Report progetto Intelligenza artificiale Interpretable Dropout (Dropout accademico)

Corso di Laurea Magistrale in Informatica
Anno Accademico 2022-2023

Gruppo: **OkComputer**
Alessandro Pistola (1058248)

Università di Bologna
Dipartimento di Informatica – Scienza e Ingegneria
Mura Anteo Zamboni 7
Bologna (BO), Italia

Indice contenuti

1 Introduzione

1.1 Descrizione del problema

Il problema del dropout accademico riguarda gli studenti che non completano gli studi. Questo rappresenta un problema poichè tale abbandono può causare costi elevati e rendere la società meno produttiva. Al fine di aiutare gli studenti a non abbandonare gli studi, si sta cercando di sviluppare un sistema di intelligenza artificiale che possa prevedere il rischio di dropout per poi andare a proporre soluzioni personalizzate. Tutto questo, con l'obiettivo di aiutare gli studenti a completare gli studi migliorandone il tasso di successo.
La natura "black box" dei sistemi di machine learning non permette però di comprendere il processo di decision making del sistema.
Ciò rappresenta un problema poichè può suscitare incertezza e diffidenza nei confronti della predizione soprattutto quando il campo di applicazione è un settore critico o la decisione riguarda un ambito eticamente sensibile.
Inoltre, la mancanza di interpretabilità rende difficile il processo di debugging e l'individuazione di eventuali bias nel modello.
Proprio per queste ragioni, il focus principale del seguente progetto è lo sviluppo di un sistema di machine learning e l'utilizzo di tecniche di interpretabilità al fine di garantire la trasparenza nel processo di decisione del sistema proposto.

1.2 Descrizione della soluzione proposta

La soluzione proposta, è una soluzione nata da un processo di sviluppo comprendente operazioni di preprocessing dei dati, ricerca del modello ottimale, hyperparameter tuning del modello stesso e operazioni di interpretazione locale e globale.
Per lo sviluppo di tale progetto, si è fatto uso di un dataset pseudonimizzato fornito dall'Università di Bologna comprendente informazioni relative a studenti e corrispondenti esami sostenuti.
La presenza di informazioni sensibili come genere o condizione economica ha reso le operazioni di interpretazione dei risultati ottenuti molto importanti e sensate, al fine di constatare se il modello discriminasse o meno sulla base di tali attributi.
La soluzione proposta, in conclusione, è un notebook jupyter comprendente tutte le fasi di sviluppo. Il modello finale scelto per l'interpretazione è un Extreme Gradient Boosting (xgb).
Le fasi sopracitate, verranno introdotte e analizzate in dettaglio nei paragrafi seguenti.

1.3 Revisione della letteratura

A priori, per analizzare le principale tecniche adottate per la predizione dell'abbandono del percorso universitario, è stata eseguita una revisione della letteratura.
Sebbene molti lavori facciano uso di dataset simili a quello in questione, come riportato dettagliatamente da Albreiki et al. in [1] ognuno presenta delle differenze dagli altri, da feature socio demografiche alle differenze nel collezionamento dei dati o più in generale dalle caratteristiche degli studenti e degli istituti accademici.
Analizzando vari lavori tra cui quelli di Rovira et al. [12], Fernandez-Garcia et al. [6]

e Del Bonifro et al. [5] ciò che è possibile evidenziare è l'utilizzo degli stessi modelli di machine learning quali SVM, ANN, RF e Gradient Boosting in generale. Da tali lavori e da tutti quelli analizzati da Albreiki et al. in [1] si conclude che generalmente sono quelli che ottengono performance migliori su questo specifico prolema.
Tuttavia, ciò che va specificato è che le operazioni di preprocessing in tali lavori risultano fondamamentali essendo in grado di influire notevolmente sulle performance del modello oppure nella scelta di applicazione di un modello rispetto ad un altro. Infatti, operazioni di preprocessing come under sampling o over sampling possono modificare la natura stessa del problema, spostando il focus da un problema di rilevamento di anomalie (con dataset non bilanciato) a un problema di classificazione standard (con dataset bilanciato).

1.4 Presentazione dei risultati ottenuti

Per i motivi sopra elencati, i risultati ottenuti non sono stati confrontati con quelli di nessun'altra pubblicazione.
Il lavoro proposto è altamente dipendente dal dataset utilizzato, l'unico confronto possibile è quello con il lavoro proposto da Del Bonifro et al. in [5] in quanto viene utilizzata una versione parziale del dataset corrente.
Dopo queste doverose premesse, va specificato che negli ultimi paragrafi vengono riportati dei confronti fra le varie configurazioni possibili, specificando di volta in volta le varie metriche di performance che sono state tenute in considerazione.

2 Metodo proposto

2.1 Ricerca della soluzione

In questo paragrafo si espongono le principali fasi di progettazione e sviluppo della soluzione finale, spiegando di volta in volta, le motivazioni delle decisioni prese.
Come prima cosa, si è proceduto con la fase di *preprocessing*. Il preprocessing è una delle fasi più importanti di un progetto di machine learning perché permette di preparare i dati in modo che siano adeguati per l'addestramento del modello.
Questa fase include attività come la pulizia dei dati, la normalizzazione, la codifica delle variabili categoriche e la selezione delle features. Il preprocessing aiuta a migliorare la qualità e la coerenza dei dati, a evitare problemi di sovrapposizione o di correlazione tra le variabili e a ridurre l'overfitting del modello. In sintesi, il preprocessing è fondamentale per ottenere un modello di machine learning affidabile e preciso.
Terminata questa fase, con cui si genererà un dataset singolo, si addestreranno vari modelli, si sceglierà poi il migliore in base a delle specifiche metriche di performance e su questo si applicheranno tecniche di interpretability/explainability al fine di comprendere il processo di decisione del modello stesso.

2.1.1 Acquisizione dati

Come primo step della fase di *preprocessing* si è proceduto con l'acquisizione dei dataset forniti, 3 in totale, *stud_2016_2018.xlsx* contenente informazioni sugli studenti, *esami_2016_2018.xlsx* contenente informazioni sugli esami verbalizzati dagli studenti e *EtichetteDBImmatricolati.xlsx* contenente la trascrizione con relativa spiegazione dei nomi delle feature di entrambi i fogli excel.

2.1.2 Gestione valori mancanti

Appurata la presenza di valori mancanti per la feature 'voto_scuola_superiore' e assumendo potesse essere di particolare rilevanza per il task in questione, si è scelto di operare un'operazione di riempimento dei valori mancanti.
Come tecnica di riempimento, al posto di utilizzare la classica media aritmetica, si è scelto di utilizzare la mediana dei valori, in quanto statisticamente più robusta e minormente influenzata da valori anomali.
Durante la fase di esplorazione si è però constatata una forte variazione tra i valori della feature e si scelto di calcolare la mediana localmente, ovvero, è stata calcolata la mediana di 'voto_scuola_superiore' (per ogni studente senza voto) calcolandola tra l'insieme di studenti che hanno in comune 'area_geografica_scuola_superiore' e 'Diploma_scuola_superiore'.

La seconda operazione di riempimento deriva dalla constatazione che per molti studenti non è stato possibile ottenere valori di ISEE oppure quest'ultimi hanno deciso di non presentarlo, sebbene questo non crei problemi per la feature 'Classe_ISEE' che agisce come buon indicatore di reddito familiare, tutti gli studenti con valore 'Classe_ISEE' uguale a 0 o 1 presentano un valore mancante per la feature

'Merito_ISEE'. Proprio per questo, si è proceduto creando una nuova feature artificiale denominata 'contribuzione_tasse' in grado di indicare con un valore compreso nell'intervallo [0, 1] la situazione di contribuzione tasse del singolo studente. L'implementazione del calcolo dei nuovi valori risulta congrua con gli scaglioni delle tasse dell'Università di Bologna negli anni 2016/2017 - 2017/2018 - 2018/2019.
A questo punto tutti gli studenti che presentavano valori mancanti per la feature 'Merito_ISEE' (rimossa dal dataset) avranno un valore di 'contribuzione_tasse' uguale a 1 (massimizzato).

2.1.3 Data transformation

I modelli di machine learning richiedono che tutte le variabili di input e output siano numeriche.
Ciò significa che se tra i dati del dataset vi sono feature categoriche, queste si devono codificare prima di poter addestrare e valutare il modello.
Le due tecniche più popolari sono l'**Ordinal encoding** ed il **One-Hot encoding**. Sebbene in molti casi, si possano utilizzare entrambi, va fatta una doverosa distinzione.
Come riportato in *Feature engineering for machine learning* [15] per le variabili categoriche in cui non esiste alcuna relazione ordinale, l'ordinal encoding, che avviene semplicemente codificando ogni valore con un numero intero, potrebbe non essere sufficiente nel migliore dei casi, o addirittura fuorviante per il modello nel peggiore dei casi.
Forzare una relazione ordinale tramite una codifica ordinale e consentire al modello di assumere un ordinamento naturale tra le categorie può comportare prestazioni scadenti o risultati imprevisti.
In questo caso, è possibile applicare una codifica One-Hot alla rappresentazione ordinale. Si rimuove quindi la feature intera e si aggiunge una nuova feature binaria per ogni valore intero univoco nel feature space.
Seguendo tale principio è stato applicato il One-Hot encoding alle seguenti feature: 'Coorte', 'Diploma_scuola_superiore', 'area_geografica_scuolasuperiore', 'area_geografica_residenza', 'CdS', 'Ambito' e 'Campus'.
Si è invece applicato un ordinal encoding unito a delle trasformazioni per le feature: 'DataNascita', la quale è stata trasformata in 'AnnoNascita' estrapolando l'anno dalla data, 'DurataCorso', il quale è stato convertito in intero, preservando quindi l'ordinamento naturale della relazione e per la feature 'Genere' convertita nei valori numerici 0 e 1.
Come ultima operazione è stata rimossa la feature 'Sede' in quanto presentava valori ridondanti alla feature 'Campus'.

2.1.4 Feature augmentation

Le operazioni di feature augmentation consistono semplicemente nell'inserimento delle informazioni relative agli esami.
Si sono estrapolate le seguenti informazioni per poi essere aggiunte al dataset degli studenti:

- Numero di cfu conseguiti da ogni studente

- Media dei voti degli esami

Durante l'aggregamento si è potuta constatare la presenza di circa 900 studenti per cui non è possibile calcolare la media voto degli esami (oltre i 5500 che non hanno neanche cfu conseguiti).
Si è scelto di riempire la colonna della media voti con la media di tutti gli studenti così da non influire statisticamente mentre si è optato per rimuovere tutti gli studenti per cui non sono presenti informazioni sugli esami.
Si nota come sarebbe stato possibile, in questo caso, aggiungere anche per quest'ultimi valori fittizi (impostando 0 come valore per la feature 'cfu_conseguiti'), tuttavia, più di 5500 elementi su un totale di 50000 rappresenterebbe una percentuale dell'11% finendo per modificare significativamente la distribuzione dei dati, per lo più senza basarsi su nessun dato scientifico, infatti non è possibile sapere a priori se tutti quegli studenti abbiano o meno conseguito dei cfu.

2.1.5 Data normalization

Durante la pre-elaborazione dei dati è importante effettuare la normalizzazione per standardizzare l'intervallo delle variabili (*feature*).
La normalizzazione rende i dati più adatti ad una convergenza e una comparazione, in particolare, il **Min-Max scaling** utilizzato nel progetto è una buona tecnica quando non si è a conoscenza della distribuzione dei dati [7].
Con questo *scaler*, i dati vengono ridimensionati nell'intervallo tra 0 e 1.
La formula è la seguente:

$$z = \frac{x - min(x)}{max(x) - min(x)}$$

2.1.6 Dataset finale

Normalizzato l'intero dataset, si procede salvandolo in un nuovo foglio excel denominato *'final_dataset.xlsx'*. Tale dataset costituisce la versione definitiva su cui i modelli proposti nei successivi paragrafi vengono addestrati.

2.2 Modelli proposti

Dopo aver eseguito un'attenta revisione della letteratura scientifica inerente a problemi di classificazione con dataset non bilanciati ad alta dimensione (sparsi) [9], [8], [13], [3], si è deciso di testare una serie di modelli di machine learning per valutare la loro efficacia nel risolvere il problema in questione.
In particolare, si è scelto di esaminare sette diversi modelli:

1. Support Vector Machines (SVM)
2. Radial Basis Function nuSVM
3. Weighted Class Linear SVM
4. Weighted Samples Linear SVM

5. Random Forest
6. Autoencoder
7. XGBoost

Questa selezione rappresenta una varietà di approcci al problema, che consente di confrontare e valutare i loro risultati. L'obiettivo è quello di fornire una soluzione ottimale tramite l'utilizzo di quest'ultimi in maniera trasparente.

3 Risultati sperimentali

3.1 Tecnologie utilizzate

Lo sviluppo dei modelli di machine learning per risolvere il problema del dropout accademico (classificazione con dataset non bilanciato ad alte dimensioni) è avvenuto utilizzando il linguaggio di programmazione Python e delle librerie Scikit-learn [23], Pandas [22], NumPy [21], Keras [19], Matplotlib [20] e Xgboost [24] mentre alcune tecniche di interpretabilità sono state implementate utilizzando la libreria InterpretML [18], il tutto sfruttando l'ambiente di lavoro in cloud messo a disposizione da Google Colab [16].

3.2 Training data e Test data

Sapendo che lo scopo di un modello di apprendimento automatico è quello di compiere predizioni attendibili su un set di dati mai visto, è di grande importanza specificare come questo set di dati venga ottenuto.
La tecnica più utilizzata è quella dello splitting, ovvero, il dataset iniziale viene diviso in due partizioni [17]:

- Training set: sotto-insieme utilizzato nella fase di addestramento.
- Test set: sotto-insieme utilizzato nella fase di testing, successiva all'addestramento, che ne verifica l'efficacia.

Nell'attuale implementazione, si utilizza la funzione della libreria Scikit-learn 'train_test_split':

```
x_train, x_test, y_train, y_test =
sklearn.model_selection.train_test_split(x, y, stratify=y, test_size=0.25)
```

Specificando la grandezza del test data uguale al 25% del totale e impostando il valore stratify a y; questo parametro di stratificazione esegue uno *splitting* in modo che la proporzione di valori nel campione prodotto sia la stessa della proporzione di valori forniti al parametro di stratificazione.
Ad esempio, se la variabile y è una variabile categorica binaria con valori 0 e 1 e ci sono il 25% di valori 0 e il 75% di valori 1, stratify=y farà in modo che la divisione casuale abbia il 25% di 0 e il 75% di 1.

3.3 Misurazione delle performance

Terminata la fase di addestramento, il modello viene testato sul set di testing generando delle metriche di valutazione. Per i problemi di regressione, le più diffuse sono le seguenti: **Mean Absolute Error**, **Mean Squared Error** e **Root Mean Squared Error**.
Al contrario, nei problemi di classificazione, si adottano metriche di valutazione differenti.
Di seguito vengono brevemente introdotte le metriche schematizzabili nella matrice di confusione (**confusion matrix**) e il **punteggio ROC-AUC** utile in problemi di classificazione in cui la distribuzione degli esempi tra le classi note è altamente distorta (90/10) (**Imbalanced Classification**).

3.3.1 Confusion matrix

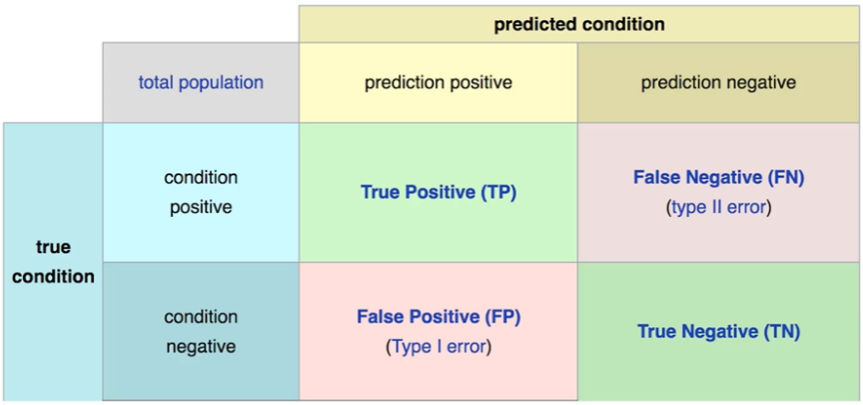
Come riportato nel paragrafo precedente, le metriche di performance nei problemi di classificazione differiscono da quelle di regressione, le più popolari sono:

- **Accuratezza** (Accuracy): è il numero delle predizioni corrette effettuate dal modello diviso il numero totale delle predizioni, è, quindi, la percentuale di predizioni corrette. E' molto utile in presenza di dataset bilanciati, dove le classi (categorie) sono uniformemente distribuite.
- **Richiamo** (Recall): rappresenta l'abilità del modello di predire tutti i casi rilevanti. Si calcola come il numero dei veri positivi diviso la somma dei veri positivi e falsi negativi (true positives / (true positives + false negatives)).
- **Precisione** (Precision): è l'abilità del modello di identificare solo i punti rilevanti. Si calcola come il numero dei veri positivi diviso la somma dei veri positivi e falsi positivi.
- **F1-score**: è utile per trovare un bilanciamento tra precisione e richiamo essendo la media armonica delle due.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

  Si utilizza la media armonica per penalizzare maggiormente grandi differenze tra precisione e richiamo

Queste quattro metriche sono schematizzate nella **matrice di confusione**, una matrice utile per calcolare le metriche più rilevanti in un problema di classificazione. Un esempio di matrice di confusione è riportato nella figura sottostante.



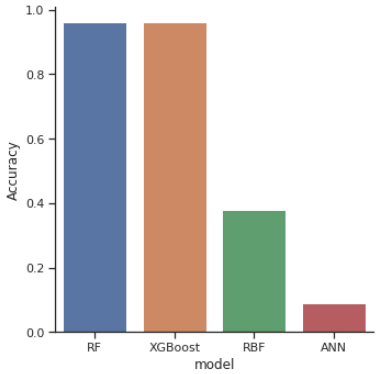
3.3.2 ROC-AUC score

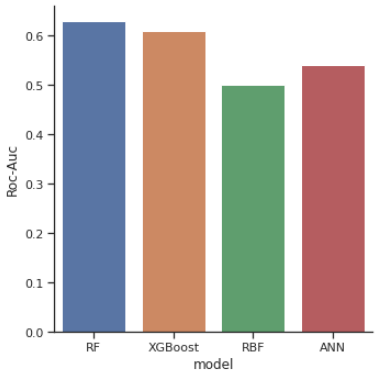
Le metriche di performance di un modello di machine learning che fa classificazione binaria sono utilizzate per valutare l'accuratezza e l'efficacia del modello nell'identificare correttamente le due classi.
Alcune delle metriche più comuni includono l'accuratezza, la precisione, il recall e

la F1-score. Tuttavia, la metrica ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) è particolarmente importante perché tiene conto sia della capacità del modello di identificare correttamente le istanze positive (True Positive), sia della sua capacità di evitare falsi positivi (False Positive). La curva ROC mostra la relazione tra il TPR (True Positive Rate) e il FPR (False Positive Rate) e l'AUC rappresenta la performance del modello in termini di capacità di distinguere tra le due classi. Una ROC-AUC di 1 indica una perfetta separazione delle classi, mentre una ROC-AUC di 0.5 indica che il modello non è in grado di distinguere tra le due classi.

3.4 Risultati

In questo paragrafo sono riportate le perfomance valutate attraverso le metriche di **Accuratezza** (al fine di poter comparare i risultati ottenuti con il lavoro di Del Bonifro et al. [5]) e attraverso il **punteggio ROC-AUC** ritenuta una buona metrica di valutazione per le motivazioni precedentemente riportate.
I modelli riportati nello schema fanno riferimento ai migliori 4, in quanto, per motivi riassuntivi, è stato tenuto in considerazione il modello facente parte delle SVM con performance migliori (RBF nuSVM).





Per i 2 migliori modelli (**Random Forest con 1000 stimatori** e **XGBoost**) vengono riportate le relative matrici di confusione: Risultati schematizzati:

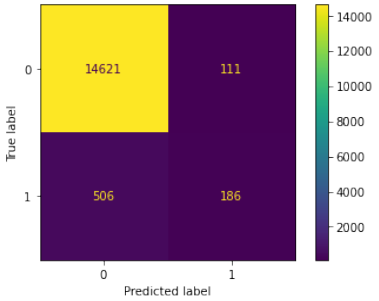


Figure 1: Confusion matrix RF

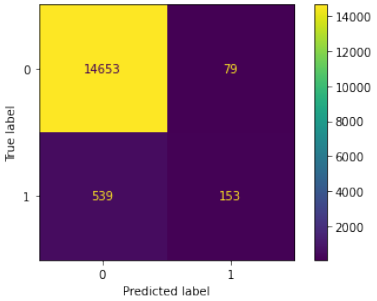


Figure 2: Confusion matrix XGBoost

Modello	Accuratezza	Richiamo	ROC-AUC
XGBoost	0.96	**0.81**	0.61
RF	0.96	0.80	**0.63**

3.5 Studio di ablazione

Il modello XGBoost (eXtreme Gradient Boosting) è una tecnologia di apprendimento automatico che ha guadagnato popolarità in molte applicazioni di classificazione. La recente letteratura scientifica [14], [4] evidenzia la sua particolare utilità in problemi di classificazione con dataset non bilanciati e ad alte dimensioni. Ciò è dovuto alla sua capacità di gestire dati sparsi e di effettuare selezione automatica delle caratteristiche. Inoltre, XGBoost utilizza un algoritmo di gradient boosting che combina molte decision trees, aumentando così la capacità di modellizzare relazioni complesse tra le variabili.

Uno dei principali vantaggi di XGBoost è la sua flessibilità, poiché presenta molti parametri che possono essere ottimizzati per migliorare le prestazioni del modello. Tuttavia, la presenza di così tanti parametri può rendere difficile la scelta della configurazione ottimale. Per questo motivo, XGBoost è un ottimo candidato per uno studio di abalazione con diverse configurazioni. Tale studio consente di esplorare come diverse combinazioni di parametri influiscono sulle prestazioni del modello e di identificare la configurazione ottimale. In sintesi, XGBoost è un modello molto potente e flessibile che offre molte opportunità per migliorare le prestazioni di classificazione in problemi complessi.

3.5.1 GridSearchCV

La configurazione ottimale dei parametri di un modello è un fattore cruciale per ottenere prestazioni ottimali. La libreria scikit-learn offre una funzione di ricerca per griglia, nota come GridSearchCV, che consente di esplorare rapidamente una vasta gamma di combinazioni di parametri e di identificare la configurazione ottimale.
L'utilizzo di GridSearchCV è molto conveniente in quanto automatizza il processo di ottimizzazione dei parametri, consentendo di esplorare rapidamente una vasta gamma di combinazioni di parametri senza la necessità di scrivere codice manualmente. Inoltre, poiché la funzione di ricerca per griglia valuta automaticamente le prestazioni di ogni configurazione, è possibile identificare rapidamente la configurazione ottimale senza la necessità di effettuare molte prove manuali.

Nel caso specifico del modello XGBoost, la libreria GridSearchCV è stata utilizzata per identificare la configurazione ottimale dei parametri. Questa tecnica consiste nell'eseguire una serie di prove con combinazioni diverse di parametri e valutare le prestazioni di ogni configurazione. La configurazione che fornisce le prestazioni migliori viene quindi selezionata come configurazione ottimale.

3.5.2 SMOTE + UnderSampling XGBoost

Come successiva configurazione progettuale, si è pensato di utilizzare le tecniche di **SMOTE** [2] e **UnderSampling** in concatenazione al fine di ottenere un dataset bilanciato.

SMOTE (Synthetic Minority Over-sampling Technique) è una tecnica di bilanci-

amento dei dati che genera esempi sintetici della classe minoritaria, basandosi sui vicini di ogni esempio nel feature space. Ciò aumenta la rappresentazione della classe minoritaria, migliorando la capacità del modello di riconoscere questa classe.

L'UnderSampling, d'altra parte, è una tecnica che consiste nella riduzione del numero di esempi della classe più frequente. Questo rende il dataset, più adatto all'addestramento di un modello affidabile.

L'utilizzo di tali tecniche in combinazione, ha permesso di migliorare notevolmente le performance del modello XGBoost.
Di seguito è riportata la matrice di confusione con alcune metriche di valutazione (accuratezza, richiamo, ROC-AUC score).
E' importante precisare che contrariamente a quanto si possa immaginare il valore di accuratezza è diminuito non per demerito del modello; tale calo è ovviamente da attribuire al bilanciamento del dataset, infatti anche un modello che classificasse sempre 0 ("non abbandono") con il dataset precedente avrebbe ottenuto una accuratezza del 95.51%.

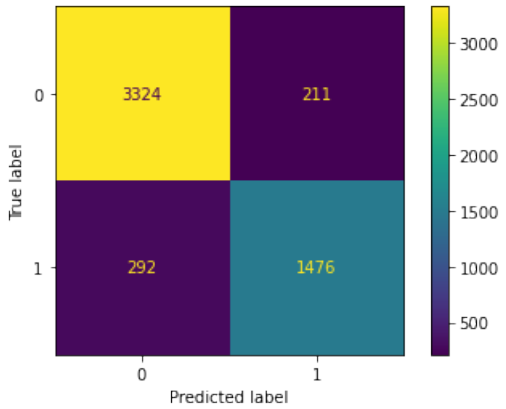


Figure 3: Confusion matrix XGBoost

Modello	Accuratezza	Richiamo	ROC-AUC
XGBoost	0.91	0.90	0.89

3.6 Interpretable ML

La seconda parte del progetto ha riguardato l'applicazione di alcune tecniche di interpretable machine learning al modello XGBoost.
Gli obiettivi dell'utilizzo di tale tecniche hanno riguardato principalmente:

- Debugging del modello: per cercare di comprendere se vi fossero delle inferenze errate o se venissero compiute utilizzando feature che potessero in qualche modo discriminare uno studente.
- Comprensione del funzionamento: cercare di comprendere il "perchè" di una determinata classificazione può garantire la trasparenza del processo di decisione del sistema proposto diminuendo incertezza e diffidenza nell'accettazione dell'output.

Esistono due categorie principali di tecniche di interpretabilità dei modelli di machine learning: quelle globali e quelle locali.

Le tecniche di interpretabilità globale forniscono una comprensione complessiva del modello, ad esempio visualizzando la struttura del modello o i pesi associati alle diverse feature. Queste tecniche sono utili per comprendere come il modello elabori i dati a livello generale, ma non forniscono informazioni sulle singole decisioni prese dal modello.

Al contrario, le tecniche di interpretabilità locale si concentrano sulle singole decisioni prese dal modello, ad esempio attraverso l'analisi di singoli esempi o la visualizzazione della decisione del modello in uno spazio delle feature. Queste tecniche sono utili per comprendere come il modello elabori i dati a livello individuale e per identificare eventuali decisioni che potrebbero essere in conflitto con le esigenze etiche e sociali.

Nei prossimi paragrafi verrà utilizzata la feature importance, i partial dependence plot e dei modelli glassbox e global surrogate come tecniche di interpretabilità globale.
L'utilizzo delle tecniche di interpretabilità locale è stato reso possibile dalla specifica implementazione di quelle globali, infatti sia i modelli glassbox che il modello global surrogate sono stati implementati utilizzando la libreria InterpretML che permette automaticamente anche ispezioni locali.

3.6.1 Feature importance

Brevemente, la **feature importance** di una feature è l'aumento dell'errore di previsione del modello dopo che si è permutato il valore della feature, rompendo la relazione tra la feature e il vero outcome [10].
Di grande importanza, come riportato nel libro Interpretable Machine Learning di Molnar [10] è interpretare correttamente cosa si sta calcolando.
Di seguito vengono riportati due grafici che mostrano 2 diverse metriche, il primo mostra le prime 20 feature importance calcolate con la metrica 'gain' mentre la seconda facendo uso della metrica 'weight'.

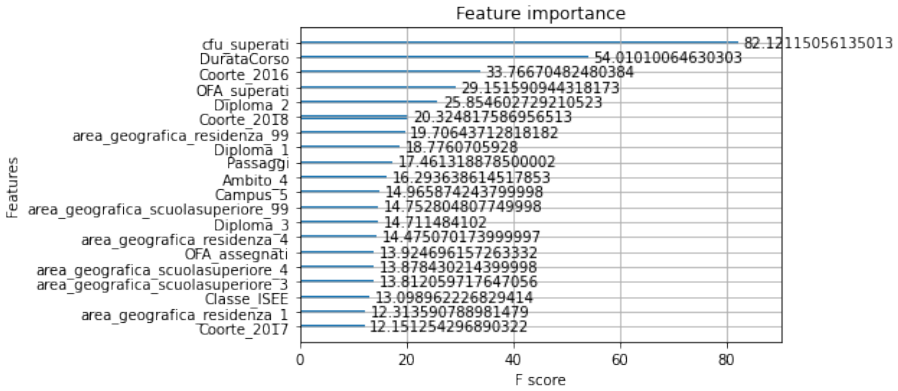


Figure 4: Feature importance: Gain

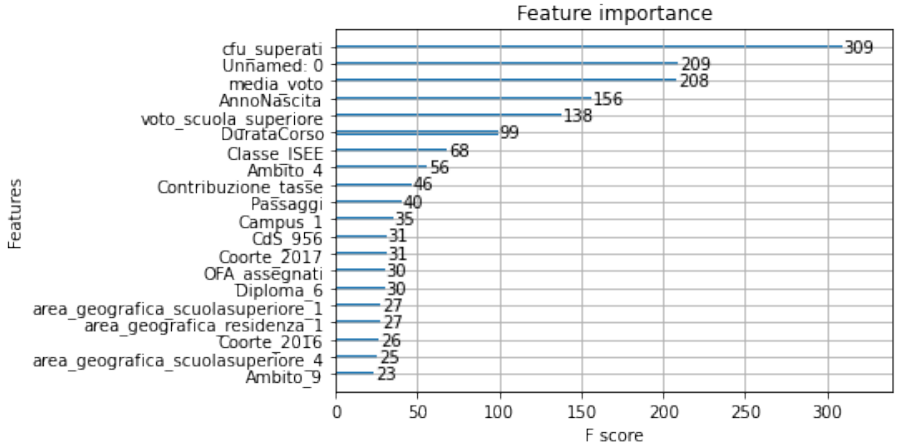


Figure 5: Feature importance: Weight

Il valore di **Gain** è l'aumento dell'accuratezza apportato dalla presenza della feature nei rami in cui si trova.

Il valore **Weight** è la percentuale che rappresenta il numero relativo di volte in cui una paticolare feature appare negli alberi del modello.

Si è calcolata la feature importance con metrica 'weight' poichè si sarebbe potuto pensare che la feature 'cfu_superati' avesse il valore più alto essendo una feature continua. Infatti, solitamente, le feature categoriche (soprattutto binarie) hanno un insieme di valori possibili ridotto rispetto ad una feature continua, questo permette al modello di utilizzare tale feature più volte all'interno di un albero, ad esempio suddividendo il valore in intervalli. Quindi, le feature categoriche solitamente hanno una bassa feature importance basata sulla metrica 'weight' ma, se veramente rilevanti, un'alta feature importance basata sulla metrica 'gain'.
Proprio per questo, ad esempio, è plausibile ipotizzare che la feature 'media_voto' seconda per importanza basata su 'weight' non appaia tra le prime 20 per importanza basata su 'gain'.

3.6.2 Partial dependence plot

Avendo a disposizione un ridotto numero di feature rispetto al numero di partenza (312), si è proseguita l'analisi delle feature indagando la tipologia di relazione esistente.
Lo strumento d'analisi utilizzato in questo paragrafo è il **partial dependence plot**. Il Partial dependence plot (PDP) è una tecnica di interpretazione di modelli di machine learning che permette di visualizzare la dipendenza parziale di una singola feature sulla predizione del modello. Ciò significa che il PDP mostra come varia la predizione del modello man mano che la feature in esame viene modificata, mantenendo costante il valore di tutte le altre feature. Questo strumento è molto utile per comprendere la relazione tra una singola feature e la predizione del modello, e per identificare eventuali effetti non lineari o interazioni tra le feature.

Si è scelto quindi di indagare più a fondo il ruolo delle feature con valori di feature importance più alti, in particolare:

- PDP singolo delle prime 3 feature per weight e gain, per confermare eventuali correlazioni positive o negative.
- Condizione economica: analisi del PDP delle feature (contribuzione_tasse, Classe_ISEE, cfu_superati) e del PDP congiunto delle feature (Classe_ISEE, contribuzione_tasse).
- Condizione pre-universitaria: analisi di come la tipologia di diploma, il voto di diploma influiscano sull'abbandono universitario. Aggiunta e analisi della feature media_voto (anche in relazione a voto_scuola_superiore).
- Condizione geografica: analisi di come l'area geografica di residenza e di scuola superiore influiscano l'abbadono universitario.

Situazione generale

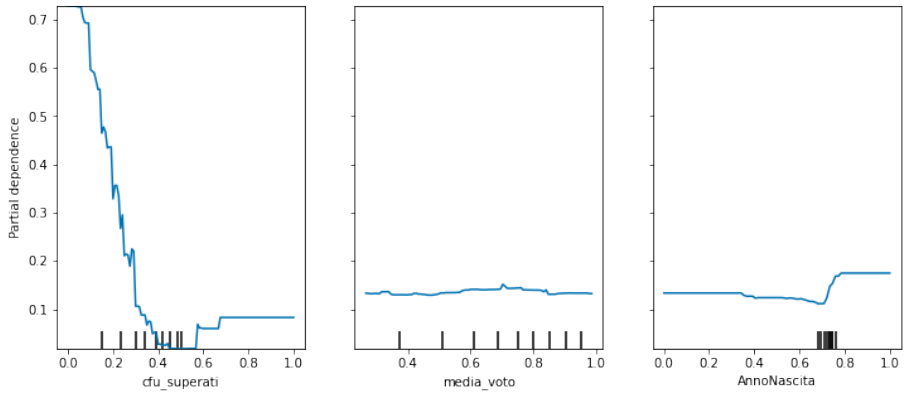


Figure 6: Prime tre feature per 'weight'

Dall'analisi delle prime tre feature per 'weight' (figura 6) si può notare una correlazione negativa tra abbandono universitario e 'cfu_superati' ed una positiva tra

'AnnoNascita' e abbbandono, lasciando presupporre che l'età anagrafica sia quindi un fattore che influisce negativamente sulla scelta di abbandonare.
A conferma della bassa rilevanza di 'media_voto' è possibile notare dal suo PDP la bassa influenza di tale feature.

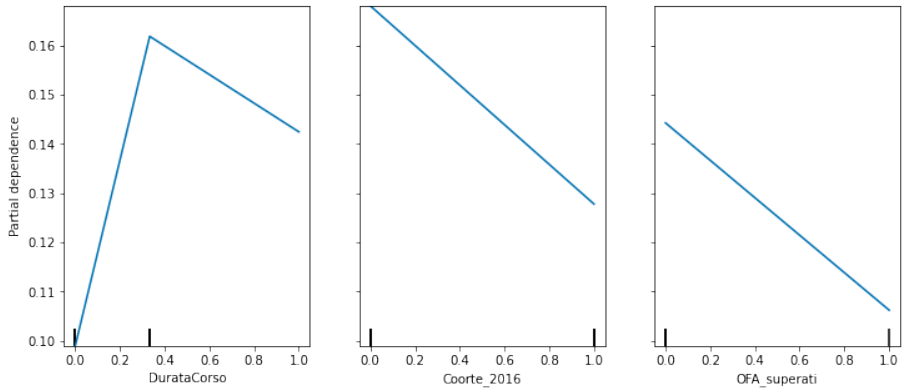


Figure 7: Prime tre feature per 'gain'

Analizzando la feature 'DurataCorso' (primo grafico in figura 7) si evince che la probabilità di abbandono è relativamente bassa per corsi triennali, raggiunge il picco per le lauree magistrali per poi diminuire leggermente per le lauree magistrali a ciclo unico.
L'appartenenza alla 'Coorte_2016' decresce la probabilità di abbandono, tuttavia, tale feature presa singolarmente non è utile alla comprensione del modello, in quanto quasi tutte le feature categoriche binarie abbassano la probabilità di abbandono quando impostate a 1; per analisi più dettagliate occorrerebbe controllare "di quanto" diminuisce in termini relativi, ovvero il coefficiente angolare della retta. Stesso ragionamento può essere proposto per la feature 'OFA_superati'.

Condizione economica
Per studiare gli effetti delle feature che evidenziano la condizione economica di uno studente si è fatto uso (oltre ai classici PDP) di un PDP tra due feature, ovvero un grafico che va a mostrare l'effetto congiunto della variazione dei due valori delle feature (feature interaction).
Attraverso i grafici in questo paragrafo si è concretizzata una delle motivazioni alla base dell'applicazione delle tecniche di interpretabilità, ovvero il debugging.
Infatti, analizzando la PD della feature 'contribuzione_tasse' e della feature 'Classe_ISEE' (figura 8) si può notare un pattern monotono decrescente per la seconda e un pattern non lineare per la prima. Ciò ha indotto ad una più accurata analisi.
Controllando come viene creata la feature è possibile notare che quando non viene presentano l'ISEE o quando questo non è presente, il valore della feature 'contribuzione_tasse' viene massimizzato mentre il valore della feature 'Classe_ISEE' presenta valori bassi (0 oppure 0.125). Tale comportamento spiega la correlazione positiva tra le due feature per valori prossimi a 1.
Una possibile soluzione potrebbe essere lo spostamento dei valori 'isee non presentato' e 'non presente' della feature 'Classe_ISEE' nel dataset originale a codifiche intere con valori più elevati, permettendo di sfruttare l'encoding ordinale in

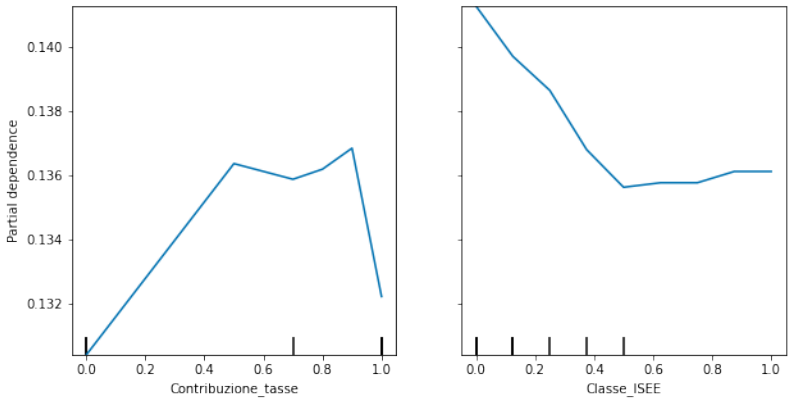


Figure 8: PDP singola di Contribuzione tasse e classe ISEE

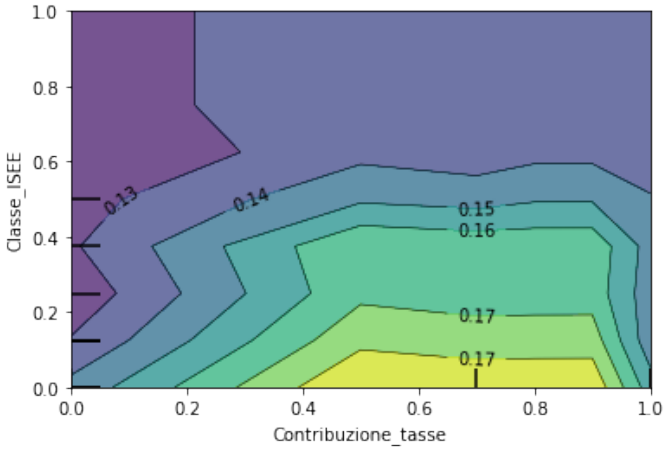


Figure 9: Feature interaction tra Contribuzione tasse e classe ISEE

grado di preservarne appunto l'ordine naturale. Tuttavia, tale soluzione risulterebbe un'assunzione a priori non dimostrabile in quanto non è possibile sapere se un valore di 'Classe_ISEE' uguale a 'non presente' sia dovuto ad una scelta dello studente (causa ISEE elevato) o ad un errore nell'ottenimento dei valori.

Tenendo a mente le considerazione appena fatte, dai grafici è possibile constatare che ad una migliore condizione economica corrispondano valori di probabilità di abbandono relativamente più bassi.

Condizione pre-universitaria

L'analisi della condizione pre-universitaria prende in considerazione le feature categoriche binarie che segnalano l'aver conseguito o meno una determinata tipologia di diploma.
Essendo, appunto, feature categoriche binarie tali PDP (figura 10) non sono altamente informativi, tuttavia, tra tutte le feature, quella che presenta gain maggiore è diploma 2 (Liceo scientifico).

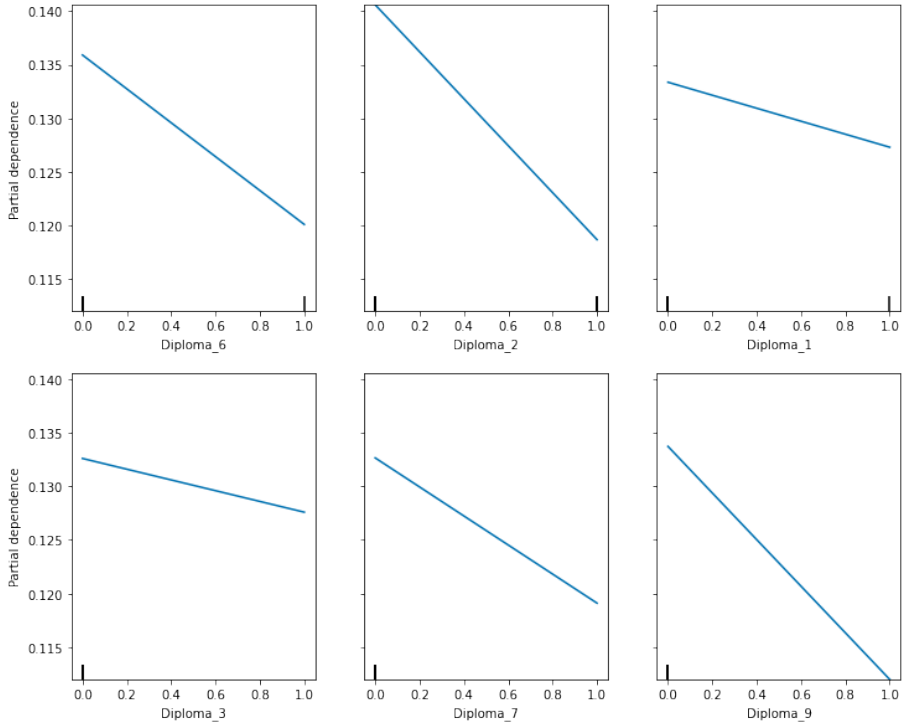


Figure 10: PDP singole per tipologie diplomi

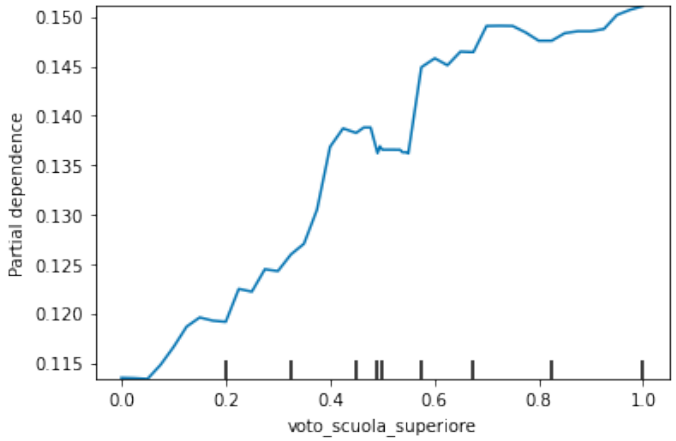


Figure 11: PDP voto scuola superiore

Controintuitivamente, si riporta la PDP della feature 'voto_scuola_superiore' (figura 11), la quale mostra una correlazione positiva tra la probabilità di abbandono e la votazione finale di conseguimento del diploma.

Condizione geografica
Sebbene i PDP singoli riguardanti le condizioni geografiche non siano estremamente informativi, è possibile affermare che c'è un lieve aumento nelle probabilità di abbandono quando uno studente proviene dalla regione Emilia Romagna, al contrario, la feature minimizzante fa riferimento al valore 'Sud e Isole'.

3.6.3 IntepretML - Glassbox

Il pacchetto Python InterpretML fornisce una suite di strumenti per l'interpretazione dei modelli di machine learning, compresi i modelli Glassbox, ovvero modelli che possono essere interpretati e compresi dall'analista. I modelli Glassbox comprendono Explainable Boosting Machine (EBM), Decision Tree e Lime. Ciascuno di questi modelli ha i suoi punti di forza, che dipendono dal tipo di problema che si vuole risolvere.
Di seguito sono riportate le schermate interattive prodotte da InterpretML per ogni tipologia di modello Glassbox.

Explainable boosting machine
L'Explainable Boosting Machine (EBM) è un modello additivo ad albero, a gradiente ciclico, con rilevamento automatico delle interazioni. Le EBM sono spesso accurate quanto i modelli blackbox più avanzati, pur rimanendo completamente interpretabili.
Difatti tale modello ottiene una test accuracy pari a 0.91, uguale a quella del modello XGBoost. E' possibile visulizzare un riassunto della spiegazione globale (figura 12) oppure singolo per ogni feature (figura 13) o feature interaction rilevata.

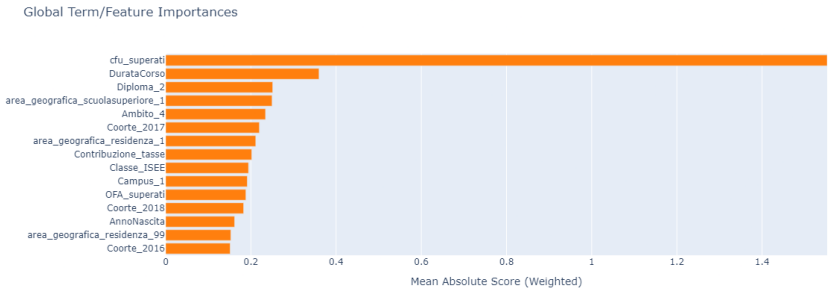


Figure 12: Riassunto EBM

Decision tree
Si tratta di un wrapper leggero per gli alberi decisionali esposti in scikit-learn. Gli alberi decisionali singoli hanno spesso prestazioni deboli del modello, ma sono veloci da addestrare e ottimi per identificare le associazioni. Gli alberi decisionali a bassa profondità sono facili da interpretare, ma diventano rapidamente complessi e incomprensibili con l'aumentare della profondità dell'albero.
Attraverso tale approssimazione, si ottiene un albero intrinsecamente interpretabile (vedi figura 14) con una test accuracy di 0.86.

LIME
Local interpretable model-agnostic explanations (LIME [11]) funziona perturbando ogni singolo datapoint e generando dati sintetici che vengono valutati dal sistema blackbox e infine utilizzati come set di addestramento per il modello glassbox. I vantaggi di LIME sono che si può interpretare una spiegazione nello stesso modo in cui si ragiona su un modello lineare e che può essere utilizzato su quasi tutti

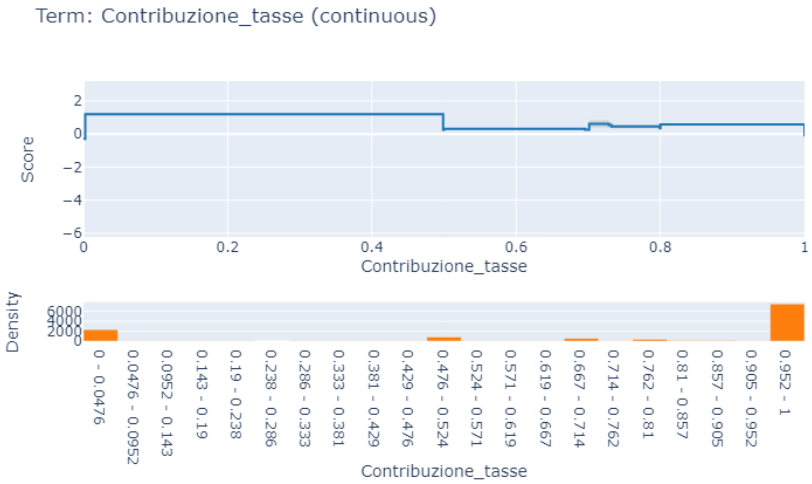


Figure 13: Visualizzazione spiegazione globale feature 'contribuzione tasse'

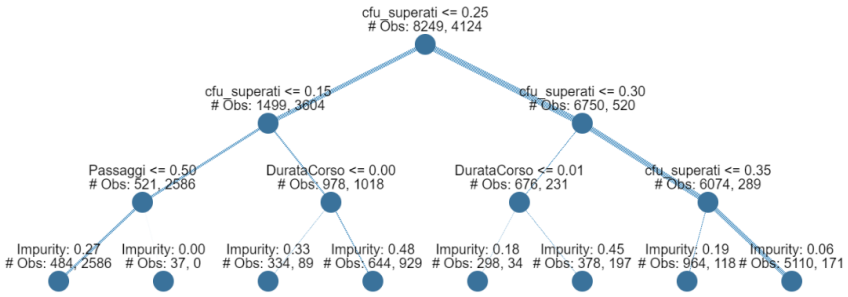


Figure 14: Decision tree

i modelli. D'altra parte, le spiegazioni sono occasionalmente instabili e altamente dipendenti dal processo di perturbazione.
Nell'esempio riportato in figura 15 si può osservare quanto ogni feature abbia influito sul valore di predizione finale di un particolare studente (in questo caso scelto casualmente).
Tale metodo è utile per ricercare approssimazioni locali o analizzare gli errori di predizione. Infatti, come riportato in figura 16 si può notare che per lo studente esaminato il modello ha erroneamente predetto 0.33 (probabilità di abbandono) invece di 1 e che la feature che ha inciso maggiormente è 'DurataCorso' che presenta un valore pari a 0.

3.6.4 InterpretML - Global surrogate

I modelli Glassbox di InterpretML sono considerati troppo semplici poiché si basano su algoritmi di apprendimento supervisionato come Decision tree, Logistic regression o Lime che non sempre sono in grado di gestire la complessità dei modelli di machine learning già esistenti. Questi modelli Glassbox forniscono spiegazioni relative a solo una piccola parte delle decisioni prese dal modello, ma non permettono di

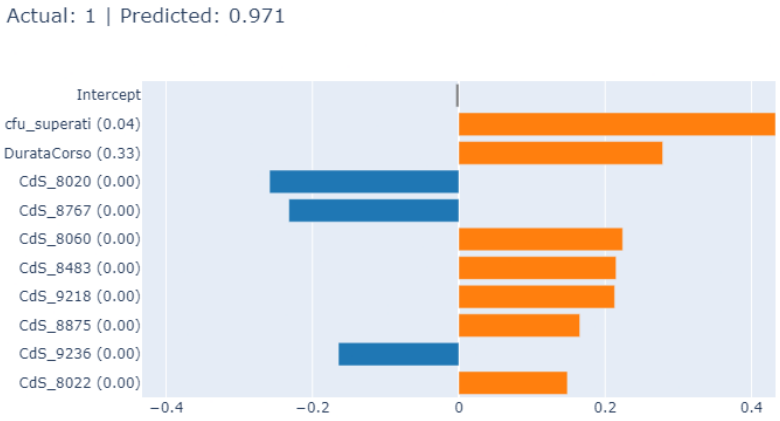


Figure 15: LIME per studente classificato correttamente

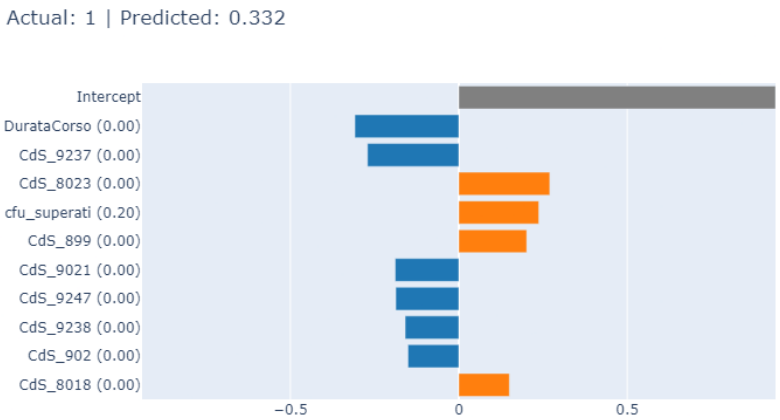


Figure 16: LIME per studente classificato erroneamente

interpretare l'effetto cumulativo delle feature sulle predizioni. Pertanto, sebbene siano utili per comprendere l'influenza di singole feature, questi modelli Glassbox non possono sostituire la necessità di utilizzare tecniche di interpretazione più sofisticate per i modelli di machine learning più complessi.
Come ultimo modello si è pensato di realizzare il Global Surrogate come presentato da Molnar in Interpretable Machine Learning [10] utilizzando però la potenza e la facilità d'uso del pacchetto InterpretML per l'effettiva implementazione.
Un global surrogate model è un modello interpretabile che viene addestrato per approssimare le previsioni di un modello black box. Si possono trarre conclusioni sul modello black box interpretando il modello surrogato.
Come riportato da Molnar in [10] "*solving machine learning interpretability by using more machine learning*".
Gli step per generare un modello surrogato sono:

1. Selezionare un dataset X
2. Per il dataset selezionato, ottenere le predizioni del modello black box
3. Selezionare un modello interpretabile e addestrarlo sul dataset X e sulle predizioni del modello black box
4. Interpretate il modello surrogato

Per applicare questo algoritmo all'attuale caso di studio (sfruttando le potenzialità del pacchetto InterpretML) è stato sufficiente classificare i punti del training set con il modello XGBoost ed addestrare un modello EBM (di interpretML) passandogli il training set e le classificazioni del modello XGBoost come output desiderato.

```
xgb_train = xgb_uo.predict(x_train_uo.values)
xgb_ebm = ExplainableBoostingClassifier()
xgb_ebm.fit(x_train_uo, xgb_train)
```

Questo nuovo modello ottiene un'accuracy in fase di training (rappresentante il grado di approssimazione del modello nel training set) di 0.99, un'accuracy in fase di testing di 0.98 ed un'accuracy finale rispetto il vero dataset di 0.91.
si può concludere che il modello EBM approssima quasi perfettamente il modello XGBoost e grazie a interpretML ne aumenta considerevolmente il grado di interpretabilità e trasparenza.
Tuttavia, una cosa importante che non va tralasciata è che durante l'interpretazione, le conclusione che si traggono riguardano il modello, e non i dati, poichè il modello surrogato non "vede" mai gli outcome reali.
In figura 17 è riportato il riassunto della global explanation del modello global surrogate.

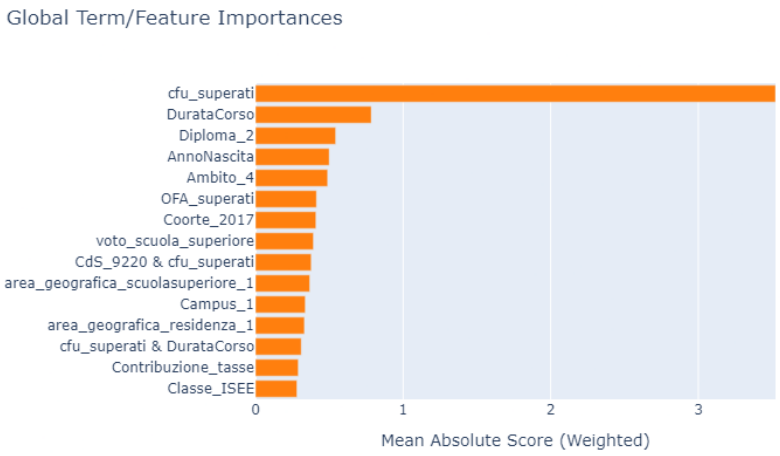


Figure 17: Spiegazione globale del modello global surrogate

4 Discussione e conclusioni

4.1 Discussione dei risultati

Riassumendo, sono state applicate delle procedure di data pre-processing ai dataset forniti dall'Università di Bologna e si è ricercata la tipologia di modello in grado di ottenere le performance migliori su problemi di classficiazione binaria con dataset non bilanciato a molte dimensioni.
Scelto il modello ottimale, XGBoost, si è proceduto effettuando un hyper-parameter tuning attraverso la libreria GridSearchCV. Al fine di ottenere performance migliori si sono concatenate le tecniche SMOTE e UnderSampling per bilanciare il dataset, ottenendo un modello che potesse raggiungere perfomance considerevoli (Accuracy: 0.91, Recall: 0.90, ROC-AUC: 0.89).
A questo punto sono state applicate con successo varie tecniche di interpretabilità (Feature Importance, PDP, Feature Interaction, modelli Glassbox, LIME e Global surrogate) con lo scopo di aumentare l'interpretabilità e l'affidabilità del modello stesso cercando di individuare se presenti bug o bias nel processo di classificazione.
Se i modelli di InterpretML garantiscono l'interpretazione e la comprensione da parte dell'analista, il modello che si è proposto in questo progetto (utilizzando l'algoritmo di Molnar esposto in [10]) viene approssimato da un modello EBM con un'approssimazione del 98%, quindi si può definire, senza problemi, altamente interpretabile. Concludendo, gli obiettivi del progetto possono essere considerati raggiunti.

4.2 Limiti e lavori futuri

Sicuramente tra i limiti della soluzione proposta va menzionato il rischio di overfitting non citato finora. Sebbene le tecniche di interpretazione possano aiutare nel comprendere eventuali situazioni in cui è presente, queste non sono infallibili e sopratutto i risultati devono essere ben interpretati.
Un altro limite consiste nella possibile presenza di istanze artificiali (dovute all'applicazione della tecnica SMOTE) che possano non corrispondere a dati reali (sebbene si sia cercato di ridurre al minimo questo rischio impostanto parametri accurati nella funzione) ad esempio, studenti con durata corso non esistente o apparteneneti a più corsi di studio ecc.
Il limite maggiore rimane dunque l'aver a disposizione un dataset altamente sbilanciato.

Possibili continuazioni di lavoro possono essere:

- Comparare configurazioni del dataset senza una tra le due feature ridondanti (a fronte dell'analisi dei PDP) (Classe_ISEE / contribuzione_tasse).
- Progettare un modello che fa uso di un minor numero di feature, quindi processare nuovamente il dataset includendo solo le feature più informative o applicare metodi di feature selection.

- Applicare congiuntamente ai PDP tecniche quali Individual Conditional Expectation (ICE plot), in grado di fornire più informazioni in caso di interazione tra feature.
- Indagare attraverso tecniche di interpretabilità locale per cercare appunto approssimazioni locali.

References

[1] Albreiki, B., Zaki, N., & Alashwal, H. (2021). A systematic literature review of student'performance prediction using machine learning techniques. Education Sciences, 11(9), 552.

[2] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[3] Chen, Z., Yeo, C. K., Lee, B. S., & Lau, C. T. (2018, April). Autoencoder-based network anomaly detection. In 2018 Wireless telecommunications symposium (WTS) (pp. 1-5). IEEE.

[4] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[5] Del Bonifro, F., Gabbrielli, M., Lisanti, G., & Zingaro, S. P. (2020). Student dropout prediction. In Artificial Intelligence in Education: 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part I 21 (pp. 129-140). Springer International Publishing.

[6] FernándezGarcía, A. J., Preciado, J. C., Melchor, F., RodriguezEcheverria, R., Conejero, J. M., & Sanchez-Figueroa, F. (2021). A real-life machine learning experience for predicting university dropout at different stages using academic data. IEEE Access, 9, 133076-133090.

[7] Géron, A. (2017). Hands-on machine learning with scikit-learn and tensorflow: Concepts. Tools, and Techniques to build intelligent systems.

[8] Haixiang, G., Yijing, L., Shang, J., Mingyun, G., Yuanyue, H., & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. Expert systems with applications, 73, 220-239.

[9] Lin, W. J., & Chen, J. J. (2013). Class-imbalanced classifiers for high-dimensional data. Briefings in bioinformatics, 14(1), 13-26.

[10] Molnar, C. (2020). Interpretable machine learning. Lulu. com.

[11] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

[12] Rovira, S., Puertas, E., & Igual, L. (2017). Data-driven system to predict academic grades and dropout. PLoS one, 12(2), e0171207.

[13] Tanha, J., Abdi, Y., Samadi, N., Razzaghi, N., & Asadpour, M. (2020). Boosting methods for multi-class imbalanced data classification: an experimental review. Journal of Big Data, 7(1), 1-47.

[14] Wang, C., Deng, C., & Wang, S. (2020). Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost. Pattern Recognition Letters, 136, 190-197.

[15] Zheng, A., & Casari, A. (2018). Feature engineering for machine learning: principles and techniques for data scientists. " O'Reilly Media, Inc.".

[16] Google Colab, URL: https://research.google.com/colaboratory/faq.html

[17] Generalization: Peril of Overfitting URL: https://developers.google.com/machine-learning/crashcourse/generalization/peril-of-overfitting

[18] InterpretML, model interpretability, URL: https://interpret.ml/

[19] Keras, deep learning framework, URL: https://keras.io/

[20] Matplotlib, python package, URL: https://matplotlib.org/

[21] NumPy, scientific computing library, URL: https://numpy.org/

[22] Pandas, data manipulation library, URL: https://pandas.pydata.org/

[23] Scikit-learn, URL: https://scikit-learn.org/stable/

[24] XGBoost python package, URL: https://xgboost.readthedocs.io/en/stable/python/index.html