UNIVERSITÀ
DI PARMA

Università degli studi di Parma

DIPARTIMENTO DI SCIENZE MATEMATICHE, FISICHE E INFORMATICHE
Corso di Laurea Triennale in Informatica

Tesi di Laurea

PREVISIONI SULLE CARRIERE DEGLI STUDENTI: UN APPROCCIO BASATO SU MACHINE LEARNING

Student career prediction: a machine learning approach

Candidato:
Nicolò Bertoli

Relatore:
Prof. Flavio Bertini

Correlatore:
Prof. Alessandro Dal Palù

Anno Accademico 2021-2022

Indice

Introduzione

Ogni anno, i corsi di studio universitari compilano la "**Scheda di monitoraggio Annuale**", un documento di autovalutazione atto a verificare il raggiungimento degli obiettivi di apprendimento che il corso si era posto, così da individuare le cause di eventuali risultati insoddisfacenti, e attuare le opportune misure di correzione e miglioramento.

All'interno del documento, il corso di studi discute il valore di 37 indicatori, che danno una rappresentazione dei risultati raggiunti in 5 ambiti: "didattica", "internazionalizzazione", "regolarità delle carriere", "soddisfazione e occupabilità degli studenti", "consistenza e qualificazione del corpo docente".

Tra gli indicatori relativi alla didattica, uno dei più rilevanti è l'**indice iC16**, definito come la percentuale degli studenti iscritti al I anno che proseguono al II nello stesso corso di studi, avendo conseguito almeno 40 CFU (i.e., Crediti Formativi Universitari).

Massimizzare l'iC16 è di grande interesse per le università, in quanto equivale a migliorare il rendimento degli studenti al I anno, e a ridurre i relativi abbandoni universitari. Dato che il primo anno d'iscrizione a un ateneo è quello più critico per la maggior parte degli studenti [Contini et al., 2017], migliorare le condizioni dei neo-immatricolati assume un'importanza particolare.

Visto che questo indice può essere calcolato in modo esatto solo al termine del I anno, ai corsi di studio potrebbe essere utile avere uno strumento che riesca a stimarlo in anticipo, in modo da poter capire se e in quale misura sia necessario attuare misure migliorative.

L'obiettivo di questa tesi sarà quello di realizzare un sistema capace di prevedere con la massima precisione possibile l'indice iC16 per il Corso di

Laurea Triennale in Informatica dell'Università di Parma.

Per realizzare il sistema finale, verranno addestrati due modelli predittivi, il cui utilizzo combinato permetterà di effettuare previsioni sull'iC16. Dato un certo insieme di studenti, il primo modello avrà l'obiettivo di prevedere quanti di questi conseguiranno almeno 40CFU a fine anno, mentre il secondo si occuperà di stimare quanti si iscriveranno al II anno.

La pipeline di lavoro seguita è mostrata in figura 1.

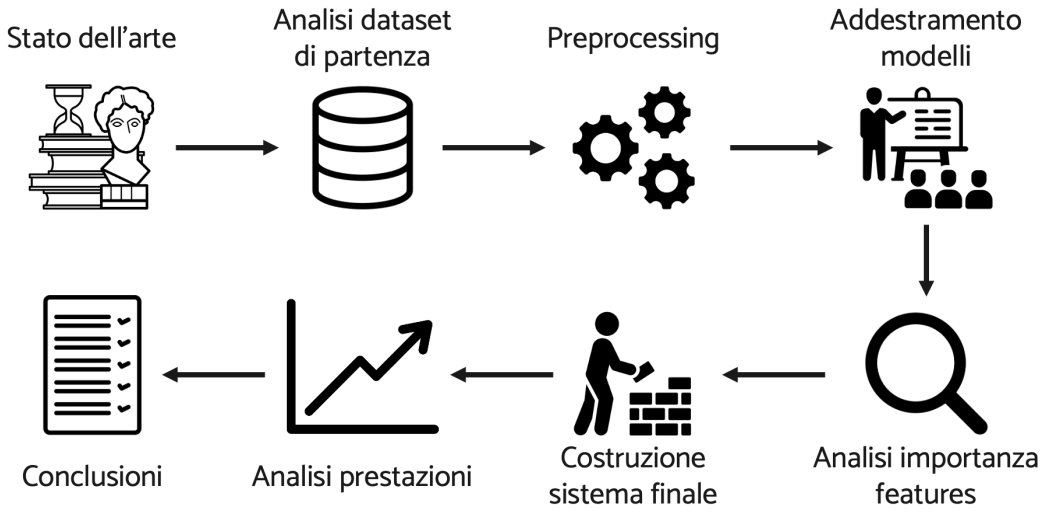

Figura 1: pipeline di lavoro seguita.

Capitolo 1

Stato dell'arte

I principali motori di ricerca nel campo della letteratura scientifica non hanno restituito risultati riguardanti la previsione dell'iC16, sono invece stati trovati numerosi trattati sulla previsione dei risultati accademici e degli abbandoni universitari.

1.1 Previsione risultati accademici

A livello internazionale, la maggior parte delle ricerche in questo ambito esprime i "risultati" conseguiti dagli studenti in termini di voti ottenuti o di esami superati. I modelli che si sono dimostrati più efficaci per questo tipo di previsione sono:

- rete neurale [Lykourentzou et al., 2009], [Nuankaew et al., 2020], [Kabakchieva, 2012];
- random forest classifier [Gkontzis et al., 2018];
- decision treee [Pandey and Sharma, 2013], [Kabakchieva, 2012];
- gradient boosting classifier [Jaiswal et al., 2021].

Tutti gli studi citati utilizzano come input per i modelli le **performance universitarie** degli studenti (voti conseguiti, numero di esami superati, numero di tentativi fatti per superare ogni esame, ...). Dati di questo tipo hanno dimostrato un impatto molto positivo sulle capacità predittive dei sistemi, ma per essere raccolti richiedono che gli studenti abbiano già sostenuto alcuni esami, quindi non permettono di effettuare previsioni prima dell'inizio degli esami.

La maggior parte degli studi analizzati (tutti tranne [Lykourentzou et al., 2009]) affianca alle features precedenti dati relativi alle **condizioni socio-demografiche** degli studenti (età, cittadinanza, genere, residenza, tipologia della scuola di provenienza, informazioni sull'andamento dello studente alle superiori, ...). Oltre a migliorare le performance dei sistemi predittivi, queste features permettono di effettuare previsioni già al termine delle immatricolazioni.

Esistono poi alcuni sistemi che utilizzano features particolari, come quello sviluppato in [Ilgan, 2013], che prevede i voti degli studenti nel corso di "Metodo di ricerca scientifica" in base all'opinione (espressa in termini numerici) che questi hanno su vari aspetti del corso, come l' interesse negli argomenti trattati, il gradimento delle metodologie d'insegnamento, ...

1.2 Previsione abbandoni universitari

La maggior parte degli articoli in questo ambito utilizza modelli analoghi a quelli utilizzati per la previsione dei risultati accademici. Alcuni esempi sono: [Del Bonifro et al., 2020], che mostra l'efficacia del decision tree, [Nuankaew et al., 2020], [Tan and Shao, 2015], che mostrano i risultati ottenuti sfruttando le reti neurali, e [Utari et al., 2020], che utilizza un classificatore random forest. Tutte queste ricerche utilizzano features sia relative alle performance universitarie degli studenti, che alle loro condizioni socio-demografiche.

A livello italiano, un sistema predittivo molto interessante è quello sviluppato dall'**Osservatorio Abbandoni di Cineca**, che permette ai corsi di studio di stimare le probabilità di abbandono accademico per i propri studenti, tramite un portale online [Macaluso, 2018], .

Il "Datamart Studenti", che raccoglie dati relativi agli studenti di 69 università italiane, ha permesso addestrare il sistema sviluppato dall'Osservatorio con numerose features, che si dividono in tre tipologie:

- socio-demografiche (età, cittadinanza, genere, residenza, tipo scuola di provenienza, voto di maturità, ...);
- caratteristiche della carriera (tipo di corso seguito, presenza test d'ingresso, dipartimento, ...);
- performance universitarie precedenti (CFU acquisiti, esami superati, media voti, numero anni fuori corso, ...).

Il sistema effettua le proprie previsioni sfruttando 3 modelli:

- decision tree;
- random forest;
- gradient boosting classifier;

e restituisce come probabilità di abbandono finale una media dei risultati restituiti dai singoli modelli.

Capitolo 2

Background

2.1 Ambiente di lavoro

Il progetto esposto in questa tesi è stato sviluppato utilizzando il linguaggio Python [Van Rossum and Drake, 2009], e alcune librerie di base utilizzate comunemente per lo sviluppo di sistemi di machine learning:

- **scikit-learn** permette di implementare in modo semplice numerosi modelli di machine learning, impostando i relativi iperparametri a piacere, ed è alla base di altre librerie utilizzate, come TPOT e SMOTE (vedi sezioni 2.1.2 e 2.1.1) [Pedregosa et al., 2011];
- **numpy** implementa diverse funzioni per il calcolo scientifico, e in particolare per la manipolazione efficiente di array e matrici [Harris et al., 2020];
- **pandas** permette di gestire facilmente tabelle di grandi dimensioni [pandas development team, 2020]. Questa libreria, assieme a numpy, è stata fondamentale per la fase di preprocessing;
- **matplotlib** fornisce supporto per la generazione di grafici, la maggior parte di quelli esposti in questa tesi sono stati generati grazie a questa libreria [Hunter, 2007].

Altre librerie che si sono rivelate utili sono state **tensorflow** [Abadi et al., 2015] e **keras** [Chollet et al., 2015], che forniscono metodi per la costruzione e l'addestramento di reti neurali. Le reti testate nella sezione 4.2 sono state implementate grazie a questi strumenti.

Nelle sottosezioni successive verranno approfondite alcune altre librerie che hanno avuto un ruolo centrale nello sviluppo del progetto.

2.1.1 SMOTE

Addestrare modelli di machine learning su dataset sbilanciati è una pratica sconsigliata, in quanto generalmente porta i modelli a predirre più spesso la classe maggioritaria. Una possibile soluzione a questo problema è **l'oversampling**, che consiste nell'aumentare le dimensioni della classe minoritaria fino ad avere un dataset bilanciato.

La forma più semplice di oversampling prende il nome di **random oversampling**, e consiste nel duplicare elementi della classe meno rappresentata in modo casuale. Questo approccio può portare a overfitting sui dati di training, e quindi generalmente è sconsigliato.

Un approccio molto più diffuso ed efficace è chiamato **Synthetic Minority Oversampling Technique** (SMOTE), e prevede di creare delle copie degli elementi della classe minoritaria, che vengono poi perturbate casualmente, in modo da sintetizzare nuovi elementi, tutti diversi da quelli originali [Brownlee, 2020].

Imbalanced-learn è una libreria opensource basata su scikit-learn (vedi sezione 2.1), che permette di bilanciare facilmente un dataset utilizzando varie tecniche, tra cui SMOTE [Lemaître et al., 2017]. Per farlo basta creare un'istanza della classe "imblearn.over_sampling.SMOTE", e utilizzarla per chiamare il metodo "fit_resample", passandogli il dataset che si desidera bilanciare, come mostra il codice 2.1

```
1 import imblearn
2 from imblearn.over_sampling import SMOTE
3
4 # creazione istanza classe
5 smote = SMOTE()
6
7 # bilanciamento dataset
8 X_bilanciato, y_bilanciato = smote.fit_resample(X,y)
```

Codice 2.1: esempio di bilanciamento di un dataset con imbalanced-learn.

2.1.2 TPOT

Tree-Based Pipeline Optimization Tool (TPOT) è una libreria basata suscikit-leran (vedi sezione 2.1), che permette di esplorare in modo automatico migliaia di possibili pipeline di modelli di machine learning, per trovare la più adatta a risolvere un certo problema classificazione o

regressione. Quando la ricerca termina, viene restituito il codice Python associato alla migliore pipeline trovata [Olson et al., 2016].

TPOT prende in considerazione più modelli di apprendimento automatico, e li ordina in vari modi all'interno di pipeline, utilizzando per ciascuna diverse tecniche di preprocessing, e provando a impostare diversi valori per gli iperparametri di ogni modello. Di conseguenza la ricerca di un modello ottimale può richiedere diverse ore, o addirittura giorni per terminare, a seconda delle dimensioni del dataset e delle impostazioni di ricerca scelte. Nel caso delle esecuzioni fatte per individuare le pipeline della sezione 4.2, in cui sono state mantenute le impostazioni di ricerca di default, TPOT ha impiegato circa un giorno per restituire i risultati di ogni ricerca.

Dato che l'algoritmo di ricerca utilizzato da questa libreria ha una componente di randomicità, diverse ricerche sullo stesso dataset possono portare all'individuazione di pipeline ottimali diverse, per questa ragione nella sezione 4.2 sarà presente anche più di una sola pipeline per ogni modello.

TPOT permette di cercare la pipeline ottimale per un certo problema di classificazione creando un'istanza della classe "TPOTClassifier", e utilizzandola per chiamare il metodo "fit", al quale devono essere passati i dati di training, come mostra il codice 2.2. Per cercare pipeline che sfruttano modelli di regressione invece, basta utilizzare la classe "TPOTRegressor" al posto al posto di "TPOTClassifier".

```
1  import tpot
2  from tpot import TPOTClassifier
3
4  # creazione istanza classe
5  pipeline_optimizer = TPOTClassifier()
6
7  # ricerca pipeline ottimale
8  tpot.fit(X_train, y_train)
9
10 # salvataggio codice Python associato alla pipeline
11 tpot.export("best_pipeline.py")
```

Codice 2.2: esempio di ricerca pipeline ottimale tramite TPOT.

2.1.3 LIME

Per poter fare affidamento su un certo sistema d'intelligenza artificiale, è fondamentale riuscire a motivare le sue scelte. **Local Interpretable Model-agnostic Explanations** (LIME) è una libreria che permette di spiegare le decisioni di un qualsiasi modello di apprendimento automatico di tipo black box, approssimandolo con un sistema più semplice. L'idea alla base della libreria è quella di modificare i valori delle features, per capire in che modo questi cambiamenti vadano a condizionare le previsioni finali.

La classe "LimeTabularExplainer", implementata da LIME, può essere utilizzata per chiamare il metodo "explain_instance", al quale è necessario passare un elemento del dataset e la funzione "predict_proba" associata al modello d'interesse, come mostrato nel codice 2.3. Verrà restituita una lista, contenente l'impatto di ogni feature sulla previsione del modello. Sfruttando questo metodo su più input, è possibile calcolare l'impatto medio di ogni feature sulle previsioni del sistema.

```
import lime
from lime import lime_tabular

# creo istanza di LimeTabularExplainer
explainer = lime_tabular.LimeTabularExplainer(
    training_data = X_train,
    mode = "classification"
)

# ottengo spiegazioni
exp = explainer.explain_instance(
    data_row = X_test.iloc[1],
    predict_fn = model.predict_proba
)
```

Codice 2.3: esempio estrazione importanza features relative a una previsione tramite LIME.

2.2 Tipologia sistemi di ML utilizzati

Esistono numerose tipologie di modelli di machine learning, individuare quella più adatta a implementare i due modelli prestabiliti è un importante primo passo verso la costruzione del sistema finale.

Una prima classificazione può essere fatta distinguendo tra modelli con apprendimento supervisionato e non supervisionato. I sistemi con

apprendimento supervisionato permettono di fare **previsioni**, e richiedono di essere addestrati tramite l'utilizzo di **dati storici etichettati**, ovvero input per i quali si conosce a priori quale dovrebbe essere la relativa previsione. I sistemi con **apprendimento non supervisionato**, invece, non cercano di trovare una relazione tra predittori e previsione, non permettono di effettuare previsioni di alcun tipo, e non richiedono l'impiego di dati etichettati; servono invece per individuare forme di organizzazione e rappresentazione precedentemente sconosciute all'interno dei dati di input. Dato che i due modelli prefissati hanno lo scopo di fare previsioni, rientrano nella prima categoria.

I sistemi con apprendimento non supervisionato si dividono a loro volta in varie tipologie, le due principali sono i modelli di classificazione e quelli di regressione. I modelli di **classificazione** restituiscono un output categorico ristretto a un numero finito di possibili classi, per esempio un riconoscitore di immagini capace di distinguere le mele dalle arance verrà probabilmente implementato tramite un modello appartenente a questa categoria. I modelli di **regressione** invece producono un output nel continuo, che può assumere un numero infinito di possibili valori. Per esempio, l'implementazione di un predittore per il costo del petrolio nel prossimo mese richiede l'utilizzo di un modello di regressione.

Come mostrato in figura 4.1, i modelli necessari per la previsione dell'iC16 devono effettuare classificazioni categoriche binarie, quindi i più adatti a implementarli sono i modelli di classificazione. Tuttavia classificazione e regressione sono strettamente correlate, risulta infatti possibile trasformare un problema di regressione in uno di classificazione e viceversa. In seguito a queste considerazioni, la ricerca dei modelli si concentrerà principalmente su quelli di classificazione, ma verranno eseguiti anche alcuni test usando quelli di regressione.

I modelli di regressione saranno adattati ai problemi affrontati secondo queste metodologie:

- nel caso della previsione dei 40 CFU, saranno addestrati per prevedere quanti crediti formativi lo studente di input avrà ottenuto a fine anno (valore nel continuo tra 0 e 60). Verrà poi costruita un' etichetta con valore positivo nel caso la previsione indichi il raggiungimento di almeno 40CFU, e negativo altrimenti;
- nel caso della previsione dell'iscrizione all'anno successivo, saranno addestrati per prevedere un valore nel continuo tra 0 e 1, che sarà

poi arrotondato per ottenere un valore di output categorico.

2.3 Metriche valutazione modelli di ML

Un passo fondamentale in ogni progetto nel campo del machine learning è quello di valutare il modello ottimale individuato per la gestione del problema di interesse. A questo scopo, si possono considerare numerose metriche; per la valutazione dei modelli individuati nel capitolo 4.3.2 verranno utilizzate quelle approfondite di seguito, che sono tra le più utilizzate:

- **Accuracy**: descrive in maniera generale la capacità del modello di classificare correttamente gli input, indipendentemente dalla classe a cui appartengono. Questa metrica risulta particolarmente utile quando le classi hanno tutte la stessa importanza, e può invece risultare ingannevole nel caso si abbia un dataset sbilanciato: un modello che prevede sempre la stessa classe, testato su un dataset di elementi appartenenti unicamente a quella classe avrà un'accuracy del 100%, ma non si può certo dire che il modello sia valido.

$$\text{Accuracy} = \frac{\text{previsioni corrette}}{\text{previsioni effettuate}}$$

- **Precision**: descrive l'affidabilità delle previsioni positive: tanto più la precision è elevata, tanto minore sarà il numero di falsi positivi.

$$\text{Precision} = \frac{\text{true positive}}{\text{positive predictions}}$$

- **Recall**: descrive la capacità del sistema di classificare correttamente gli elementi appartenenti alla classe positiva. Tanto più la recall è elevata, tanto più basso è il numero dei falsi negativi.

$$\text{Recall} = \frac{\text{true positive}}{\text{positive}}$$

- **F1**: Generalmente all' aumentare della precision per un certo modello, la recall diminuisce, e viceversa. Nel caso si voglia raggiungere il miglior equilibrio possibile tra le due metriche, può essere utile considerare l' f1, che esprime la media armonica tra di esse.

$$\text{F1} = \frac{\text{true positive}}{\text{true positive} + \frac{1}{2}(\text{false positive} + \text{false negative})}$$

Capitolo 3

Dataset

3.1 Presentazione

Il dataset di partenza contiene dati su 2698 esami, relativi a 388 studenti iscritti al primo anno del Corso in Informatica dell'Università di Parma tra il 2016 e il 2018. In tabella 3.1 è mostrata la distribuzione degli esami e degli studenti negli anni.

Anno	Numero esami	Numero studenti
2016	857	122
2017	924	132
2018	917	134

Tabella 3.1: distribuzione studenti ed esami negli anni.

I corsi proposti sono gli stessi per tutti gli anni studiati, quindi tutti gli studenti hanno sempre dovuto affrontare gli stessi esami, che sono mostrati in tabella 3.2 (tra parentesi sono scritti gli acronimi con cui ci si riferirà agli esami in seguito).

I Semestre	II Semestre
Architettura degli elaboratori (arch)	Fisica (fis)
Analisi matematica (an)	Algebra e geometria (alge)
Inglese (ing)	Algoritmi e strutture dati (algo)
	Fondamenti di programmazione (fond)

Tabella 3.2: esami I anno.

Per ogni esame, il dataset contiene dati sullo studente a cui è associato, e sulle sue performance nell'affrontare la prova, come mostrato in tabella 3.3.

Dato	Tipo	Esempio
identificativo studente	stringa	238006
anno iscrizione studente all'università	intero	2015
nome insegnamento	stringa	analisi
superamento esame	booleano	true
voto ottenuto	intero	27
ottenimento lode	booleano	false
numero di tentativi fatti	intero	2
data fine lezioni	stringa	10/06/2016
data superamento esame	stringa	17/07/2016
data primo tentativo	stringa	20/06/2016
data ultimo tentativo	stringa	17/07/2016

Tabella 3.3: dati disponibili per ogni esame.

3.2 Analisi preliminari

3.2.1 Correlazione CFU ottenuti e attività II anno

Il grafico in figura 3.1 mostra quanti studenti hanno conseguito ogni possibile numero di CFU al termine del I anno, evidenziando quanti di questi provano almeno un esame durante l'anno successivo (studenti "attivi"). Gli studenti attivi al II anno dovrebbero approssimare con buona precisione quelli che vi sono iscritti.

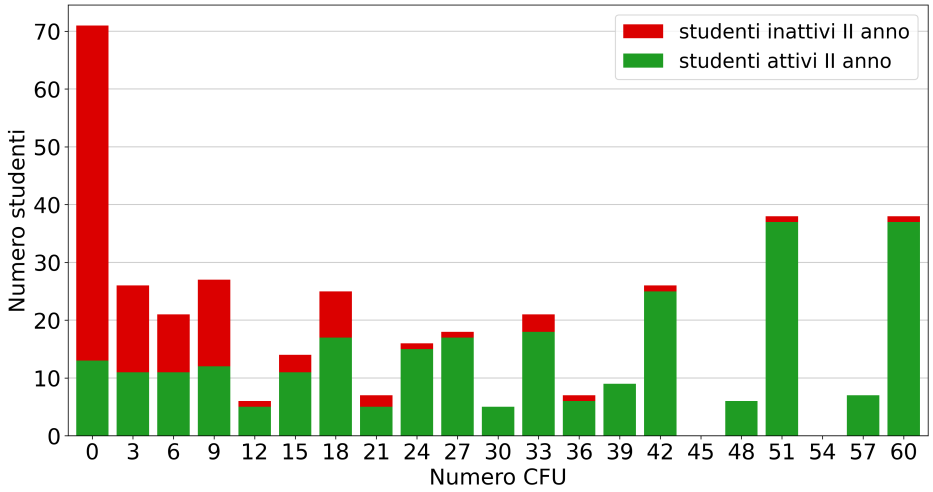

Figura 3.1: numero studenti per fascia di CFU.

Alcune considerazioni aggiuntive:

- il 29.6% degli studenti conseguono almeno 40 CFU a fine anno;
- il 18.2% non supera alcun esame;
- mediamente, uno studente ottiene 24.8 CFU, valore che si sposta a 30.4 escludendo gli studenti che non superano alcun esame;
- gli studenti inattivi al II anno sono il 31.2%;
- l'81% degli inattivi al II anno è composto da studenti che hanno ottenuto meno di 10 CFU al primo.

3.2.2 Performance studenti nei corsi

Le figure 3.2, 3.3 e 3.4 mostrano alcuni dati legati alle performance degli studenti nei diversi corsi del I anno:

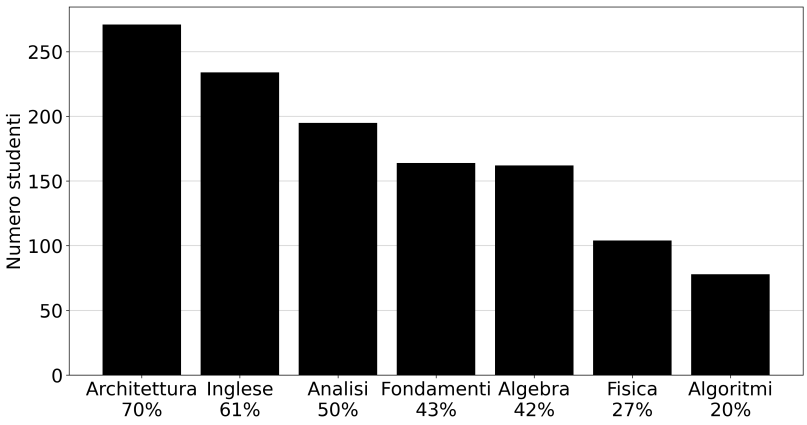

Figura 3.2: numero studenti che superano ogni esame.

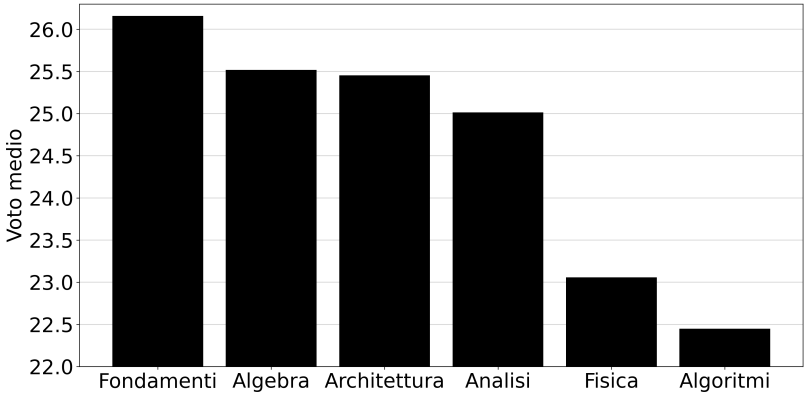

Figura 3.3: voto medio per ogni esame.

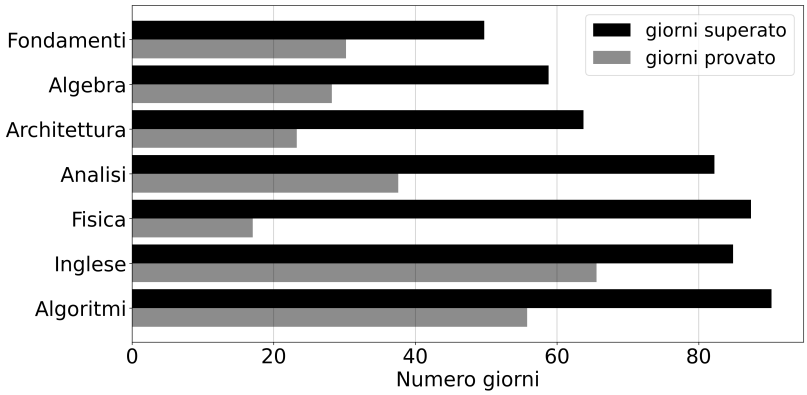

Figura 3.4: giorni medi dopo cui ogni esame viene provato/superato, a partire dal termine delle lezioni del corso.

I corsi di algoritmi e fisica sembrano essere quelli in cui gli studenti incontrano maggiore difficoltà, infatti presentano:

- i numeri di studenti che li superano più bassi;
- i voti medi più bassi;
- i numeri di giorni dopo cui gli studenti mediamente li superano più alti.

3.2.3 Pattern nel superamento esami

Dati gli studenti che superano un certo esame, quali altri esami tendono a superare più facilmente? In quali altri esami incontrano invece maggiore difficoltà? La matrice in figura 3.5 può aiutare a individuare pattern di questo tipo, mostrando quanti degli studenti che superano l'esame indicato sulla riga, in percentuale, riescono a superare anche ciascuno degli altri esami, indicati sulle colonne. La matrice considera superati solo gli esami che vengono affrontati con successo dagli studenti entro la fine del loro primo anno accademico.

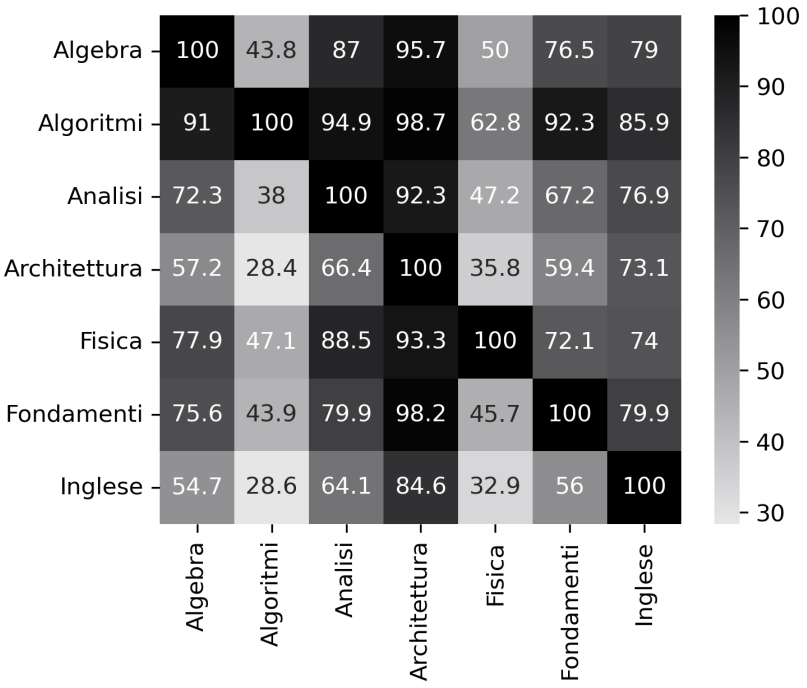

Figura 3.5

3.3 Preprocessing

3.3.1 Gestione valori mancanti e inconsistenti

Il dataset di partenza presentava alcuni valori mancanti o inconsistenti. La tabella 3.4 mostra, per ogni tipologia di questi valori, con quale frequenza si verificava, e come sono stati gestiti.

Valore mancante / inconsistente	Numero casi in cui si verifica	Soluzione utilizzata
data fine lezioni mancante	139 (5.2%)	utilizzo data associata ad altri studenti dello stesso anno
data fine lezioni successiva a data superamento esame	30 (1.1%)	data fine lezioni stimata
data fine lezioni successiva a data primo tentativo	66 (2.4%)	data fine lezioni stimata
esame superato, ma numero tentativi e data primo tentativo mancante	88 (3.2%)	valori mancanti stimati
esami relativi ad alcuni esami di certi studenti mancanti	17 esami, legati a 5 studenti	esami mancanti aggiunti, segnandoli come non superati.

Tabella 3.4: gestione valori mancanti o inconsistenti.

3.3.2 Trasformazione features

Il dataset di partenza conteneva alcune features in forma di date, quindi inadatte all'addestramento di sistemi predittivi. I dati di questo tipo sono stati utilizzati per ricavare nuove features in formato numerico:

- **numero giorni primo tentativo**, che indica il numero di giorni dopo cui lo studente ha provato per la prima volta l'esame (vale -1 nel caso l'esame non sia mai stato provato);

- **numero giorni esame superato**, che indica il numero di giorni dopo cui lo studente ha superato l'esame (vale -1 nel caso l'esame non sia stato superato).

In figura 3.6 è mostrato il processo di costruzione delle nuove features:

Figura 3.6: costruzione nuove features a partire da quelle in forma di data.

Alcune features sono state codificate all'interno di altre, in modo da rendere l'input per i modelli di machine learning più compatto:

- la feature **"superamento esame"** è stata codificata all'interno della feature "voto", segnando voto = -1 in caso di mancato superamento;
- la feature **"lode"** è stata codificata all'interno della feature "voto", segnando voto = 31 in caso di conseguimento della lode.

3.3.3 Struttura finale

Terminate queste operazioni, sono state eliminate tutte le features in formato di data ("data ultimo tentativo", "data primo tentativo", "data superamento esame", "data fine lezioni"), quelle che sono state codificate all'interno di altre ("superamento esame" e "ottenimento lode"), e quelle prive di significato per i modelli ("identificativo studente", "anno iscrizione all'università", "nome insegnamento").

A questo punto, per ogni esame, il dataset contiene queste features:

- voto ottenuto;
- numero di tentativi fatti;
- giorni esame superato;
- giorni esame provato.

Nota che, per come sono stati codificati i dati, sono presenti anche altre due features implicite:

- superamento / non superamento dell'esame;
- ottenimento / non ottenimento della lode.

Il dataset finale è composto da 388 elementi come quello mostrato in tabella 3.5, ciascuno contenente 28 features associate agli esami di un certo studente. Nel capitolo successivo, questi elementi verranno utilizzati come input per l'addestramento dei modelli predittivi.

Insegnamento	Voto	Ntentativi	Giorni sup	Giorni prov
algebra	26	2	31	3
algoritmi	0	2	-1	22
analisi	18	1	14	14
architettura	26	1	22	22
fisica	22	3	63	17
fondamenti	21	1	33	33
inglese	0	0	-1	-1

Tabella 3.5: elemento del dataset finale.

Capitolo 4

Implementazione sistema

Come anticipato, la previsione dell'iC16 relativo a un certo insieme di studenti richiede di stimare:

- quanti degli studenti avranno conseguito almeno 40 CFU a fine anno;
- quanti si iscriveranno al II anno.

Di conseguenza, per implementare il sistema finale è stato necessario addestrare due modelli di machine learning distinti. I modelli fanno le loro previsioni su singoli studenti, in base alle features relative ai loro esami, come mostrato in figura 4.1:

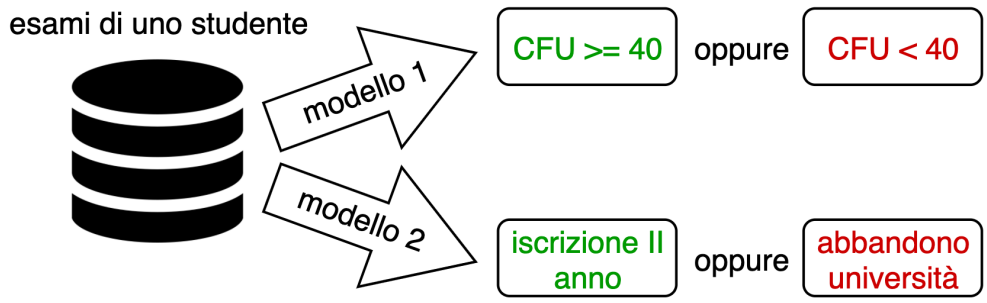

Figura 4.1: previsioni necessarie per stimare l'iC16.

Per semplicità, d'ora in poi con "**modello 1**" si farà riferimento al modello di previsione dei 40CFU, mentre con "**modello 2**" si indicherà quello per la previsione delle iscrizioni al II anno.

4.1 Costruzione train e test set

4.1.1 Features e labels utilizzate

Per addestrare i modelli, è stato utilizzato il dataset risultante dalla fase di preprocessing, esposto nella sottosezione 3.3.3, che è composto da 388 elementi, ciascuno dei quali contiene 28 features relative agli esami di un certo studente. Ogni elemento del dataset corrisponde quindi a uno studente diverso.

Per l'addestramento del primo modello, a ogni elemento del dataset è stata associata una label booleana, a indicare il raggiungimento dei 40 CFU al termine del I anno.

Per l'addestramento del secondo modello sarebbe stato ottimale associare a ogni studente una label simile alla precedente, che indicasse la sua iscrizione al II anno, purtroppo il dataset di partenza non conteneva dati sulle iscrizioni degli studenti agli anni successivi, e quindi non ha permesso di costruire labels di questo tipo.

La soluzione utilizzata è stata quella di classificare gli studenti in "attivi" e "inattivi" al II anno, seguendo queste regole:

- gli studenti che provano almeno un esame durante il II anno sono stati considerati "attivi";
- quelli che invece non provano neanche un esame per l'intero anno sono stati considerati "inattivi".

Dato che difficilmente uno studente si iscrive al II anno senza provare alcun esame, la label che indica l'attività di uno studente al II anno dovrebbe approssimare con buona precisione quella che ne indica l'iscrizione.

4.1.2 Bilanciamento dataset

Come mostrato nella sottosezione 3.2.1, i dataset per entrambi i modelli erano inizialmente sbilanciati, infatti:

- nel caso del modello 1, il dataset era sbilanciato verso la label negativa, in quanto il 70.4% degli studenti non raggiungeva i 40CFU a fine anno;
- nel caso del modello 2, il dataset era sbilanciato verso la label positiva, infatti il 68.8% degli studenti risultava attivo al II anno.

Le figure 4.2 e 4.3 mostrano il processo di bilanciamento dei dataset, avvenuto sfruttando SMOTE (vedi sezione 2.1.1).

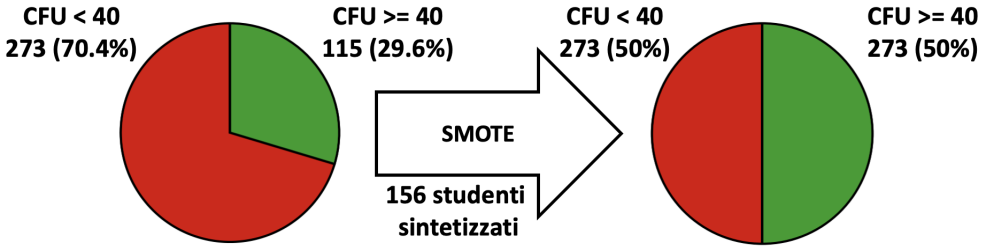

Figura 4.2: bilanciamento dataset 1.

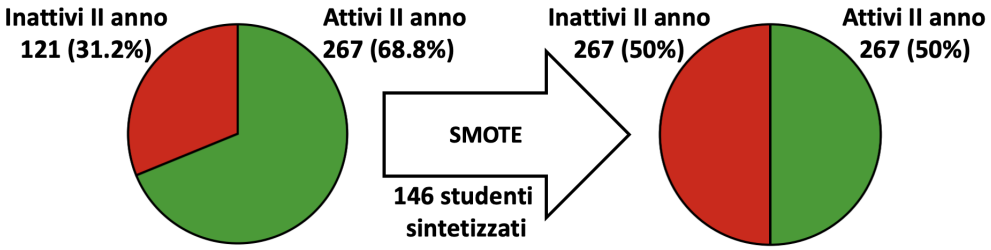

Figura 4.3: bilanciamento dataset 2.

4.1.3 Train e test set

Entrambi i dataset sono stati divisi in train e test set con un rapporto 75 a 25. In tabella 4.1 è mostrato nel dettaglio il numero di elementi di ciascun insieme.

Dataset	Elementi totali	Train set	Test set
1	546	409	137
2	534	400	134

Tabella 4.1

4.2 Ricerca modelli ottimali

I grafici in figura 4.4 e 4.5 mostrano il confronto tra le performance di alcuni modelli di machine learning, che sono stati scelti perchè la

letteratura scientifica ne ha dimostrato una particolare efficacia per le due tipologie di problemi affrontati (vedi capitolo 1), oppure perchè TPOT (vedi sezione 2.1.2) li ha individuati come pipeline ottimali, caso nel quale sono contrassegnati con " * ".

Le performance dei modelli sono state misurate con cadenza mensile, a partire dall'inizio della sessione invernale, momento nel quale gli studenti cominciano a sostenere i primi esami, e quindi a generare i primi input per i sistemi.

Per i motivi esposti nella sezione 2.2, i modelli sono stati ricercati principalmente tra quelli di classificazione, ma anche tra quelli di regressione.

Per questioni di spazio, in alcuni casi i sistemi sono stati indicati con gli acronimi mostrati in tabella 4.2

Acronimo	Modello Corrispondente
XGBR	Extreme Gradient Boosting Regressor
RFC	Random Forest Classifier
RFR	Random Forest Regressor
BNB	Bernoulli Naive Bayes
GBC	Gradient Boosting Classifier
ETR	Extra Trees Regressor
LSVC	Linear Support Vector Classifier
ENR	Elastic Net Regressor
DT	Decision Tree
NN	Neural Network

Tabella 4.2: distribuzione studenti ed esami negli anni.

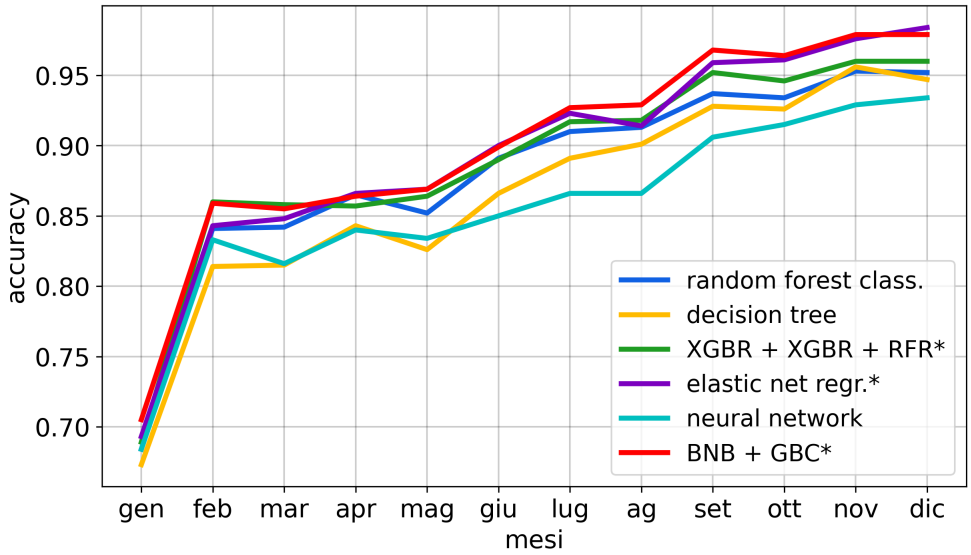

Figura 4.4: previsione 40CFU, accuracy modelli per ogni mese.

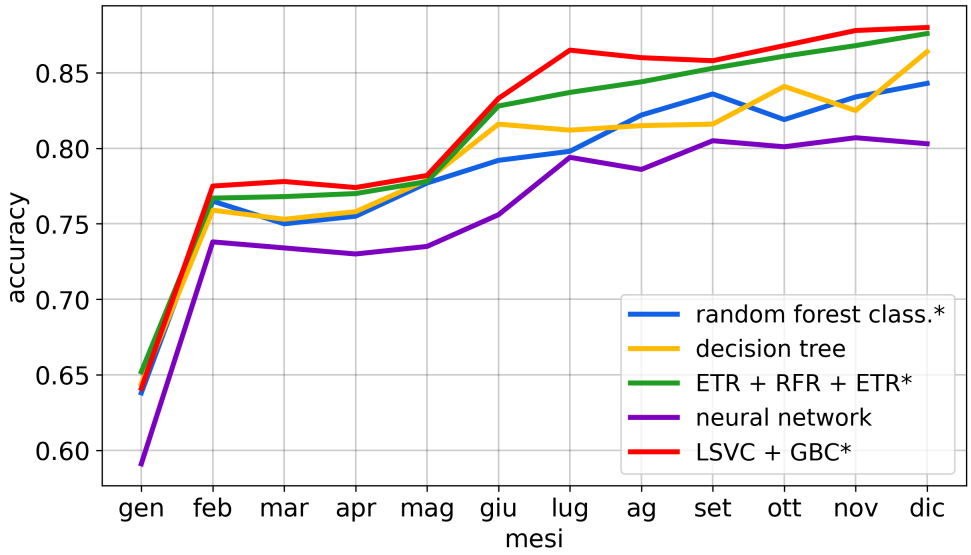

Figura 4.5: previsione attività II anno, accuracy modelli per ogni mese.

Le tabelle 4.3 e 4.4 mostrano i valori dell'accuracy associata a ciascun modello nel dettaglio (in grassetto è evidenziato il risultato migliore):

mese	LSVC+GBC	ENR	RFR	RFC	DT	NN
gen	**0.705**	0.693	0.689	0.684	0.673	0.684
feb	0.859	0.843	**0.860**	0.841	0.814	0.833
mar	0.855	0.848	**0.858**	0.842	0.815	0.816
apr	0.864	**0.866**	0.857	0.865	0.843	0.840
mag	**0.869**	**0.869**	0.864	0.852	0.826	0.834
giu	0.899	**0.900**	0.890	0.891	0.866	0.850
lug	**0.927**	0.923	0.917	0.910	0.891	0.866
ago	**0.929**	0.914	0.918	0.913	0.901	0.866
set	**0.968**	0.959	0.952	0.937	0.928	0.906
ott	**0.964**	0.961	0.946	0.934	0.926	0.915
nov	**0.979**	0.976	0.960	0.953	0.956	0.929
dic	0.979	**0.984**	0.960	0.952	0.947	0.934

Tabella 4.3: previsione 40CFU, accuracy modelli per ogni mese.

mese	LSVC+GBC	ETR+RFR+ETR	RFC	DT	NN
gen	0.641	**0.652**	0.638	0.644	0.591
feb	**0.775**	0.767	0.765	0.759	0.738
mar	**0.778**	0.768	0.75	0.753	0.734
apr	**0.774**	0.770	0.755	0.758	0.73
mag	**0.782**	0.778	0.777	0.779	0.735
giu	**0.833**	0.828	0.792	0.816	0.756
lug	**0.865**	0.837	0.798	0.812	0.794
ago	**0.860**	0.844	0.822	0.815	0.786
set	**0.858**	0.853	0.836	0.816	0.805
ott	**0.868**	0.861	0.819	0.841	0.801
nov	**0.878**	0.868	0.834	0.825	0.807
dic	**0.880**	0.876	0.843	0.0.864	0.803

Tabella 4.4: previsione attività II anno, accuracy modelli per ogni mese.

Questi dati mostrano che:

- il modello con le prestazioni migliori sul primo problema è la pipeline "Bernoulli Naive Bayes + Gradient Boosting Classifier";
- quello con le prestazioni migliori sul secondo problema è la pipeline "Linear Support Vector Classifier + Gradient Boosting Classifier";

È anche possibile osservare che i modelli testati hanno performance generalmente migliori sul I problema rispetto che sul II.

4.3 Dettagli sui modelli ottimali

4.3.1 Struttura pipeline

Entrambe le pipeline processano le features ricevute in input tramite due modelli utilizzati a catena, per restituire un output booleano, che indica la previsione finale. La loro struttura è mostrata in figura 4.6 e 4.7

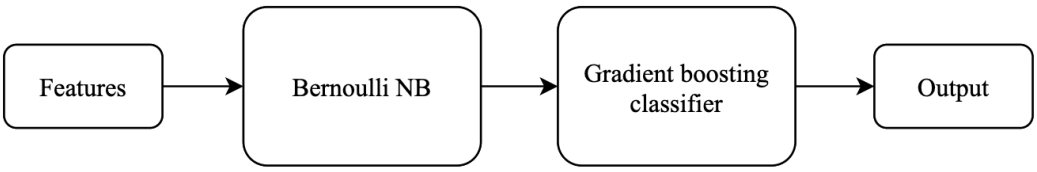

Figura 4.6: pipeline previsione 40 CFU.

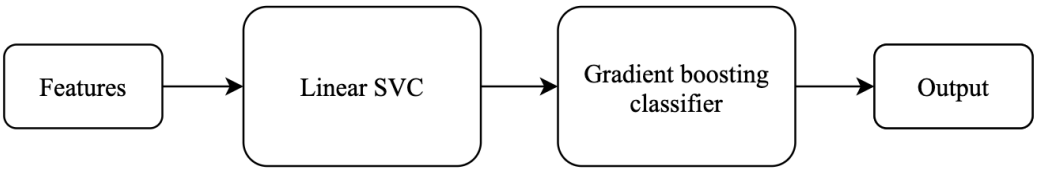

Figura 4.7: pipeline previsione iscrizioni II anno.

Il **teorema di Bayes** definisce come calcolare la **probabilità condizionata** di un evento A rispetto a un evento B, ovvero la probabilità che si verifichi A sapendo che si è verificato B. Il modello **Bernoulli Naive Bayes**, utilizzato dalla prima pipeline, sfrutta questo teorema per effettuare le proprie previsioni, andando a calcolare la probabilità che l'elemento di input appartenga a ciascuna classe, condizionata rispetto ai valori delle features. In tabella 4.5 sono riportati i parametri utilizzati per questo modello:

Parametro	Valore
alpha	10.0
fit_prior	true

Tabella 4.5: parametri BernoulliNB

Il **Linear Support Vector Classifier**, utilizzato dalla seconda pipeline, "trasforma" i dati di training in vettori grazie a una funzione di mapping, e li usa per individuare l' **iperpiano** ottimale per la separazione delle classi. Questo iperpiano permetterà poi di separare i vettori ricevuti in input in futuro, permettendo di classificarli. In tabella 4.6 sono mostrati i parametri utilizzati per questo modello.

Parametro	Valore
C	15.0
dual	false
loss	squared_hinge
penality	l1
tol	0.01

Tabella 4.6: parametri LSVC.

Il **gradient boosting classifier**, utilizzato da entrambe le pipeline, è un modello che funziona combinando gli output di più decision tree, in modo per certi aspetti simile al classificatore **random forest**; i due modelli differiscono però per il modo con il quale gli alberi vengono costruiti e combinati insieme.

Il random forest classifer costruisce una serie di alberi di decisione tutti indipendenti tra loro, ed effettua le proprie previsioni combinando i risultati che restituiscono, in parallelo.

Il gradient boosting classifier, invece, effettua le proprie previsioni grazie a un unico decision tree, ricavato utilizzando un metodo chiamato "**boosting**": a partire da un primo decision tree poco preciso, vengono costruiti in maniera sequenziale nuovi alberi, in modo che ciascuno faccia previsioni sempre più accurate del precedente, fino ad arrivare a quello ottimale.

In tabella 4.7 sono mostrati i valori per i parametri per i due gradient boosting classifier utilizzati:

Parametro	Valore pipeline I	Valore pipeline II
learing_rate	0.1	0.5
max_depth	2	8
max_features	0.1	0.1
min_samples_leaf	7	3
min_samples_split	12	11
n_estimators	100	100
subsample	0.5	0.65

Tabella 4.7: valore parametri per i gradient boosting classifier delle due pipeline.

4.3.2 Performance

Seppure un modello di machine learning possa dare buoni risultati in termini di accuracy, per avere una visione più completa sulle sue performance è necessario andare a valutare anche altre metriche. I grafici in figura 4.8 e 4.9 mostrano i valori di accuracy, precision, recall ed f1 (che sono state approfondite nella sezione 2.3) per le due pipeline individuate. Anche in questo caso, le misurazioni sono state fatte con cadenza mensile, a partire dal mese di gennaio, nel quale cominciano a essere disponibili i primi input per i modelli.

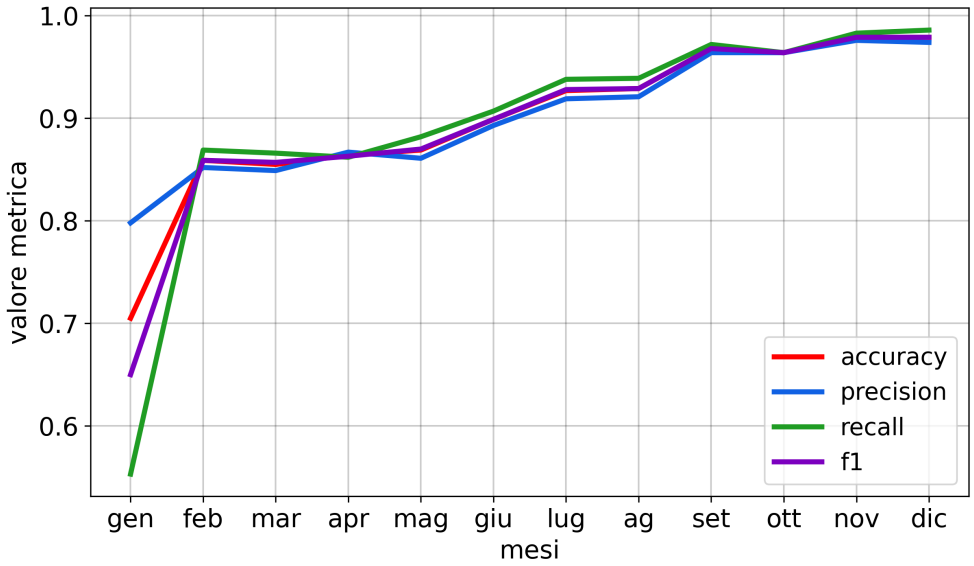

Figura 4.8: previsione 40 CFU, metriche.

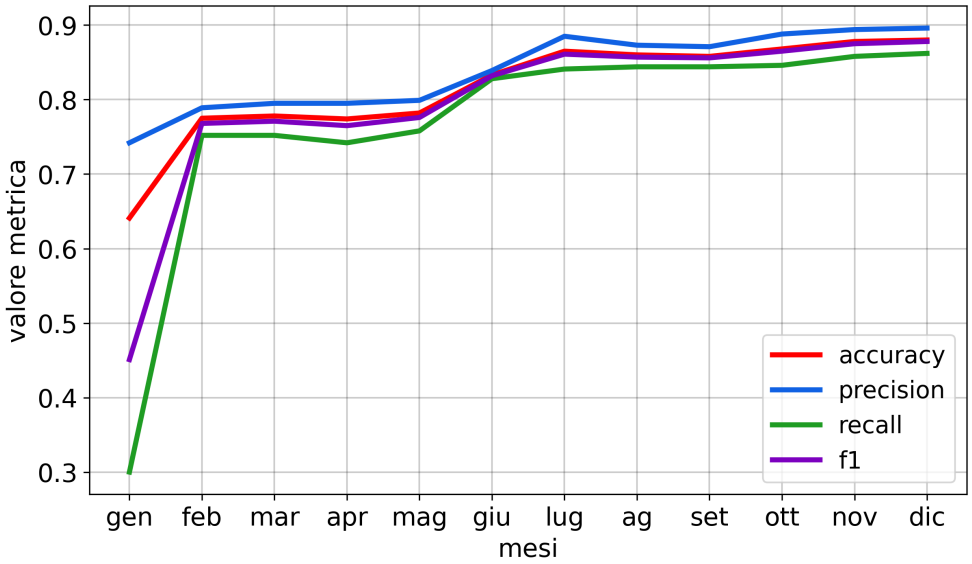

Figura 4.9: previsione attività II anno, metriche.

Le tabelle 4.8 e 4.9 mostrano più nel dettaglio i valori delle metriche:

mese	accuracy	precision	recall	f1
gen	0.705	0.798	0.553	0.650
feb	0.859	0.852	0.869	0.859
mar	0.855	0.849	0.866	0.857
apr	0.864	0.867	0.862	0.863
mag	0.869	0.861	0.882	0.870
giu	0.899	0.893	0.907	0.899
lug	0.927	0.919	0.938	0.928
ago	0.929	0.921	0.939	0.929
set	0.968	0.964	0.972	0.968
ott	0.964	0.964	0.964	0.964
nov	0.979	0.976	0.983	0.979
dic	0.979	0.974	0.986	0.979

Tabella 4.8: previsione 40 CFU, metriche.

mese	accuracy	precision	recall	f1
gen	0.641	0.742	0.300	0.451
feb	0.775	0.789	0.752	0.768
mar	0.778	0.795	0.752	0.771
apr	0.774	0.795	0.742	0.765
mag	0.782	0.799	0.758	0.776
giu	0.833	0.839	0.828	0.832
lug	0.865	0.885	0.841	0.861
ago	0.860	0.873	0.844	0.857
set	0.858	0.871	0.844	0.856
ott	0.868	0.888	0.846	0.865
nov	0.878	0.894	0.858	0.875
dic	0.880	0.896	0.862	0.878

Tabella 4.9: attività II anno, metriche

È possibile osservare come i due modelli abbiano precision e recall abbastanza vicine tra loro, questo indica che non tendono a favorire eccessivamente né la previsione positiva e né quella negativa.

4.3.3 Analisi features più significative

Tutti i dati mostrati nei grafici di questa sezione sono stati ricavati grazie a LIME (vedi sezione 2.1.3)

I grafici in figura 4.10 e 4.11 mostrano l'impatto delle features sulle previsioni dei modelli, al termine delle 3 sessioni di esame:

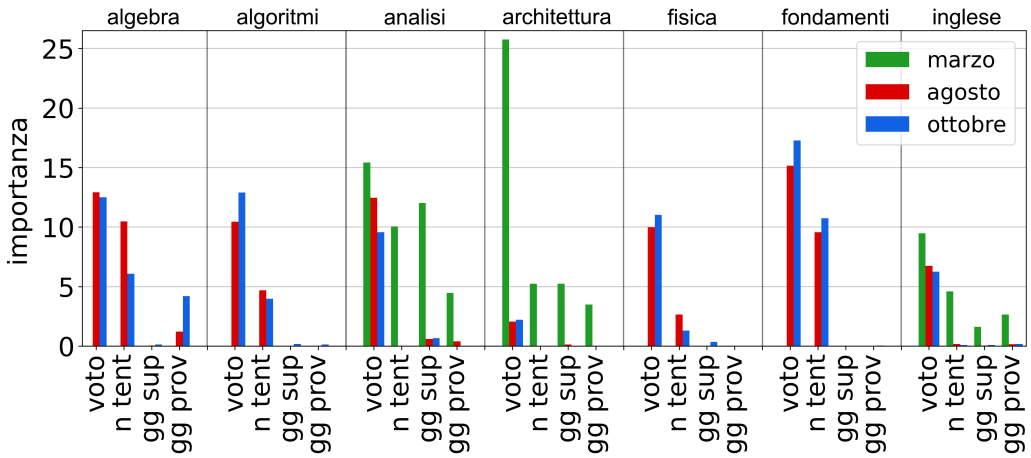

Figura 4.10: previsione 40CFU, importanza features.

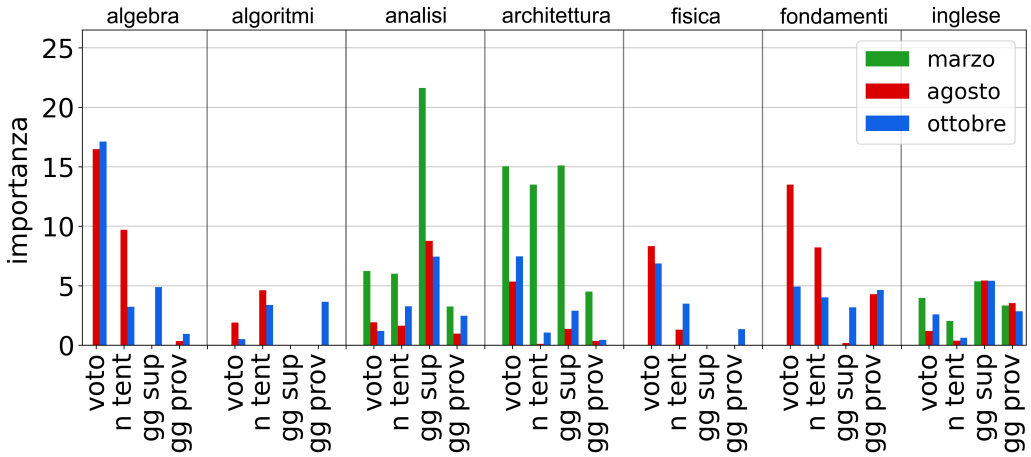

Figura 4.11: previsione iscrizioni II anno, importanza features.

A partire da questi dati è possibile valutare l'importanza media delle features generiche, indipendentemente dall'esame a cui sono associate. Questo tipo di analisi è mostrata nei grafici in figura 4.12 e 4.13:

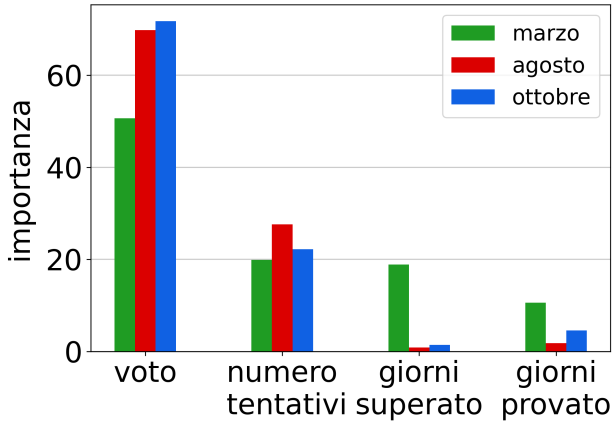

Figura 4.12: previsione 40CFU, importanza media features generiche.

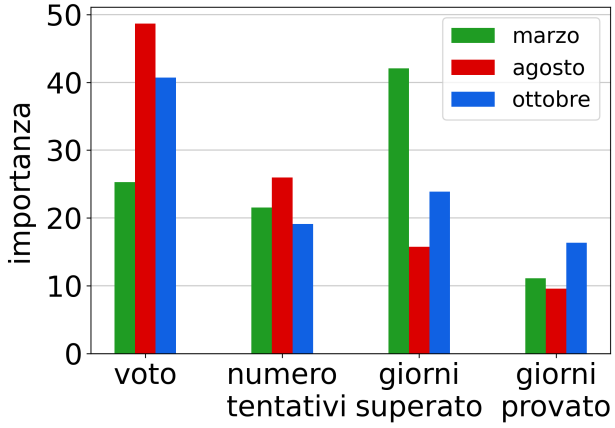

Figura 4.13: previsione iscrizioni II anno, importanza media features generiche.

È anche possibile valutare quanto i dati relativi a ogni esame contribuiscano mediamente alle decisioni del modello, indipendentemente dalle features specifiche. Questo tipo di analisi è mostrata nei grafici in figura 4.14 e 4.15:

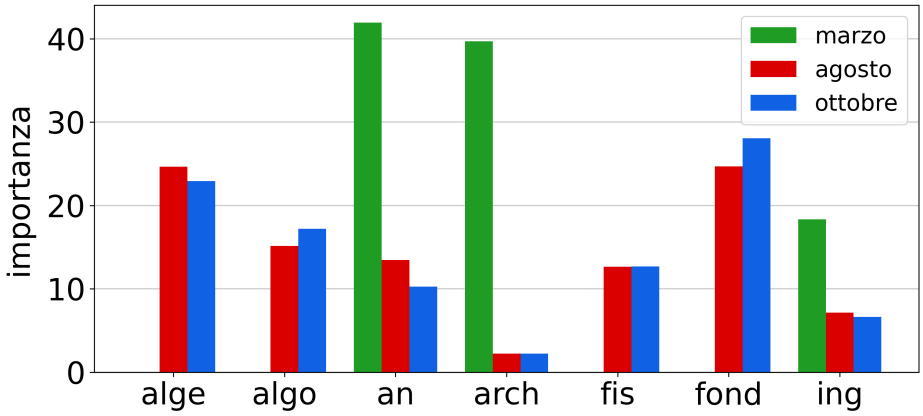

Figura 4.14: previsione 40 CFU, importanza media features associate a ogni esame.

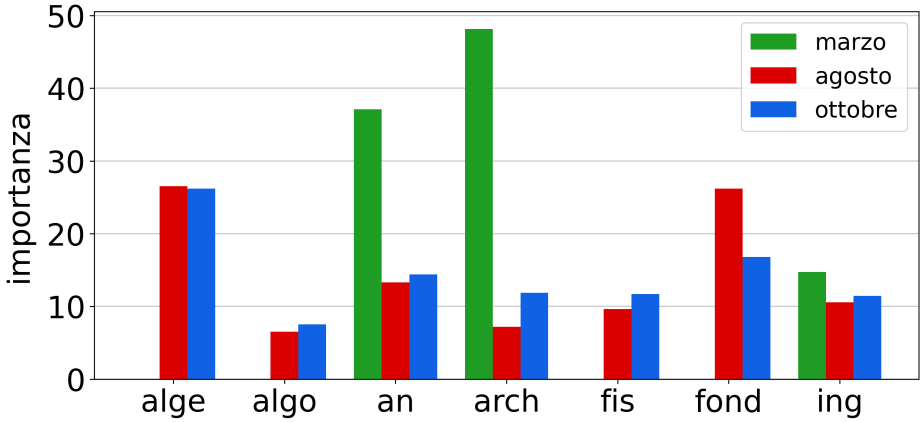

Figura 4.15: iscrizioni II anno, importanza media features associate a ogni esame.

4.4 Previsione iC16

Utilizzando i due modelli individuati in modo combinato, è possibile prevedere l'iC16 associato a un certo insieme di studenti:

- per ogni elemento, si utilizza il primo modello per prevedere se a fine anno avrà conseguito almeno 40 CFU;
- si utilizza il II modello per prevedere se sarà attivo al II anno;
- la percentuale degli studenti per cui i due modelli hanno dato entrambi previsione positiva coincide con l'iC16.

Per testare il sistema finale, i due modelli sono stati addestrati su un train set comune di 291 studenti, che è stato bilanciato in modo opportuno per ognuna delle due previsioni tramite SMOTE. Il grafico in figura 4.16 mostra l'errore medio associato alle previsioni del sistema sull'iC16 dei 97 studenti che componevano il test set. L'errore è espresso come il numero di punti iC16 che separano l'indice previsto da quello effettivo (l'iC16 esprime una percentuale, quindi può avere un valore tra 0 e 100).

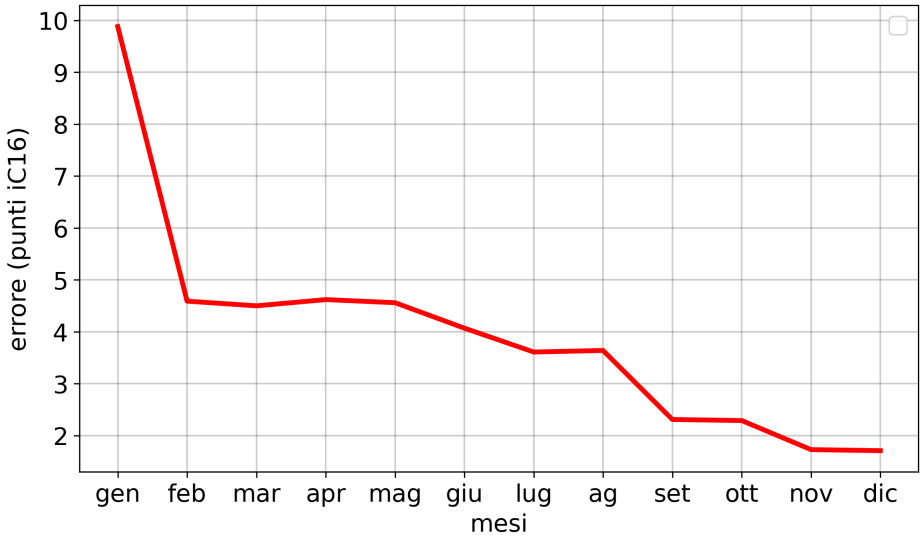

Figura 4.16: errore previsione del sistema finale.

È importante sottolineare che, per i motivi spiegati nella sezione 4.1.1, l'iC16 "corretto" utilizzato per valutare l'errore è stato calcolato secondo l'attività degli studenti al II anno, e non in base alla loro effettiva iscrizione.

Conclusioni

È stato possibile costruire un sistema per la previsione dell'indice iC16 relativo al corso in informatica dell'Università di Parma, utilizzando in modo combinato due modelli di machine learning. Dato un certo insieme di studenti, il primo modello prevede quanti di questi raggiungeranno i 40 CFU a fine anno, mentre il secondo stima quanti saranno attivi al II anno.

Le performance migliori nell'eseguire le previsioni sono state dimostrate da due pipeline individuate tramite **TPOT**, che sfruttano entrambe il **gradient boosting classifier**, combinato nel caso della prima pipeline con il **Bernoulli naive bayes classifier**, e nel caso della seconda con il **linear support vector classifier**. I modelli testati si sono dimostrati più performanti sulla prima previsione rispetto che sulla seconda.

I test sul sistema finale hanno mostrato un errore medio attorno ai 4.5 punti percentuali dopo il mese di febbraio, che scende attorno ai 3.6 al termine di luglio e circa a 2.3 a fine settembre, per arrivare a un minimo di circa 1.75 a fine novembre. Bisogna però sottolineare che risultati ottenuti potrebbero non essere totalmente accurati a causa della mancanza di dati certi sulle iscrizioni degli studenti al II anno nel dataset iniziale.

Sono inoltre state mostrate la fase di preprocessing e quella di bilanciamento dei train set tramite **SMOTE**, un' analisi preliminare carriere degli studenti e una sull'impatto delle features nelle decisioni dei modelli, sfruttando **LIME**.

I principali limiti allo sviluppo del sistema finale sono stati la quantità di dati relativamente scarsa (train e test set contenevano in totale 388 elementi), e l'assenza di dati certi sulle iscrizioni degli studenti al II anno, che ha costretto a stimare questo dato.

Di conseguenza, il sistema potrebbe essere migliorato andando a lavorare

su queste due limitazioni, raccogliendo dati relativi a un maggior numero di studenti, in modo da incrementare le dimensioni di train e test, e raccogliendo dati certi sulle iscrizioni degli studenti al II anno, in modo da utilizzare labels più affidabili per l'addestramento del II modello, e così da poter anche eseguire test più precisi sulle performance del sistema finale.

Un altro modo per migliorare le prestazioni del sistema potrebbe essere quello di affiancare alle features già in uso nuovi dati relativi alle condizioni socio-demografiche degli studenti, in modo da poter anche effettuare previsioni anticipo rispetto al sistema attuale.

Potrebbe essere interessante anche sostituire il II modello con quello implementato da Cineca (vedi 1.2).

Bibliografia

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[Brownlee, 2020] Brownlee, J. (2020). SMOTE for Imbalanced Classification with Python. `https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/`. [accessed 28/02/2022].

[Chollet et al., 2015] Chollet, F. et al. (2015). Keras.

[Contini et al., 2017] Contini, D., Salza, G., and Scagni, A. (2017). Abbandono universitario e tempi alla laurea: una criticità in evoluzione positiva?

[Del Bonifro et al., 2020] Del Bonifro, F., Gabbrielli, M., Lisanti, G., and Zingaro, S. P. (2020). Student dropout prediction. In Bittencourt, I. I., Cukurova, M., Muldner, K., Luckin, R., and Millán, E., editors, *Artificial Intelligence in Education*, pages 129–140, Cham. Springer International Publishing.

[Gkontzis et al., 2018] Gkontzis, A. F., Kotsiantis, S., Tsoni, R., and Verykios, V. S. (2018). An effective la approach to predict student achievement. In *Proceedings of the 22nd Pan-Hellenic Conference on Informatics*, pages 76–81.

[Harris et al., 2020] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg,

S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

[Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

[Ilgan, 2013] Ilgan, A. (2013). Predicting college student achievement in science courses. *Journal of Baltic Science Education*, 12(3):322.

[Jaiswal et al., 2021] Jaiswal, K., Pathak, P., Pawar, V., and Kharat, K. (2021). Prediction of degree student achievement analysis. In *IOP Conference Series: Materials Science and Engineering*, volume 1070, page 012057. IOP Publishing.

[Kabakchieva, 2012] Kabakchieva, D. (2012). Student performance prediction by using data mining classification algorithms. *International journal of computer science and management research*, 1(4):686–690.

[Lemaître et al., 2017] Lemaître, G., Nogueira, F., and Aridas, C. K. (2017). Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5.

[Lykourentzou et al., 2009] Lykourentzou, I., Giannoukos, I., Mpardis, G., Nikolopoulos, V., and Loumos, V. (2009). Early and dynamic student achievement prediction in e-learning courses using neural networks. *Journal of the American Society for Information Science and Technology*, 60(2):372–380.

[Macaluso, 2018] Macaluso, A. (2018). Osservatorio Abbandoni: il machine learning al servizio delle università. `https://it.readkong.com/page/osservatorio-abbandoni-il-machine-learning-al-servizio-6011049`. [accessed 28/02/2022].

[Nuankaew et al., 2020] Nuankaew, P., Nuankaew, W., Teeraputon, D., Phanniphong, K., and Bussaman, S. (2020). Prediction model of student achievement in business computer disciplines. *International Journal of Emerging Technologies in Learning (iJET)*, 15(20):160–181.

[Olson et al., 2016] Olson, R. S., Bartley, N., Urbanowicz, R. J., and Moore, J. H. (2016). Evaluation of a tree-based pipeline optimization tool for

automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA. ACM.

[pandas development team, 2020] pandas development team, T. (2020). pandas-dev/pandas: Pandas.

[Pandey and Sharma, 2013] Pandey, M. and Sharma, V. K. (2013). A decision tree algorithm pertaining to the student performance analysis and prediction. *International Journal of Computer Applications*, 61(13).

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Tan and Shao, 2015] Tan, M. and Shao, P. (2015). Prediction of student dropout in e-learning program through the use of machine learning method. *International journal of emerging technologies in learning*, 10(1).

[Utari et al., 2020] Utari, M., Warsito, B., and Kusumaningrum, R. (2020). Implementation of data mining for drop-out prediction using random forest method. In *2020 8th International Conference on Information and Communication Technology (ICoICT)*, pages 1–5. IEEE.

[Van Rossum and Drake, 2009] Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.