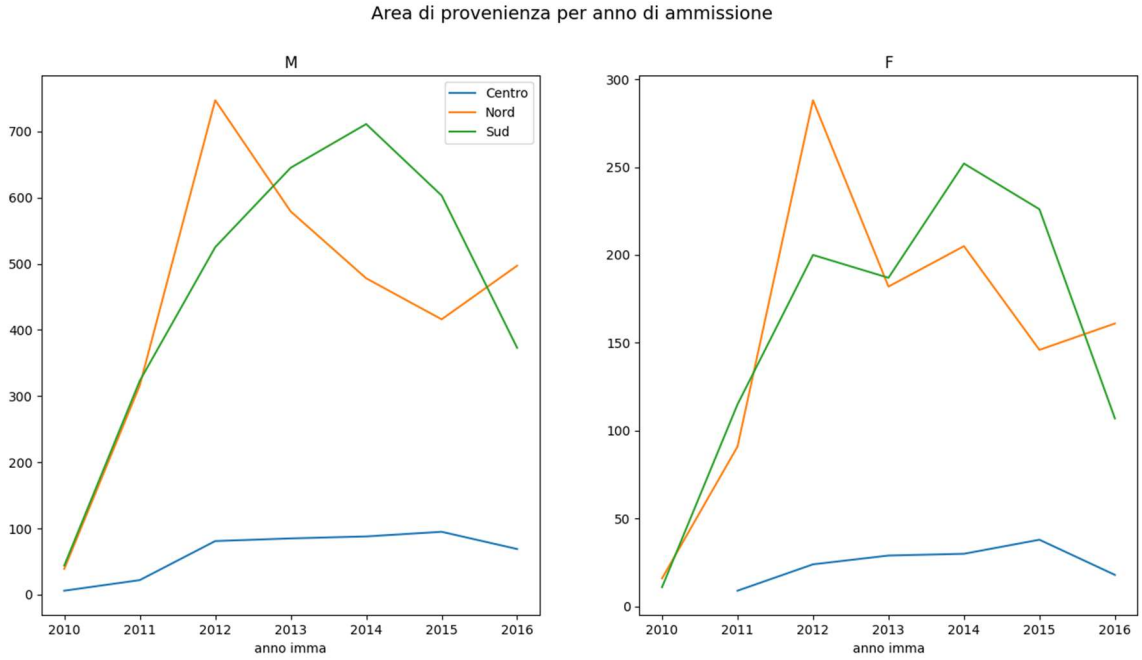


Figura 26

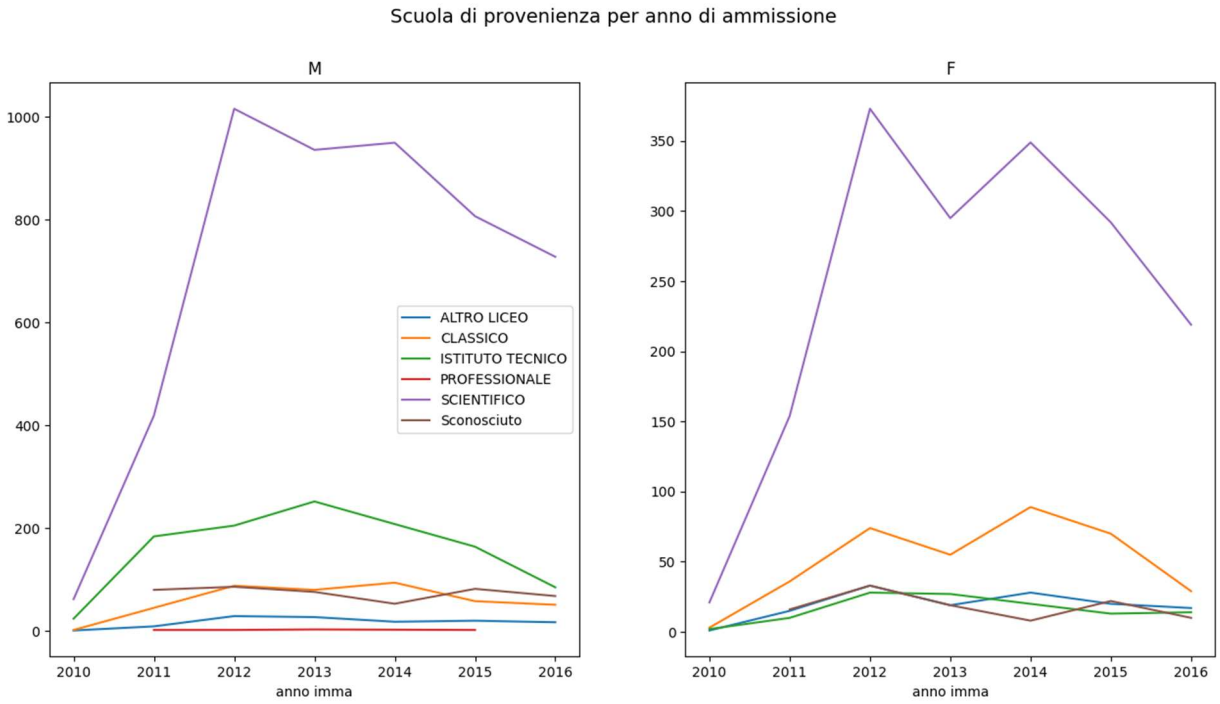


Figura 27

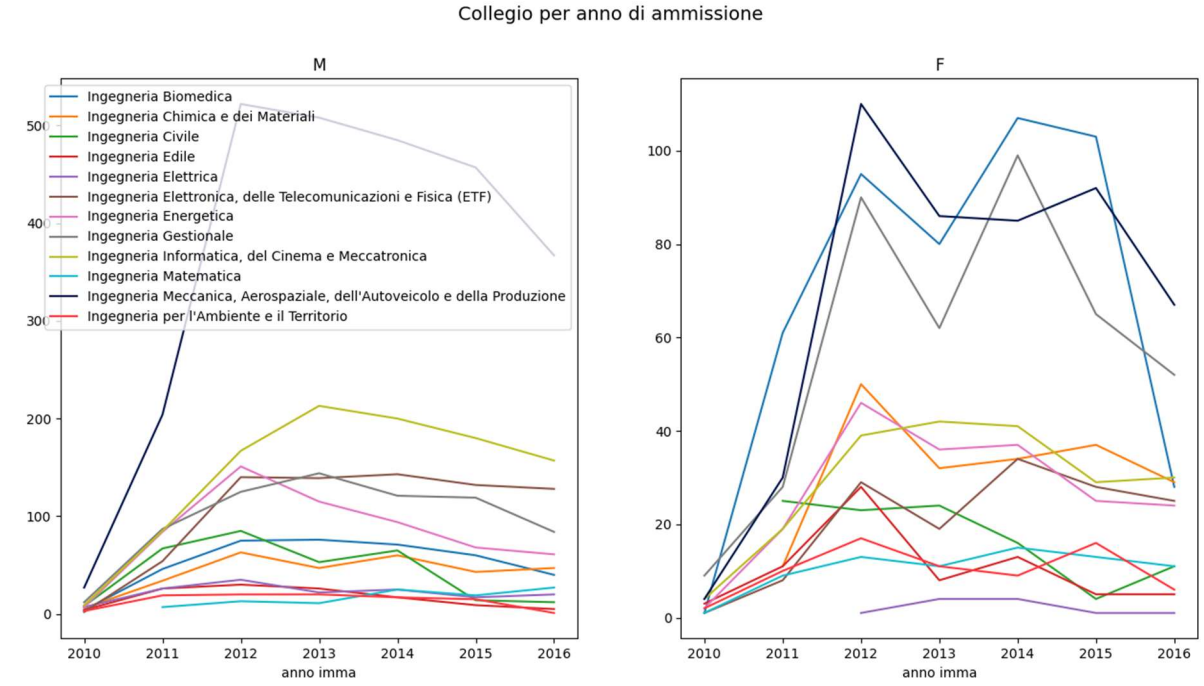


Figura 28

Questi grafici ci permettono di mostrare varie cose; per esempio, in Figura 26 la provenienza per area geografica è la stessa per entrambi i generi: pochi studenti provengono dal centro Italia, mentre Nord e Sud si equivalgono.

Per grandi linee questo vale anche per la scuola di provenienza come visto in Figura 27: buona parte degli studenti hanno frequentato il Liceo Scientifico, ma le donne tendono a provenire maggiormente dal Classico e meno dagli Istituti Tecnici, ma ciò può essere spiegato dalle differenze di genere degli studenti che frequentano quelle scuole.

Il collegio di laurea - che nella maggioranza dei casi è lo stesso con quello di iscrizione - come da Figura 28, si differenzia maggiormente: gli uomini tendono a preferire di gran lunga quello di Ingegneria Meccanica, Aerospaziale, dell'Autoveicolo e della Produzione mentre per le donne, anche se è comunque fra i collegi più frequentati, è al livello dell'ingegneria Biomedica e di quella Gestionale. La seconda specializzazione favorita degli uomini è l'Ingegneria Informatica, del Cinema e Meccatronica; per le donne è meno popolare delle tre menzionate in precedenza ed è al livello di Ingegneria Chimica e dei Materiali e di quella Energetica.

4.4 Risultati

Si sono scelti alcuni possibili modelli da allenare:

- Albero di decisione (sia regressore sia classificatore)
- Random forest (sia regressore sia classificatore)
- SVM
- Regressione lineare.

Nel caso dei classificatori, i modelli sono stati allenati due volte, una per la segmentazione basata su consiglio dell'esperto e una sui quantili.

Per ogni modello vengono scelti degli iperparametri da ottimizzare attraverso il processo di cross validation con k uguale a 5.

Nei modelli di regressione per valutare il risultato finale, si utilizza il coefficiente di determinazione, il rapporto tra devianza spiegata dal modello e la devianza totale. Il valore migliore è 1 e diminuisce per modelli meno precisi, arrivando anche nei negativi.

Per la classificazione, si utilizza semplicemente la percentuale di elementi correttamente classificati e si osserva la matrice di confusione per ulteriori deduzioni.

Nel caso degli alberi, l'unico iper-parametro che viene ottimizzato è l'altezza massima dell'albero. Sono stati fatti esperimenti sul numero massimo e minimo di elementi in ogni foglia, ma si è osservato che variano altamente senza che nei modelli prodotti i limiti vengano mai toccati; si sono lasciati quindi i valori automatici.

Questa altezza massima è anche l'iper-parametro da ottimizzare delle random forest. Sono stat fatti esperimenti anche con il numero di alberi che popolano una foresta ma, simile al massimo e minimo di elementi per foglia, rimangono altamente variabili e può essere lasciato come automatico.

Per l'SVM viene testato il kernel, il valore di C, e il metodo.

La regressione lineare non ha iperparametri, ma possiamo escludere variabili per ridurre l'overfitting.

Il primo modello testato è l'albero di decisione.

La funzione di python, che costruisce il modello, fornisce nativamente l'importanza delle variabili nella classificazione, che vengono rappresenttate utilizzando un grafico a barre.

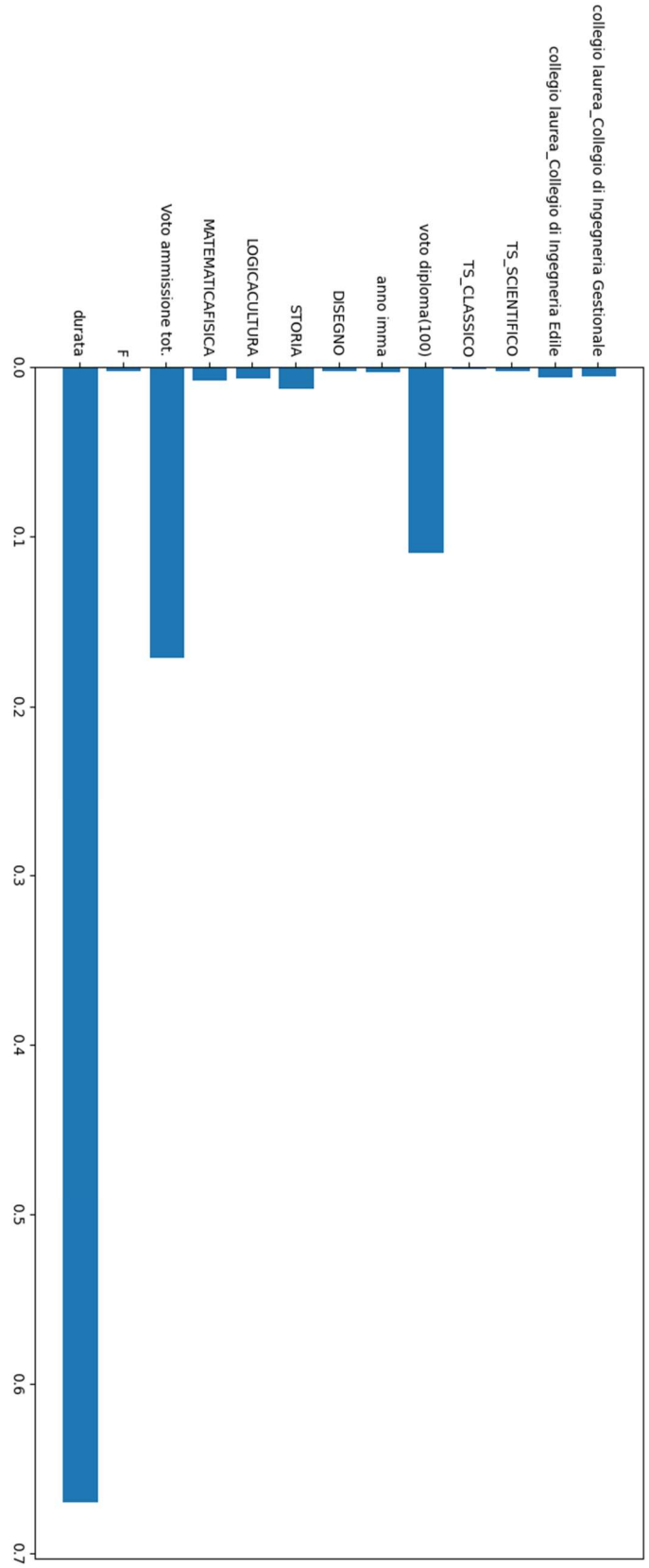


Figura 29 importanza variabili per albero di classificazione basato sui quantili

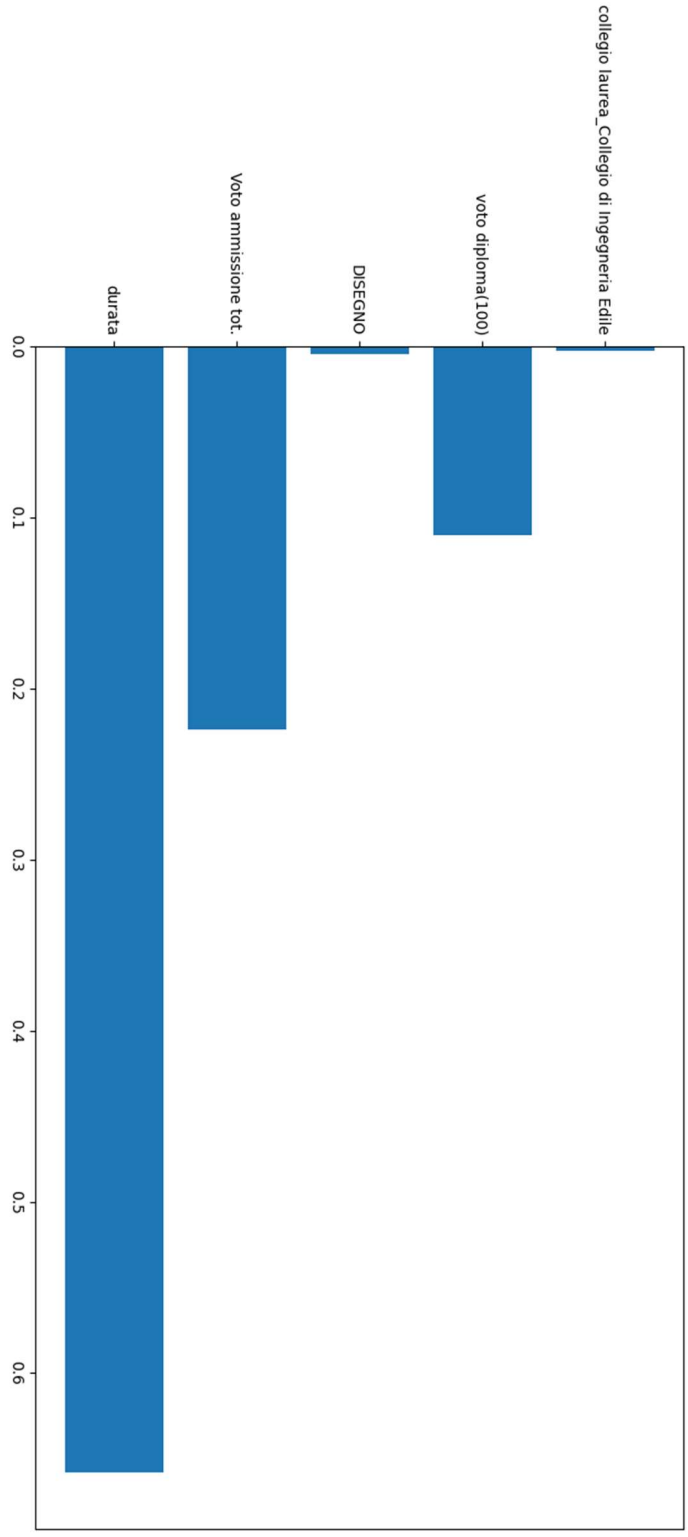


Figura 30 importanza variabili per albero di classificazione basato sull'esperto

Nel caso della classificazione, in entrambe Figura 29 e Figura 30, si osserva che le variabili più importanti sono nell'ordine: durata, voto di ammissione e di diploma. La durata in particolare spiega oltre il 60% della separazione. Solo la divisone basata sui quartili sembra essere influenzata dal genere e in misura molto minore.

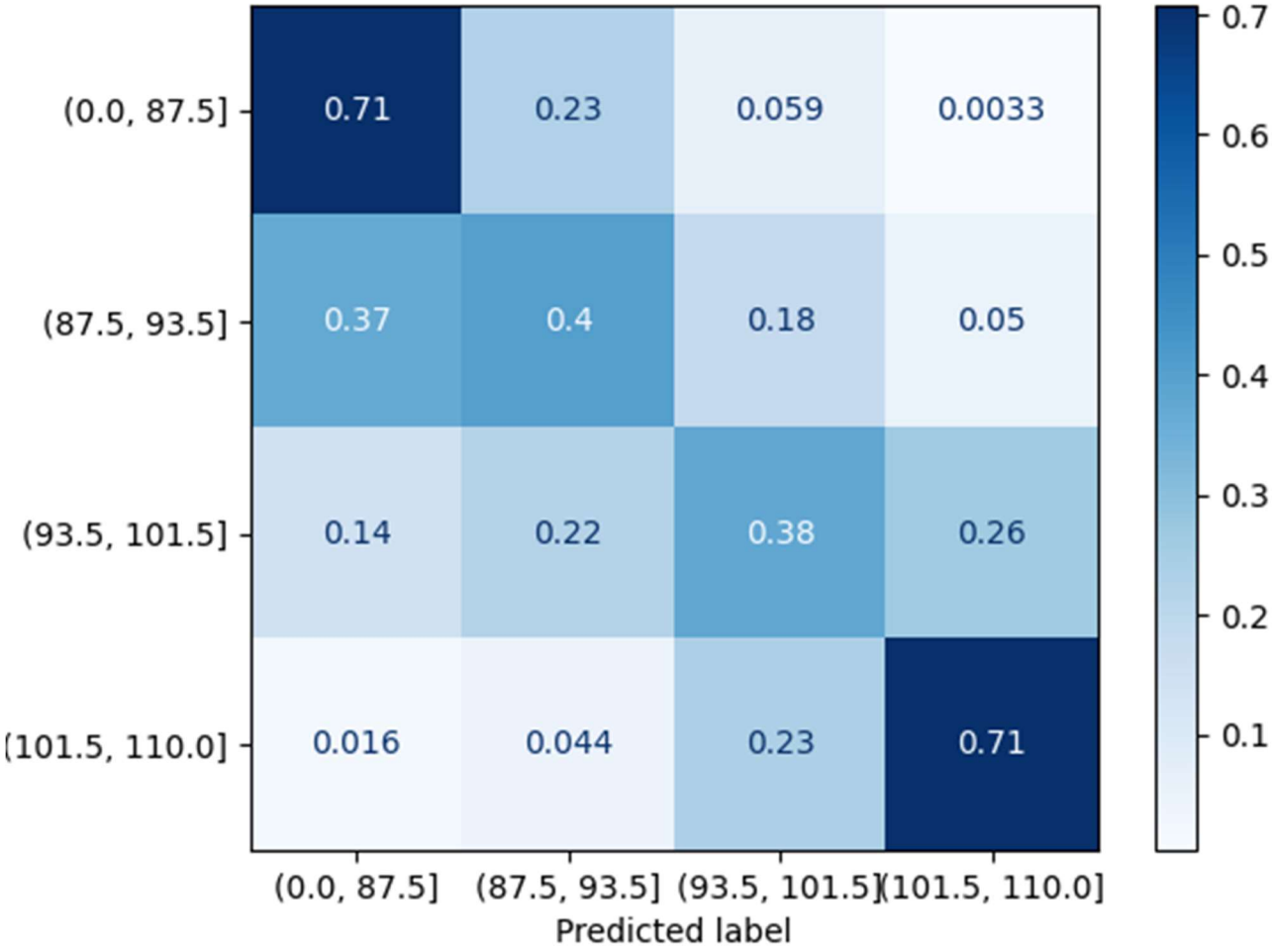


Figura 31 Matrice di Confusione per albero di classificazione basata sui quantili

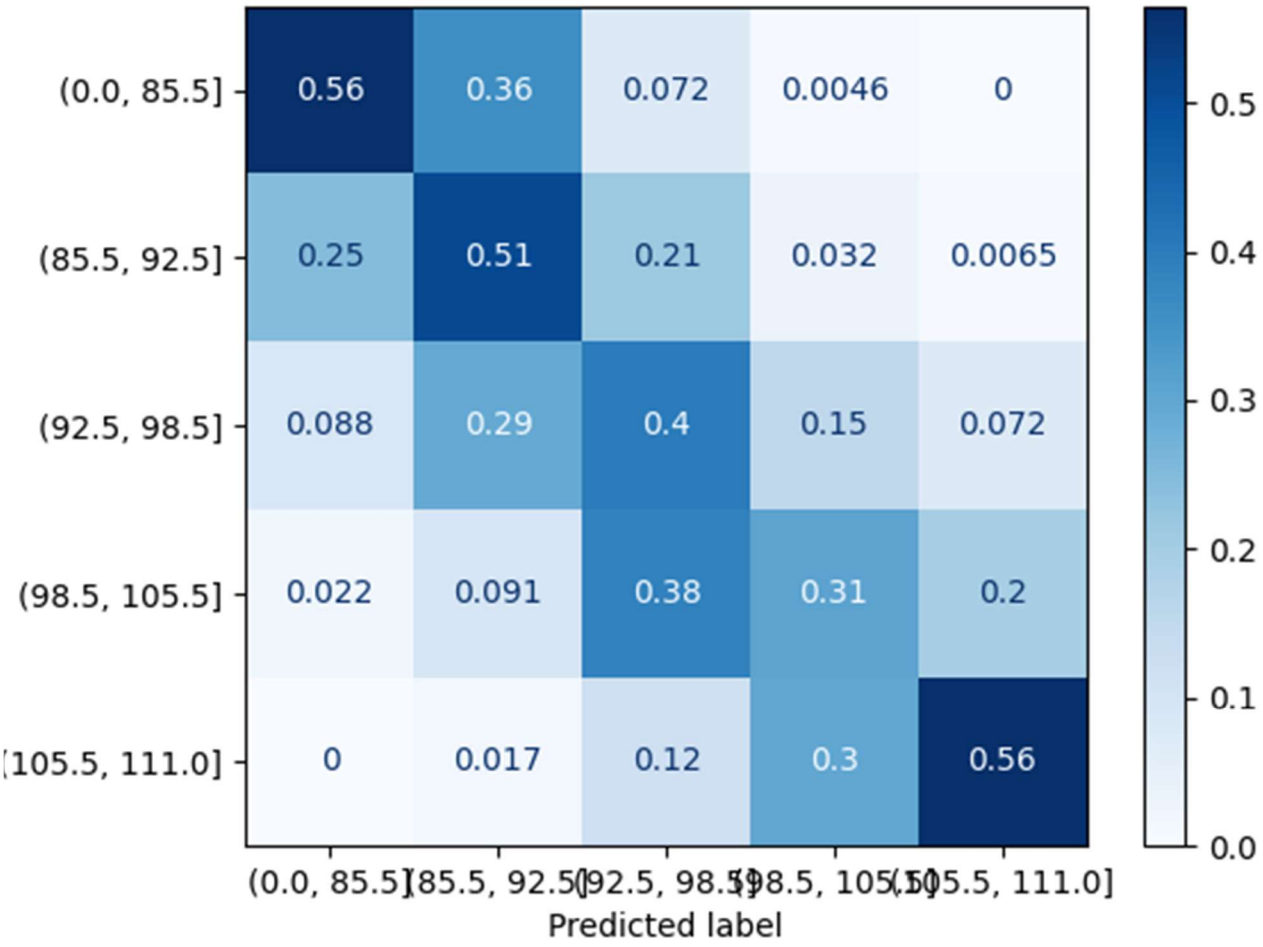


Figura 32 Matrice di Confusione per albero di classificazione sull'esperto.

Osservando le matrici di confusione Figura 31 e Figura 32, vediamo che in entrambi i casi la maggior parte degli errori appartiene alle categorie direttamente superiori o inferiori alla corretta, e che le categorie limite sono le più precise. Purtroppo, la precisione totale rimane rispettivamente al 52% e 48%.

Nel caso continuo si ritrovano la durata, il voto di ammissione e diploma come variabili più importanti. Lo score rimane comunque a 0.56.

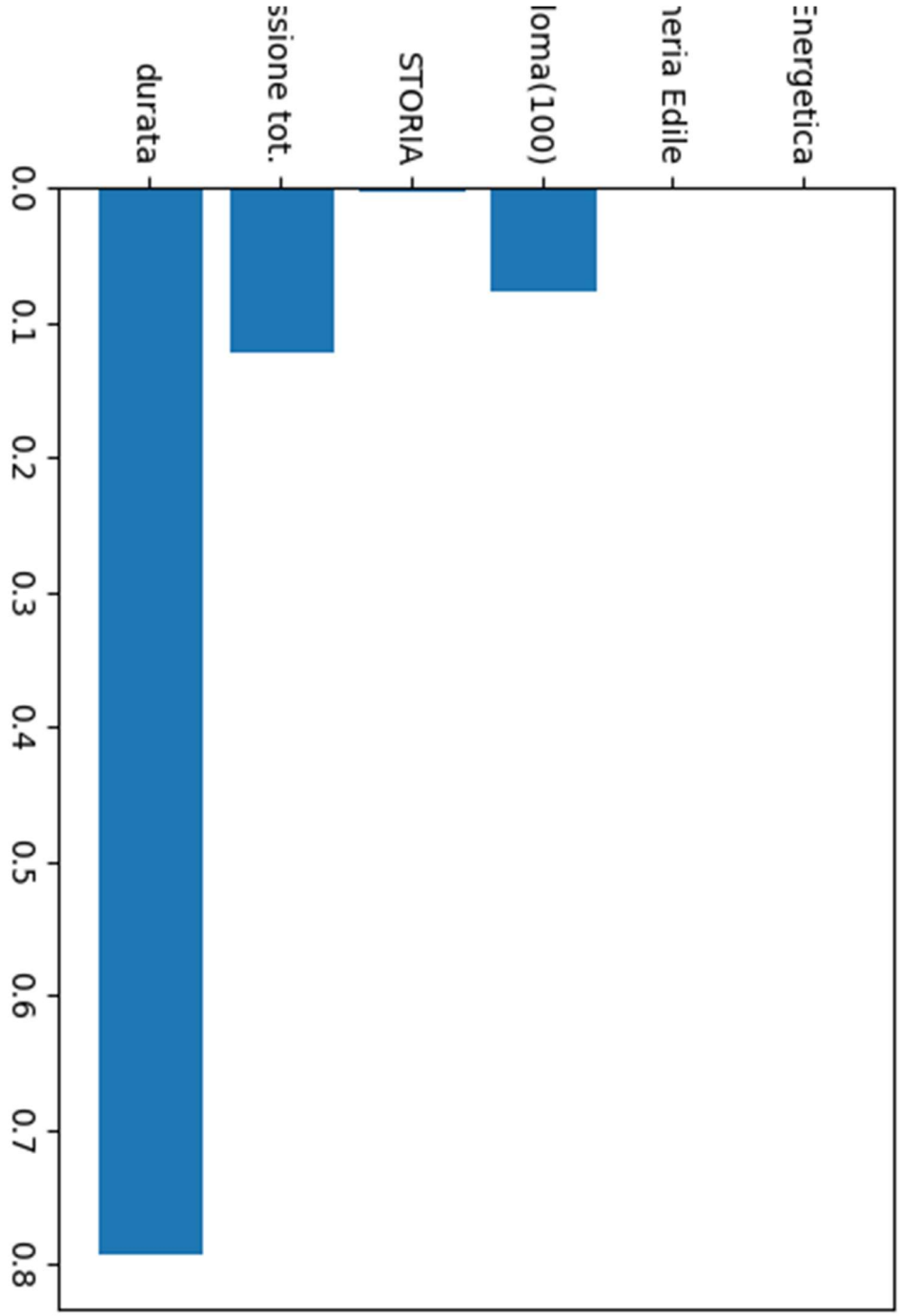


Figura 33 importanza variabili per albero di regressione

Un risultato simile è ottenuto con le random forest.

In tutti e tre i casi gli alberi sono più alti e più variabili vengono incluse, fra cui il genere, mantenendo le tre variabili più importanti nello stesso ordine.

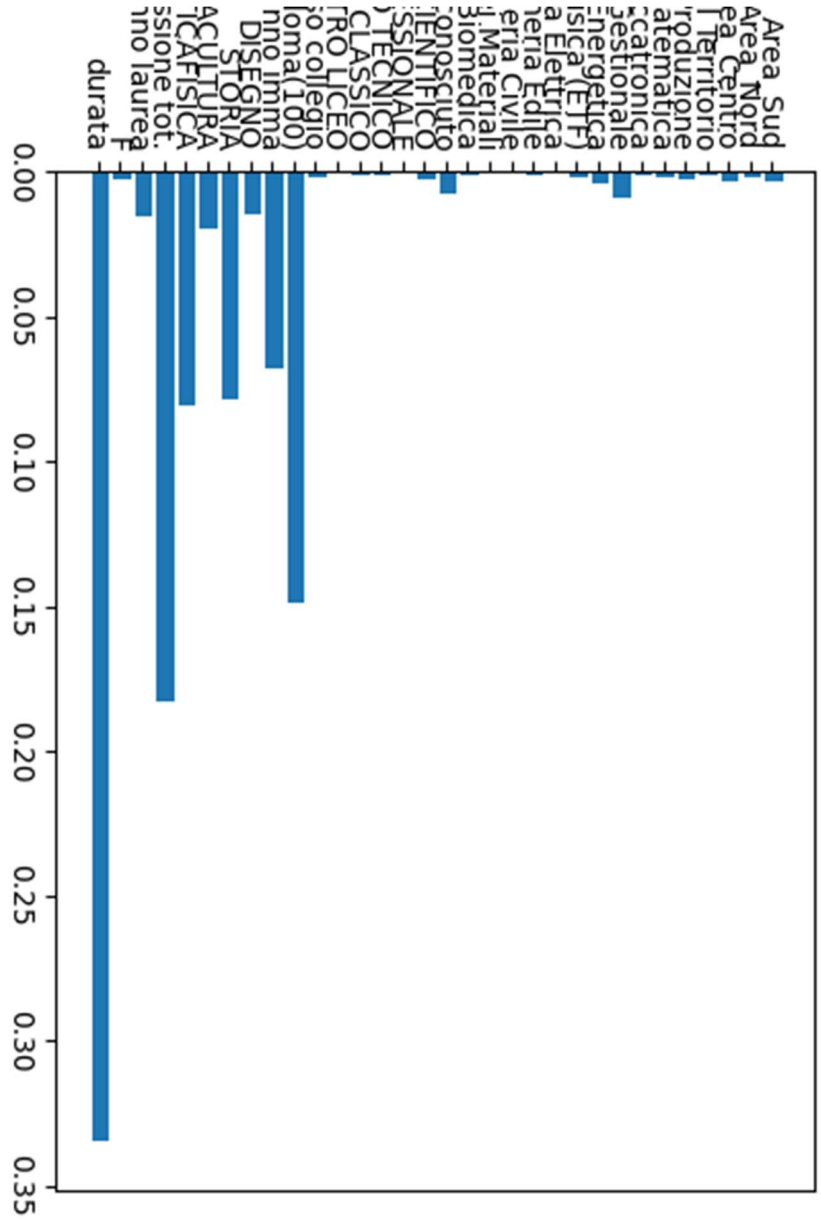


Figura 34 importanza variabili per la random forest, classificazione basato sui quantili

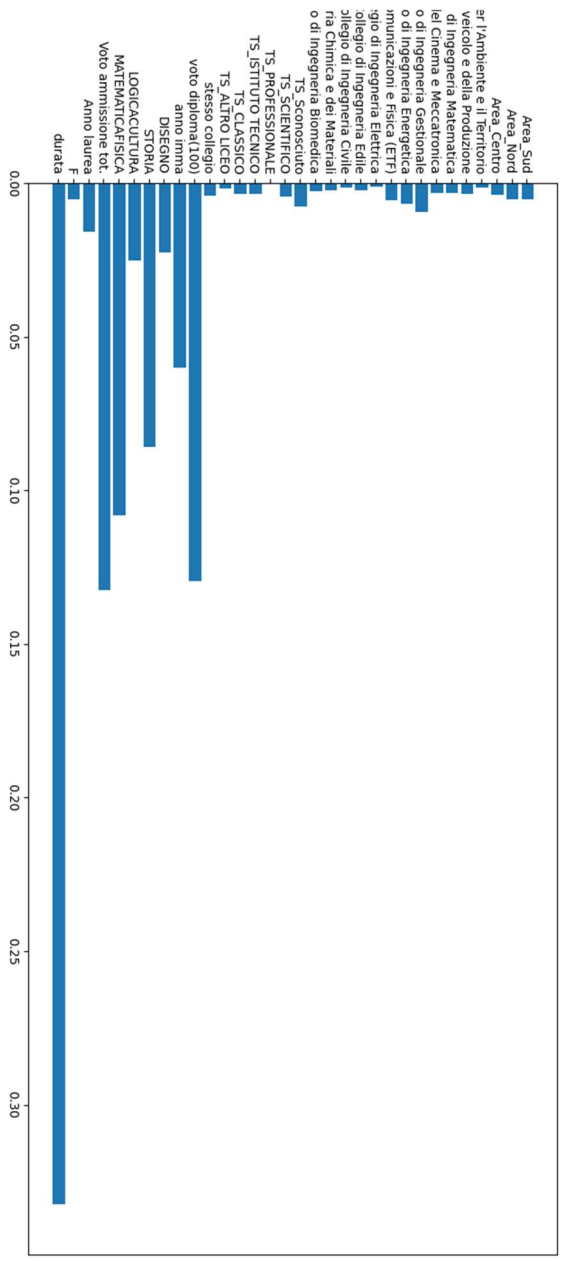


Figura 35 importanza variabili per la random forest, classificazione basato sull'esperto

In tutti e tre i casi, il genere compare ma non è fra le variabili più importanti – in generale, sembra essere al livello dell'area.

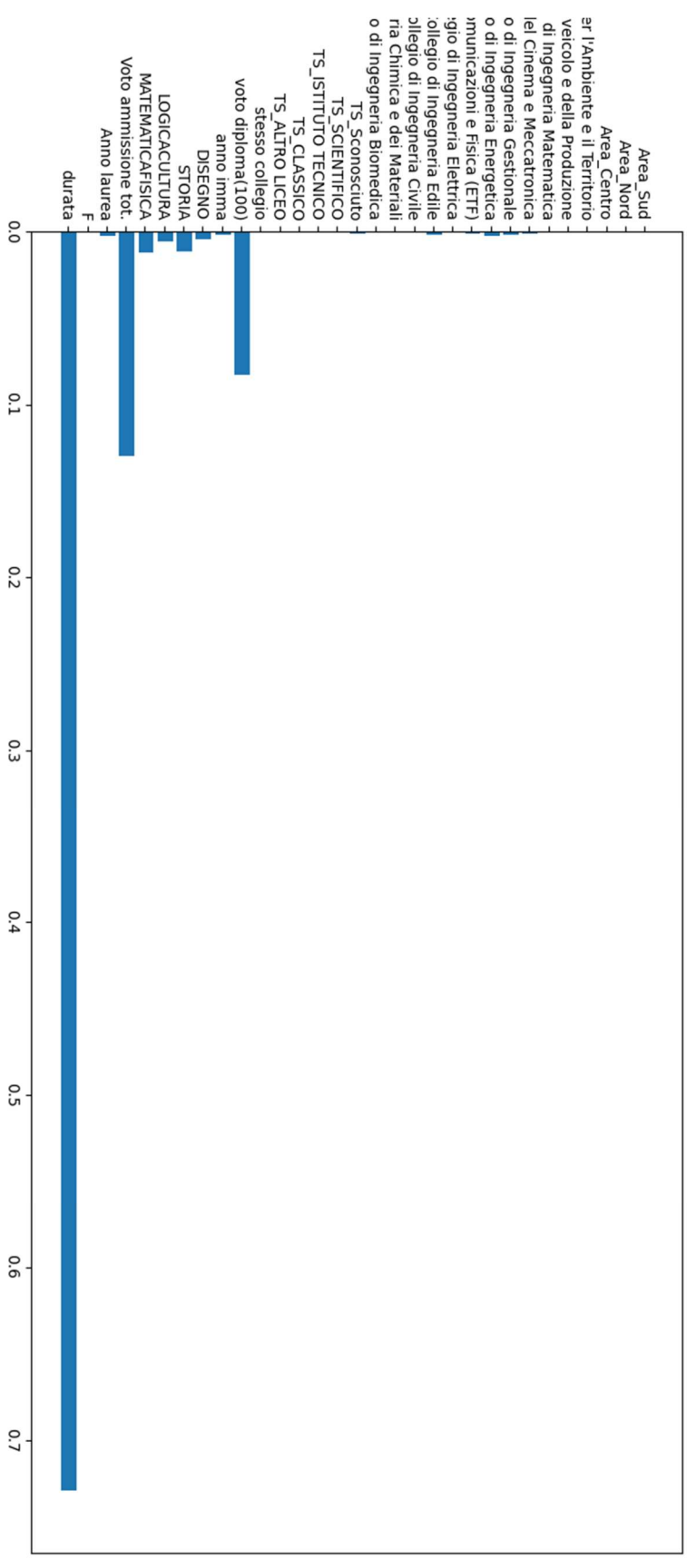


Figura 36 37 importanza variabili per la random forest, regressione

La precisione ottenuta è leggermente più alta - rispettivamente a 49% e 56% per le variabili categoriche e 0.61 per le continue.

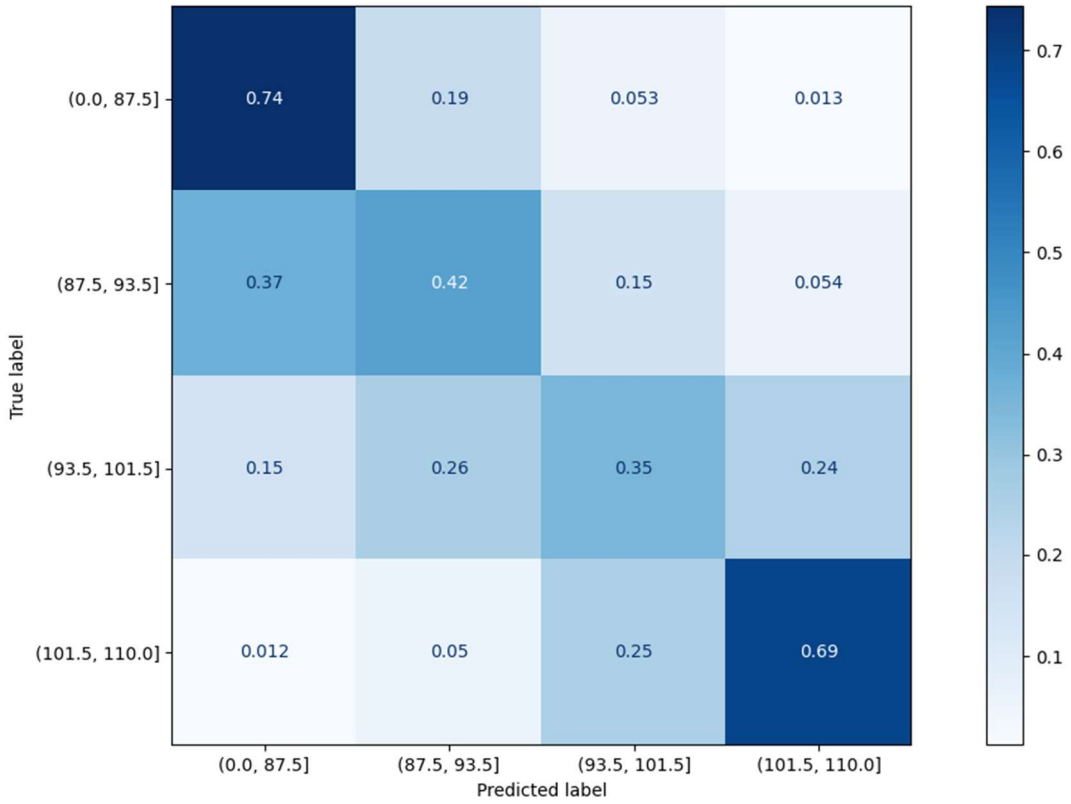


Figura 38 Matrice di Confusione per foresta, classificazione basata sui quantili.

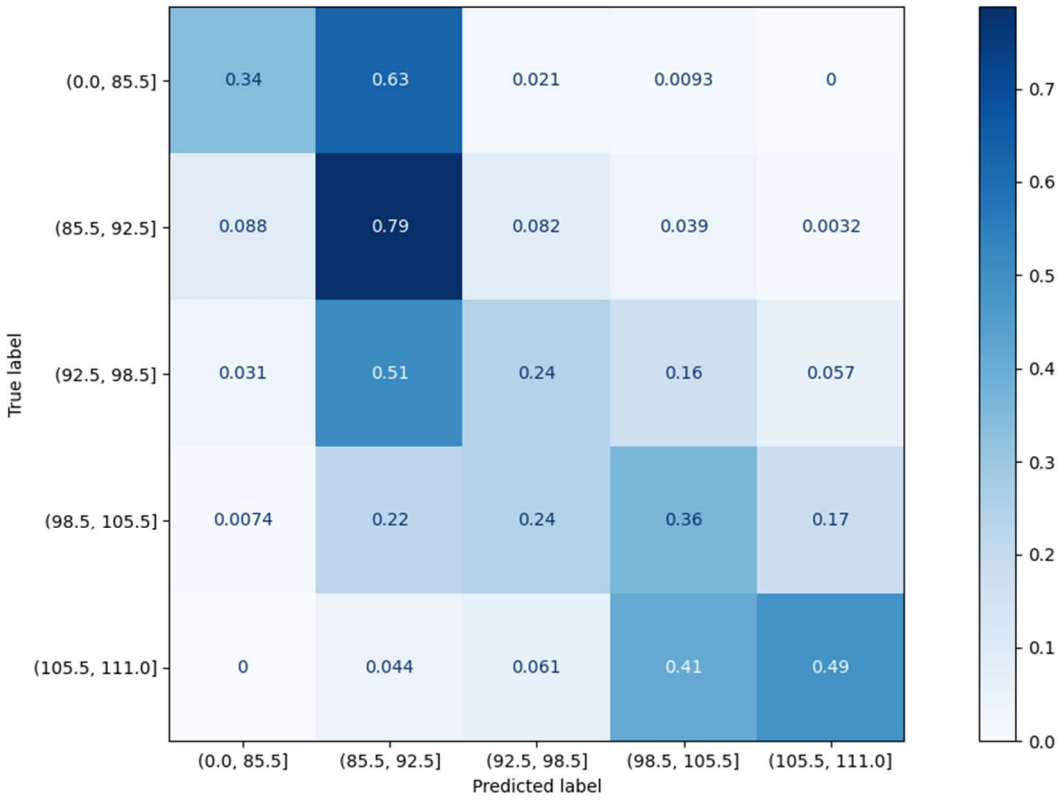


Figura 39 Matrice di Confusione per foresta, classificazione basata sul consiglio dell'esperto.

Nonostante questa teorica maggiore precisione, osservando Figura 39, è evidente che nel caso della classificazione esperta la maggior parte dei dati sono classificati come appartenenti alla seconda classe – probabilmente a causa della maggiore popolazione di essa.

Nel caso dell'SVN per entrambe le classificazioni viene scelto il metodo OVO e quasi sempre il kernel lineare. Per selezionare le variabili più importanti utilizziamo una funzione di python che per ciascuna permuta i valori e valuta quanto questo causa la precisione di variare. Anche in questo caso, utilizziamo un grafico a barre per mostrare il risultato.

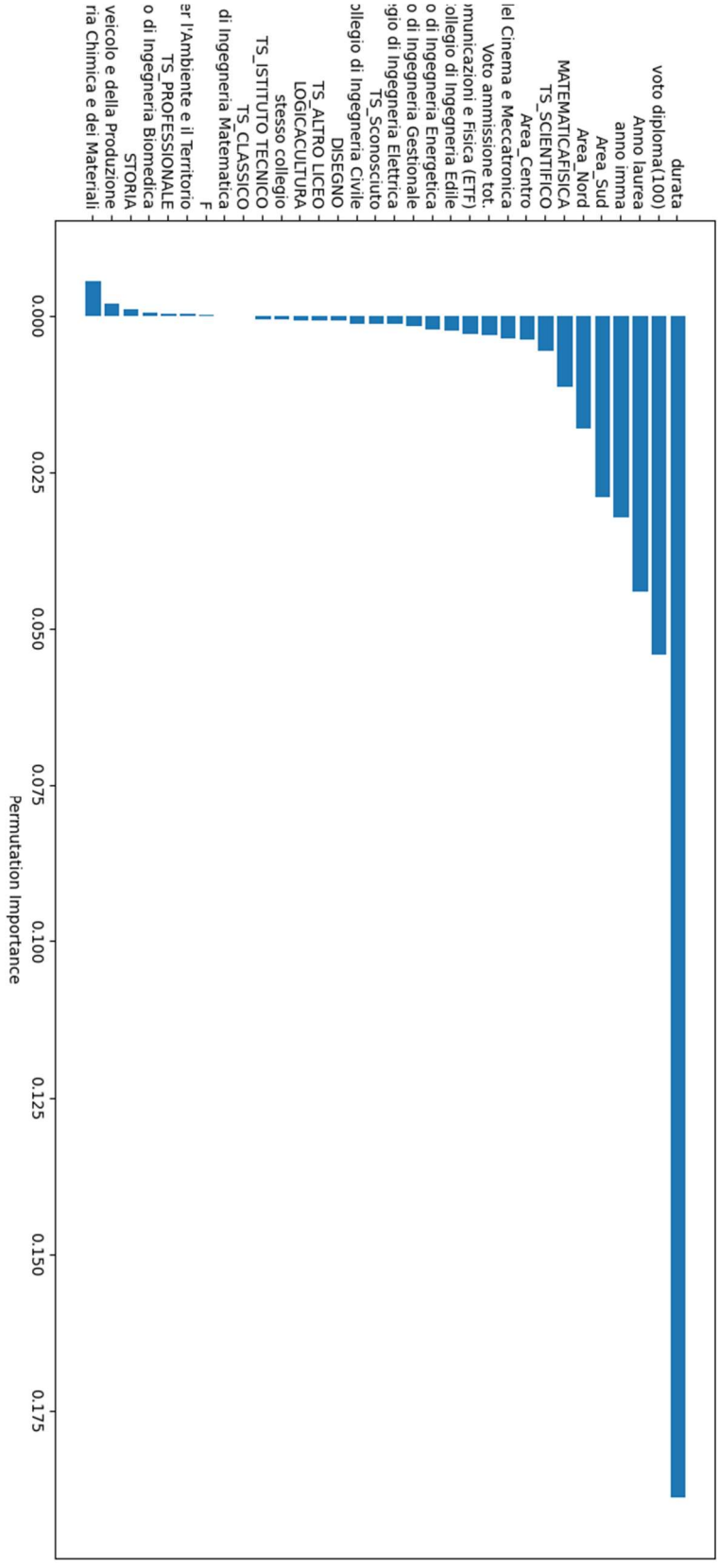


Figura 40 importanza variabili per SVM, classificazione basato sui quantili

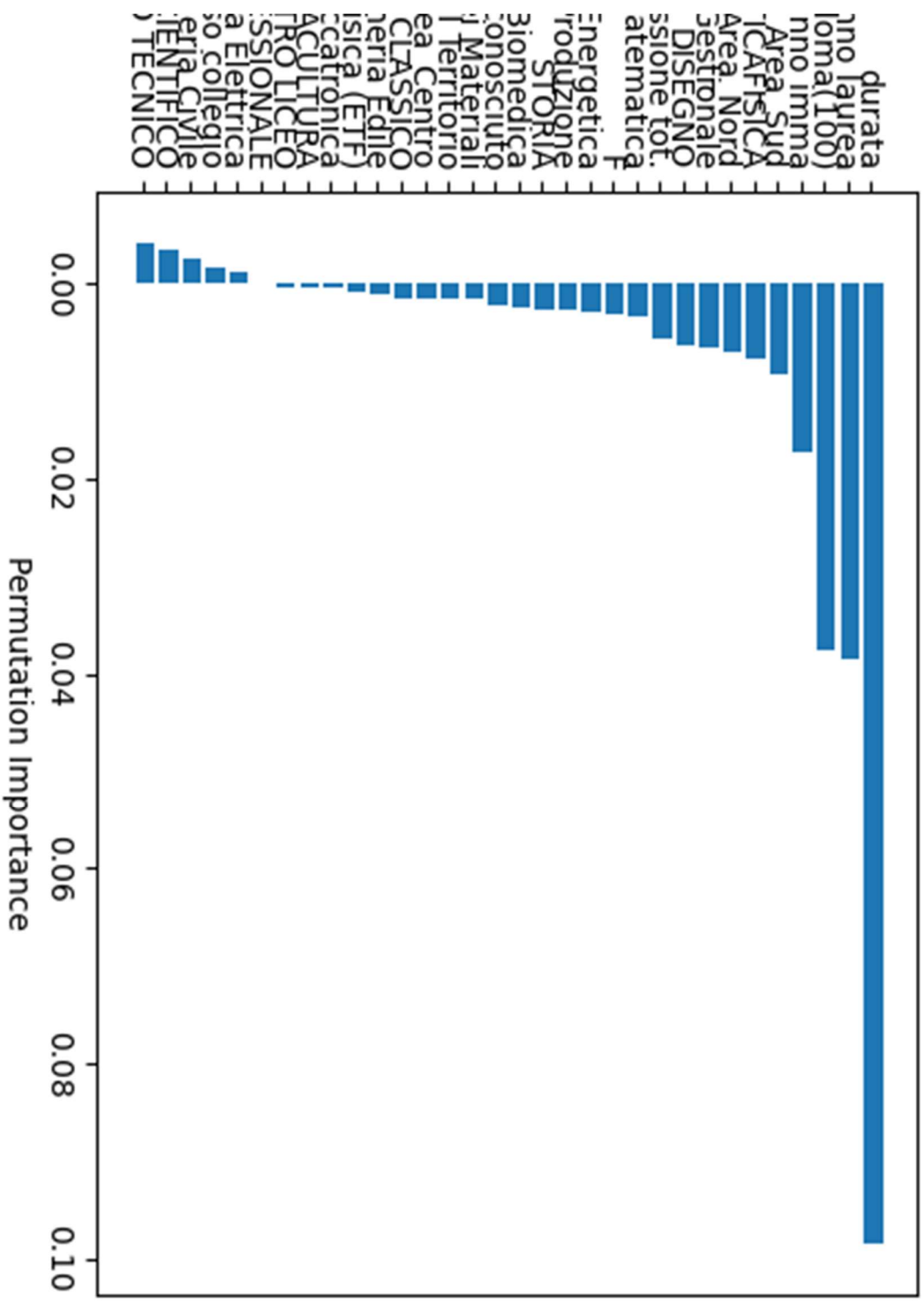


Figura 41 importanza variabili per SVM, classificazione basato sull'esperto

In entrambi i casi, la durata è ancora la variabile più importante – seguita dall'anno di laurea e dal voto di diploma (non necessariamente in questo ordine). Il genere non è

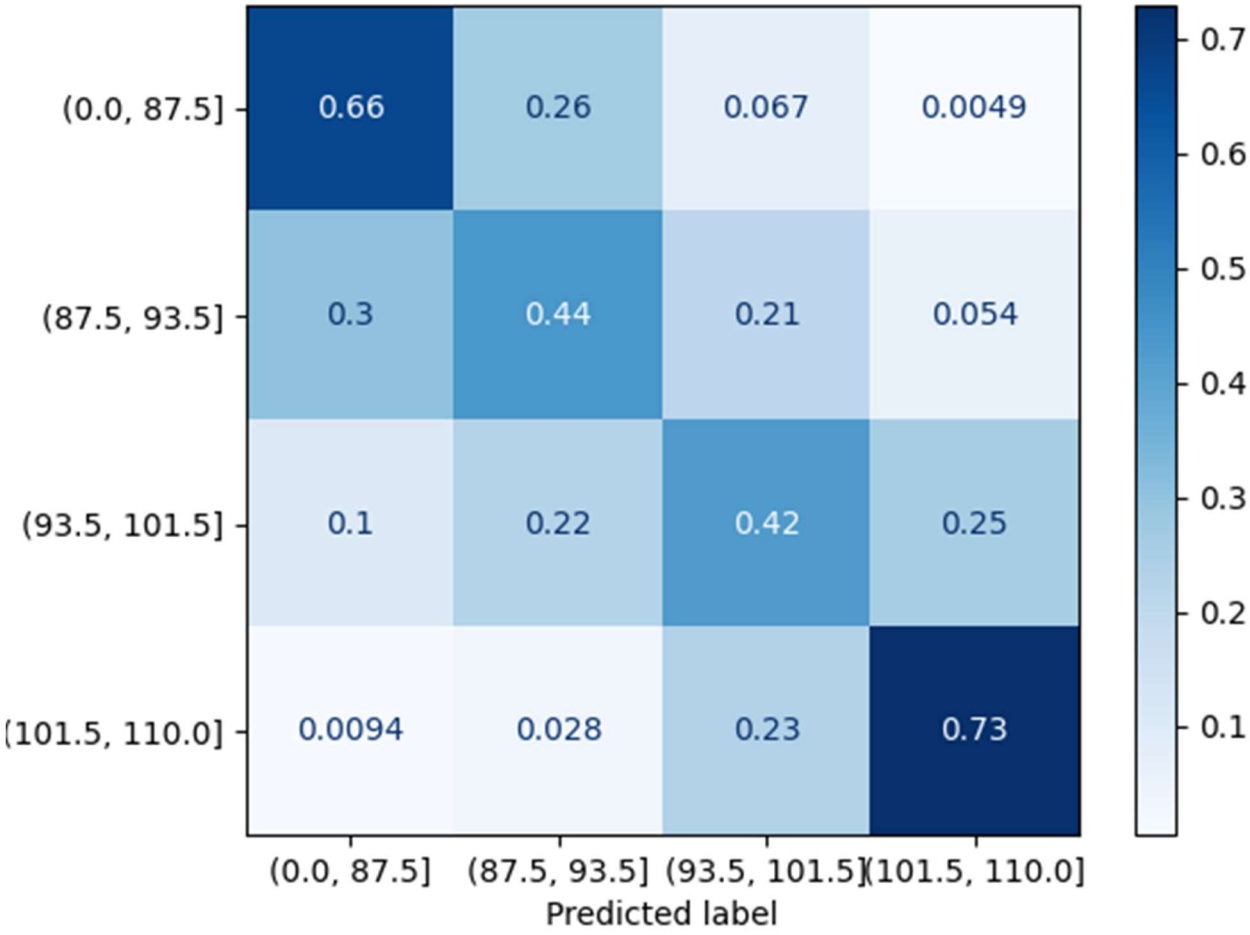


Figura 42 Matrice di Confusione per SVM, classificazione basata sui quantili.

importante nella suddivisione basata sui quantili, ma lo è in quella basata sul consiglio dell'esperto.

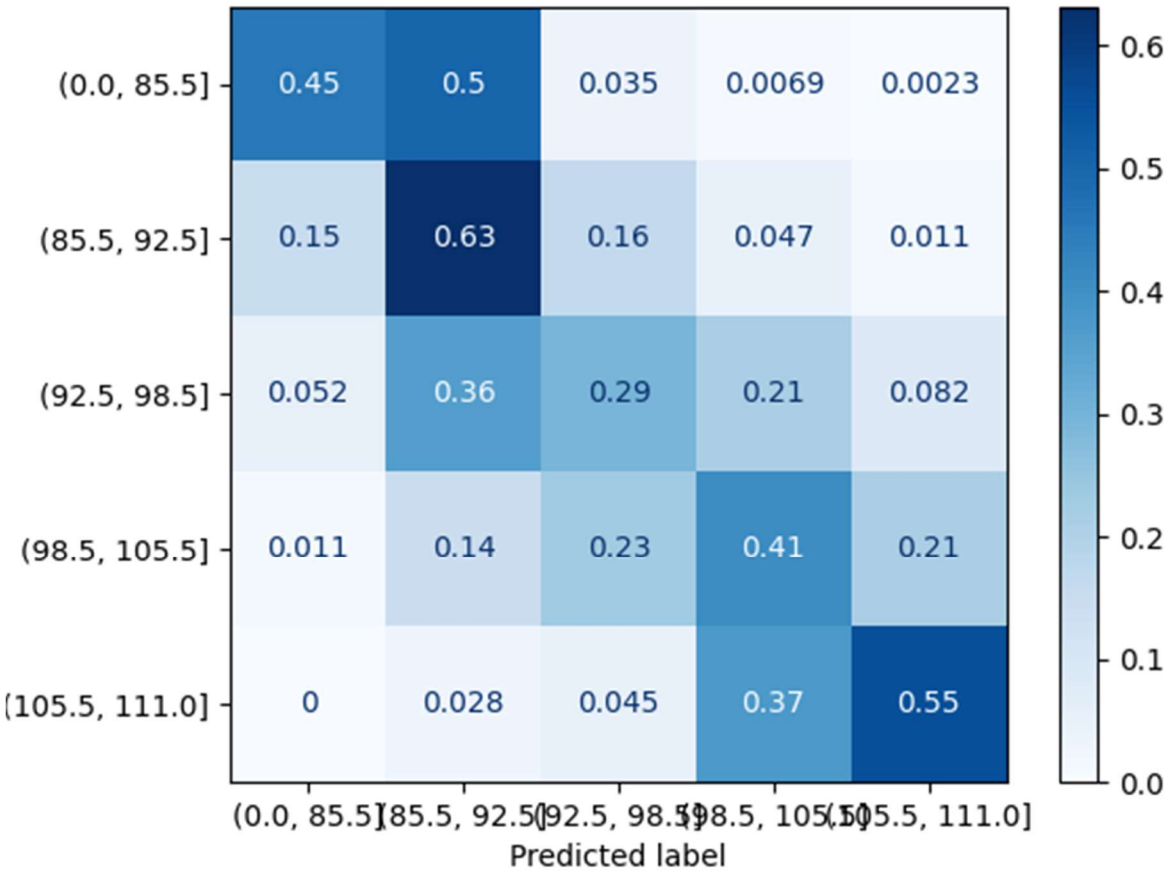


Figura 43 Matrice di Confusione per SVM, classificazione basata sul consiglio dell'esperto.

Si è ottenuta una precisione del 56% per il quartile e del 48% per la classificazione da esperto. Anche in questo caso, vi è una tendenza alla misclassificazione nella seconda categoria, anche se meno rispetto alla foresta.

5 Conclusioni

Nonostante la presenza di diverse strategie per incoraggiare le donne ad intraprendere un percorso di formazione in ambito STEM, ancora persistono forti sbilanciamenti in termini di numerosità. Questo lavoro di tesi è stato focalizzato sull'analisi dei dati relativi ad alcuni percorsi formativi universitari nelle discipline STEM al fine di analizzare quali variabili influenzano maggiormente il voto finale delle persone laureate, e se l'identità di genere ha delle influenze o meno. Nell'attività di analisi descritta in questa tesi si focalizza su un insieme di dati caratterizzanti un campione di studenti e studentesse in termini di votazione conseguita al diploma di maturità, test di ammissione e alcune informazioni relative alla carriera universitaria nelle discipline STEM. Il processo di analisi, tipico del KDD (Knowledge Discovery From Data), comprende la pulizia dei dati, una prima fase di esplorazione e caratterizzazione delle distribuzioni in termini di metriche statistiche (e.g., medie, varianze) nonché l'utilizzo di alcuni algoritmi di data mining (e.g., alberi di decisione, random forest, SVM e regressione lineare) per la modellazione della relazioni principali tra le variabili di input (quali ad esempio il percorso di formazione, i risultati di performance, e alcune caratteristiche della popolazione sotto osservazione) e il voto conseguito alla laurea triennale.

Le performance dei modelli predittivi, stimati valutando un indice statistico (e.g. R2), non sono ottimali ma in qualche caso evidenziano che la variabile genere è tra quelle che influenzano la predizione ma non è mai fra le più rilevanti. Nello specifico, nei modelli di Support Vector Machine (SVM) il genere è relativamente importante per almeno una classificazione, mentre per altri modelli, quali gli alberi di decisione e random forest, tra le variabili principali troviamo il voto di diploma e il punteggio conseguito al test di ammissione (che sappiamo dalla letteratura e dalla nostra analisi preliminare che può essere correlata con il genere) e la durata (risultata la variabile più importante in ogni caso).

Sulla base dei risultati ottenuti, non è possibile evidenziare una relazione diretta tra il genere e il voto di laurea. Tuttavia si suggerisce di estendere lo studio con: (i) un set di

variabili aggiuntivi in grado di rappresentare correttamente le relazioni principali che esistono tra percorsi di studi e performance in un'ottica di genere; (ii) un campione più rappresentativo dell'intera popolazione, relativo a più università e con dati relativi ad un periodo di osservazione di almeno 10 anni.

Bibliografia

[1] Guiloppé e Roy, *A Global Approach to the Gender Gap in Mathematical, Computing, and Natural Sciences: How to Measure It, How to Reduce It?,* 2020.

[2] Syagin, *Gender Bias in Standardized Tests: Evidence from a Centralized College Admissions System,* 2018.

[3] Kling, Noftle e Robins, *Why Do Standardized Tests Underpredict Women's Academic Performance? The Role of Conscientiousness,* 2012.

[4] Atzeni, Ceri, Fraternali, Paraboschi e Torlone, *Basi di dati,* McGraw Hill, 2018.

[5] Golfarelli e Rizzi, *Data warehouse: teoria e pratica della progettazione,* McGraw Hill, 2006.

[6] Tan, Steinbach, Karpatne e Kumar, *Introduction to data mining,* Pearson, 2019.

[7] Ramakrishnan e Gehrke, *Database Management Systems,* McGraw Hill, 2003.

[8] Chodorow e Bradshaw, *MongoDB: The Definitive Guide (Powerful and Scalable Data Storage),* O'Reilly Media, 2018.

[9] IBM Cloud Education, «Data Modeling,» 2020. [Online]. Available: https://www.ibm.com/cloud/learn/data-modeling.

[10] «Documentazione NumPy,» [Online]. Available: https://numpy.org/.

[11] «Documentazione pandas,» [Online]. Available: https://pandas.pydata.org/.

[12] «Documentazione scikit-learn,» [Online]. Available: https://scikit-learn.org/stable/.

[13] «Documentazione Matplotlib,» [Online]. Available: https://matplotlib.org/.