Report Progetto Intelligenza Artificiale
Dropout Accademico
Gruppo: La Triade

Marco Benito Tomasone 1038815
Luca Genova 1038843
Simone Boldrini 1038792

2021-2022

Indice

Capitolo 1

Introduzione

Dropout è un termine con il quale vengono indicati i ragazzi che decidono di abbandonare un percorso di studio prima della fine dello stesso. Il fenomeno del Dropout può avere ripercussioni molto importanti sulla vita dello studente ed è anche un fenomeno economicamente rilevante nella vita delle università. L'Alma Mater Studiorum - Università di Bologna ha raccolto i dati degli iscritti dal 2016 al 2018 per effettuare una ricerca relativa al fenomeno di abbandono degli studi. Lo studio ha lo scopo di andare a valutare se è possibile predire l'abbandono degli studi da parte di uno studente in modo tale da proporre agli studenti delle misure a sostegno di essi (come corsi di potenziamento) per aiutare i ragazzi ed evitare il fenomeno del Dropout.
Il lavoro proposto vuole valutare la possibilità di predire anticipatamente in vari momenti la possibilità che uno studente abbandoni gli studi. I momenti principali sui quali ci andremo a concentrare saranno tre:

1. Al momento dell'immatricolazione, non tenendo quindi conto degli esami svolti durante l'anno, ma considerando solo i dati personali ed economici del ragazzo ed eventuali OFA

2. Dopo la sessione invernale del primo anno, quindi considerando gli esami svolti al primo Marzo

3. Dopo il primo anno di corso, considerando quindi tutti gli esami svolti durante il primo anno di corso.

Questo ci permetterà di andare a valutare con che grado di accuratezza sarà possibile andare a predire la possibilità dell'abbandono da parte di uno studente durante il suo primo anno di studi, per cercare tramite delle misure di sostegno al ragazzo di scongiurare il fenomeno.
Abbiamo per i tre momenti precedentemente descritti utilizzato tre modelli di Machine Learning diversi: la *Regressione Logistica (LR)*, il *Random Forest (RF)* e la *Support*

Vector Machine (SVM). Abbiamo inoltre provato a combinare i risultati di questi tre modelli tramite lo *Stacking Algorithm.*

1.1 Lavori Correlati

Questo lavoro non è il primo che si pone come obiettivo quello di andare a valutare l'abbandono scolastico e non è il primo che cerca di farlo utilizzando il dataset offerto dall'Università di Bologna. Tra i più importanti lavori in questo senso troviamo sicuramente il lavoro di F. Del Bonifro et Al. [2], su cui il nostro lavoro si è fortemente basato e che utilizza RandomForest, SVM e LDA per andare a calcolare la possibilità che uno studente abbandoni o meno gli studi. Tramite diverse ricerche abbiamo trovato uno studio simile svolto all'Università Roma 3 [1]. In questo lavoro troviamo 3 insiemi di suddivisione delle features, il primo contiene tutte le carattteristiche sia accademiche che amministrative, il secondo contiene solo le features amministrative mentre il terzo aggrega le features amministrative insieme alle informazioni statistiche sulle carriere degli studenti, su cui vengono valutati due algoritmi basati sulle reti neurali(NB e kNN). Un lavoro che utilizza un approccio diverso dagli altri e si propone di andare a valutare il Dropout degli studenti a partire da informazioni delle emozioni provate dagli studenti e da altri fattori di emarginalizzazione economica, sociale e accademica è il lavoro proposto da Kadar et Al. [4].

1.2 Struttura del dataset

Il dataset contiene numerose informazioni che riassumiamo in questa tabella:

Campo	**Descrizione**
Coorte	Contiene l'anno di iscrizione dello studente
Data Nascita	Contiene la data di nascita dello studente
Genere	Contiene il genere dello studente o della studentessa
Diploma_Scuola_Superiore	Contiene il tipo di scuola scelta per gli studi superiori
voto_scuola_superiore	Contiene la valutazione ottenuta in sede di diploma di scuola superiore
area_geografica_scuolasuperiore	Contiene un valore indicante l'area geografica della scuola superiore. I valori possibili sono 5: Emilia-Romagna, Altre Regioni del Nord, Centro, Sud e Isole ed Estero.
area_geografica_residenza	Contiene un valore indicante l'area geografica di residenza. Sono possibili gli stessi valori del campo area_geografica_scuolasuperiore

Classe_ISEE	Contiene 9 valori possibili: non disponibile (Coorte 2016/17), ISEE non presentato, meno di 13.000, 13.000-23.000, 23.000-33.000, 33.000-45.000, 45.000-60.000, 60.000-70.000, oltre 70.000
Merito_ISEE	Contiene il valore di ISEE presentato durante la fase di iscrizione
OFA_assegnati	Contiene un valore booleano che descrive il fatto che siano stati assegnati o meno degli OFA (Obblighi Formativi Aggiuntivi)
OFA_superati	Contiene un valore booleano che descrive il fatto che siano stati superati o meno gli OFA
CdS	Descrive il corso di studi di appartenenza
TipoCorso	Descrive se si tratta di un corso di Laurea Triennale, Magistrale o Magistrale a ciclo unico
Ambito	Contiene un codice rappresentante la scuola di appartenenza del corso di studi (es. Scienze o Ingegeria)
Sede & Campus	Sono due colonne rappresentanti in testo e codice la sede nel quale il corso di studi viene erogato
Abbandoni	Contiene un valore booleano indicante se si è effettuato un abbandono del corso di studi
Passaggi	Contiene un valore booleano indicante se si è effettuato un abbandono del corso di studi
Trasferimenti	Contiene un valore booleano indicante se si è effettuato un abbandono del corso di studi

Un secondo file contiene invece per ogni studente la lista degli esami sostenuti. I campi contenuti sono:

Campo	**Descrizione**
CdS	Descrive il corso di studi di appartenenza
Cod.Materia	Codice identificativo della materia
Materia	Il nome della materia dell'esame sostenuto
Giorno Esame	Data in formato *gg/mm/yyyy* del giorno di sostenimento dell'esame
Voto_se_numerico	Campo rappresentante il voto ricevuto all'esame, se NaN si riferisce ad esami come le idoneità che non rilasciano voti.
CFUsuperati	Numero di CFU dell'esame sostenuto

1.3 Suddivisione del lavoro

Questo progetto è stato svolto in un gruppo di tre persone. Nella fase iniziale del lavoro abbiamo lavorato tutti e tre a stretto contatto per il preprocessing dei dati e la creazione di tutti i diversi dataset. Successivamente, ognuno ha applicato il proprio modello ai vari preprocessing precedentemente svolti e abbiamo poi unito e confrontato i risultati. Rispettivamente:

- Il modello di Regressione Logistica è stato utilizzato da Simone Boldrini
- Il modello Random Forest è stato utilizzato da Marco Benito Tomasone
- Il modello Support Vector Machine è stato utilizzato da Luca Genova

La motivazione della scelta di lavorare insieme per l'ottenimento dei vari preprocessamenti è legata alla volontà di unire le idee di tutti i componenti del gruppo e uniformarle per ottenere poi risultati comparabili tra i vari modelli.

1.4 Risultati Ottenuti

I risultati ottenuti dal nostro studio mostreranno, come ovviamente prevedibile, che è più facile prevedere l'abbandono di uno studente alla fine del primo anno piuttosto che a metà del primo anno, e che è più facile predire l'abbandono di uno studente a metà anno piuttosto che all'inizio dell'anno. Il modello che ottiene i risultati migliori (seppur con una differenza leggerissima rispetto agli altri) è invece il modello Random Forest.

Capitolo 2

Metodo Proposto

Questo studio ha come obiettivo quello di andare a valutare la possibilità di predire l'abbandono universitario di uno studente prima che questo avvenga. Per farlo siamo andati a valutare tre possibili momenti di previsione, il momento dell'iscrizione, la fine del primo semestre del primo anno di corso e la fine del primo anno di corso. La scelta sui tre modelli da utilizzare, cioè Regressione Logistica, Random Forest e Support Vector Machine, è stata dettata da interessi personali nello studio del funzionamento di questi modelli e dal fatto che questi siano i modelli più utilizzati in letteratura per risolvere problemi simili di predizione. Inoltre, un'altro importante motivo alla base di questa scelta è il fatto che siano stati utilizzati dall'altro lavoro che utilizza il Dataset dell'UniBo [2] per avere un riferimento per il confronto dei risultati ottenuti.
Successivamente abbiamo fatto un tentativo di Ensamble Learning utilizzando lo Stacking Algorithm con l'idea di migliorare le prestazioni rispetto ai tre modelli.

2.1 Regressione Logistica

La regressione logistica é un modello di regressione non lineare utilizzata quando la variabile dipendente é di tipo binario $[0, 1]$. Con questo modello stabiliamo la probabilitá con cui un'osservazione puó generare uno o l'altro valore della variabile dipendente; puó essere utilizzato inoltre per classificare le osservazioni in due categorie(come nel nostro caso), facendo riferimento a un problema di classificazione. A differenza dei modelli di regressione lineare che sono utilizzati per identificare la relazione tra una variabile dipendente continua e una o più variabili indipendenti, la regressione logistica stima la probabilitá del verificarsi di un evento, sulla base di uno specifico dataset di variabili indipendenti; verso un insieme di dominio della variabile dipendente $[0, 1]$. Una volta calcolato il vettore $\widehat{\beta}$ ossia la stima del vettore dei parametri β, é possibile procedere alla stima della probabilitá p. Per definizione questa probabilitá é anche il valore atteso di

Y.

$$\widehat{p} = E[Y|X] = \bigwedge(X^T\widehat{\beta}) = \frac{e^{X^T\widehat{\beta}}}{1 + e^{X^T\widehat{\beta}}}$$

dove:

- Y é la variabile dipendente binaria
- X il vettore di variabili indipendenti(o regressori) $X_1, \ldots, X_k$
- β il vettore di parametri $\beta_0, \ldots, \beta_k$
- $\bigwedge$ é la funzione di ripartizione della distribuzione logistica standard
- e il numero di eulero

La regressione logistica appartiene alla famiglia di modelli supervisionati, inoltre, é considerato un modello discriminante che prova a distinguere tra classi(o categorie). La regressione logistica puó essere incline all'`overfitting` in particolare quando é presente un numero elevato di variabili predittive nel modello.

2.2 SVM

Le macchine a vettori di supporto (support-vector machines) sono dei modelli di apprendimento supervisionato associati ad algoritmi di apprendimento per la regressione e la classificazione. Dato un insieme di esempi per l'addestramento, ognuno dei quali etichettato con la classe di appartenenza fra le due possibili classi, un algoritmo di addestramento per le SVM costruisce un modello che assegna i nuovi esempi a una delle due classi, ottenendo quindi un classificatore lineare binario non probabilistico. Un modello SVM è una rappresentazione degli esempi come punti nello spazio, mappati in modo tale che gli esempi appartenenti alle due diverse categorie siano chiaramente separati da uno spazio il più possibile ampio. I nuovi esempi sono quindi mappati nello stesso spazio e la predizione della categoria alla quale appartengono viene fatta sulla base del lato nel quale ricade.

Oltre alla classificazione lineare è possibile fare uso delle SVM per svolgere efficacemente la classificazione non lineare utilizzando il metodo kernel, mappando implicitamente i loro ingressi in uno spazio delle caratteristiche multi-dimensionale.

In questo lavoro abbiamo utilizzato *SVC* di *sckit-learn* e siamo andati a selezionare i parametri che più influivano sui risultati e solo dopo siamo andati a ricercare il valore migliore per essi (tuning).

2.2.1 Tuning degli iperparametri

Nei modelli di Machine Learning ci sono alcuni parametri, noti come Hyperparameters, che non possono essere appresi direttamente dai dati. Sono comunemente scelti dagli esseri umani sulla base di qualche intuizione o prova. Questi parametri mostrano la loro importanza migliorando le prestazioni del modello come la sua complessità o il suo tasso di apprendimento. I modelli possono avere molti iperparametri e trovare la migliore combinazione di parametri può essere trattato come un problema di ricerca.

SVM ha anche alcuni iperparametri (come i valori C o gamma da usare) e trovare l'iperparametro ottimale è un compito molto difficile da risolvere. Ma può essere trovato semplicemente provando tutte le combinazioni e vedere quali parametri funzionano meglio. L'idea principale alla base è quella di creare una griglia di iperparametri e provare tutte le loro combinazioni.

GridSearch

Scikit-learn ha questa funzionalità integrata con *GridSearchCV*. GridSearchCV prende un dizionario che descrive i parametri che potrebbero essere provati su un modello per addestrarlo. La griglia dei parametri è definita come un dizionario, dove le chiavi sono i parametri e i valori sono le impostazioni da testare.
Ci siamo soffermati sui tre parametri che abbiamo ritenuto più importanti : *C, kernel e gamma.*
Infine abbiamo scelto come kernel quello lineare e lasciato perdere il gamma (tenendo quello di default) data la natura del problema, e siamo quindi andati alla ricerca dell'iperparametro ottimo **C** utilizzando il dataset bilanciato (vedi sezione 3.6).

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV
# defining parameter range
param_grid = {'C': np.logspace(-3, 2, 6), 'kernel': ['linear']}
svm_estimator = svm.SVC()
grid = GridSearchCV(svm_estimator, param_grid, refit = True, verbose =
3, n_jobs=-1)
# fitting the model for grid search
grid.fit(X_train, y_train)
```

Come si può vedere dal codice il range di $C = \{0.001, 0.01, 0.1, 1.0, 10.0, 100.0\}$ e questo è stato scelto in base a delle ricerche nella letteratura.

```
Fitting 5 folds for each of 6 candidates, totalling 30 fits

[CV 1/5] END .....C=0.001, kernel=linear;, score=0.633 total time= 6.1min
[CV 3/5] END .....C=0.001, kernel=linear;, score=0.612 total time= 8.0min
[CV 4/5] END .....C=0.001, kernel=linear;, score=0.611 total time= 9.2min
[CV 2/5] END .....C=0.001, kernel=linear;, score=0.640 total time=10.2min
```

```
[CV 5/5] END ............C=0.001, kernel=linear;, score=0.589 total time= 8.9min
[CV 1/5] END ............C=0.01, kernel=linear;, score=0.639 total time=10.7min
[CV 2/5] END ............C=0.01, kernel=linear;, score=0.660 total time=13.0min
[CV 3/5] END ............C=0.01, kernel=linear;, score=0.627 total time=13.3min
[CV 4/5] END ............C=0.01, kernel=linear;, score=0.607 total time=13.7min
[CV 2/5] END ............C=0.1, kernel=linear;, score=0.652 total time=16.6min
[CV 5/5] END ............C=0.01, kernel=linear;, score=0.601 total time=23.7min
[CV 3/5] END ............C=0.1, kernel=linear;, score=0.618 total time=15.2min
[CV 1/5] END ............C=0.1, kernel=linear;, score=0.628 total time=23.2min
[CV 4/5] END ............C=0.1, kernel=linear;, score=0.604 total time=14.8min
[CV 5/5] END ............C=0.1, kernel=linear;, score=0.602 total time=17.0min
[CV 1/5] END ............C=1.0, kernel=linear;, score=0.612 total time=17.2min
[CV 2/5] END ............C=1.0, kernel=linear;, score=0.666 total time=25.4min
[CV 3/5] END ............C=1.0, kernel=linear;, score=0.624 total time=17.4min
[CV 4/5] END ............C=1.0, kernel=linear;, score=0.608 total time=15.2min
[CV 5/5] END ............C=1.0, kernel=linear;, score=0.611 total time=14.2min
[CV 3/5] END ............C=10.0, kernel=linear;, score=0.624 total time=13.2min
[CV 1/5] END ............C=10.0, kernel=linear;, score=0.624 total time=17.1min
[CV 2/5] END ............C=10.0, kernel=linear;, score=0.651 total time=27.1min
[CV 5/5] END ............C=10.0, kernel=linear;, score=0.607 total time=14.7min
[CV 4/5] END ............C=10.0, kernel=linear;, score=0.616 total time=29.5min
[CV 1/5] END ...........C=100.0, kernel=linear;, score=0.615 total time=20.0min
[CV 4/5] END ...........C=100.0, kernel=linear;, score=0.608 total time= 9.8min
[CV 2/5] END ...........C=100.0, kernel=linear;, score=0.643 total time=15.5min
[CV 3/5] END ...........C=100.0, kernel=linear;, score=0.622 total time=14.3min
[CV 5/5] END ...........C=100.0, kernel=linear;, score=0.605 total time=12.4min

Best params : {C=0.01, kernel='linear'}
```

E quindi è stato scelto $C = 0.01$.

2.3 Random Forest

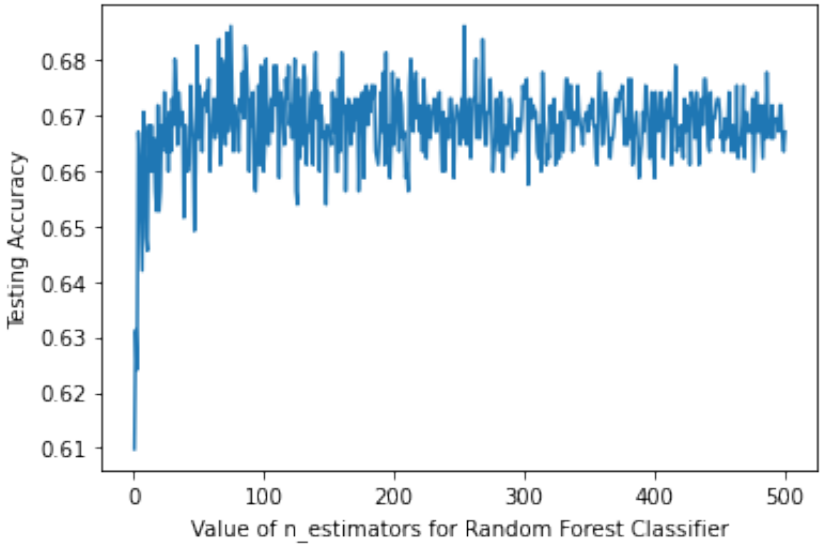
L'algoritmo *Random Forest* o *Random Decision Forest* è un algoritmo di classificazione e regressione che opera costruendo una moltitudine di alberi di decisione durante la fase di training. Per i task di classificazione l'output sarà la classe più selezionata dagli alberi. Per i task di regressione l'output sarà la media delle predizioni dei singoli task. In questo lavoro, essendo un lavoro di classificazione abbiamo utilizzato il *RandomForestClassifier* di *sckit-learn*. Il lavoro è stato organizzato in questo modo:

1. Utilizzando solo i dati personali degli utenti ed escludendo quelli degli esami superati e dei CFU raggiunti (il dataset bilanciato al momento dell'iscrizione, 3.6) sono state provate varie modifiche dei parametri del modello, cambiando il dato relativo al *numero di estimatori.*

2. Successivamente, scelto il parametro che offre risultati migliori sono state provate diverse configurazioni del dataset.

Nella prima fase del lavoro siamo andati quindi a valutare quanto il cambiamento degli iperparametri del modello incidessero sui risultati. Avendo notato come il parametro che

incidesse di più sul cambiamento dei risultati sia *n_estimators* e avendo notato come il variare di altri parametri quali *max_depth, min_samples_leaf* incidano poco sul risultato, per motivi di potenza di calcolo, abbiamo preferito effettuare il tuning solo sull'iperparametro n_estimators. Per prima cosa eravamo interessati a vedere di quanto potesse variare l'accuratezza al variare del parametro. Abbiamo, quindi, costruito un grafico che mette in relazione l'accuratezza al numero di estimatori nel modello. I tentativi di calcolare questo rapporto anche fino a valori molto alti fallivano a causa di esecuzioni troppo lunghe per la nostra potenza di calcolo. Per avere un'idea del fenomeno siamo andati quindi a calcolare l'accuracy in base al numero di alberi per tutti i valori da uno a cinquecento. Mostriamo ora il grafico ottenuto:



Come si può notare dal grafico piccole variazioni del parametro possono far variare il risultato di accuratezza ottenuto. Ci siamo chiesti quindi se valori molto piu grandi di questo iperparametro potessero cambiare sensibilmente la situazione. Abbiamo utilizzato in questo caso, il metodo *GridSearchCV*. Tramite questo metodo è possibile definire una griglia di valori di iperparametri sui quali randomicamente verrà fatto train e test di un modello applicando una k-fold cross validation. La ricerca è stata effettuata sui valori da 1 a 500 e su 1000, 10000, 50000, 100000 con un k pari a 3. Alla fine dell'esecuzione abbiamo stampato quindi il valore di *best_params_* che è risultato pari a 207. È stato quindi utilizzato sempre questo come parametro in tutti i modelli creati successivamente. Per garantire, inoltre, una riproducibilità dei risultati è stato impostato un *RandomState* pari a 42.

```
[ ]  print(rf_random.best_params_)

     {'n_estimators': 207}
```

2.4 Ensemble Learning: Stacking Algorithm

Abbiamo anche provato una metodologia per aumentare le prestazioni del sistema utilizzando modelli multipli per ottenere una migliore prestazione predittiva rispetto ai modelli da cui è costituito, lo **Stacking algorithm**.
Stacking o Stacked Generalization è un algoritmo di apprendimento automatico di insieme. Utilizza un algoritmo di meta-apprendimento per apprendere come combinare al meglio le previsioni di due o più algoritmi di apprendimento automatico di base.
L'architettura di un modello di impilamento (stacking model) coinvolge due o più modelli di base, spesso indicati come modelli di livello 0 e un meta-modello che combina le previsioni dei modelli di base indicati come modello di livello 1:

- Modelli di livello 0 (modelli di base): i modelli si adattano ai dati di addestramento e le cui previsioni vengono compilate.
- Modello di livello 1 (Meta-Modello): modello che impara a combinare al meglio le previsioni dei modelli di base.

Capitolo 3

Risultati Sperimentali

3.1 Tecnologie utilizzate

Per tutti gli esperimenti è stato utilizzato il linguaggio Python 3.7.13 con l'utilizzo del tool *sckit-learn* che offre una serie di modelli di Machine Learning. Per quanto riguarda il bilanciamento dei dati è stata utilizzato il tool *imbalanced-learn*. Per l'esecuzione degli esperimenti è stato utilizzato invece Google Colaboratory nella sua versione gratuita.

3.2 Metriche di valutazione

Definiamo innanzitutto la nostra matrice di confusione che sarà così fatta:

Confusion matrix

	Positive	**Negative**
Positive	TP	FP
Negative	FN	TN

Definiamo a questo punto:

- **TP:** i valori dropout effettivamente riconosciuti come dropout dal modello
- **FP:** i valori non dropout riconosciuti come dropout dal modello
- **FN:** i valori dropout riconosciuti come non dropout dal modello
- **TN:** i valori non dropout effettivamente riconosciuti come non dropout dal modello

Le metriche di valutazione utilizzate sono l'*accuracy, la precision, la recall, l'F1, la macro Avg e la weighted Avg.*

Accuracy

L'accuracy è il rapporto tra le predizioni corrette sul numero di predizioni totali.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

Precisione

La precisione è il rapporto tra i valori positivi previsti correttamente e i valori positivi previsti. La precisione evidenzia le previsioni positive corrette tra tutte le previsioni positive. Una precisione alta implica un basso tasso di falsi positivi.

$$Precision = \frac{TP}{TP + FP}$$

Recall

La recall è il rapporto tra i valori positivi correttamente previsti e i valori positivi effettivi. Recall mette in evidenza la sensibilità dell'algoritmo cioè tra tutti i positivi effettivi quanti sono stati effettivamente rilevati. Un livello di recall elevato significa che un algoritmo restituisce la maggior parte dei risultati rilevanti.

$$Recall = \frac{TP}{TP + FN}$$

F1-score

Il punteggio F1 é la media ponderata di Precisione e Recall. È una metrica molto più sofisticata dell'accuratezza perché tiene conto sia dei falsi positivi che dei falsi negativi. L'accuratezza è adatta solo quando sia i falsi positivi che i falsi negativi hanno un costo simile.

$$\text{F1-Score} = \frac{2 * (Recall * Precision)}{Recall + Precision}$$

Macro AVG

Per la precision, la recall e l'F1-Score abbiamo calcolato la media.

$$Score_{MacroAVG} = \frac{\text{Score}_{class_1} + \text{Score}_{class_0}}{2}$$

Dove $Support_{Class_i}$, con $i \in 0, 1$ è il numero di occorrenze effettive della classe i nel set di dati specificato.

Weighted AVG

Per la precision, la recall e l'F1-Score abbiamo calcolato la media pesata.

$$Score_{WeightedAVG} = \frac{\text{Score}_{class_1} * Support_{Class_1} + \text{Score}_{class_0} * Support_{Class_0}}{Support_{Class_1} + Support_{Class_0}}$$

Dove $Support_{Class_i}$, con $i \in 0, 1$ è il numero di occorrenze effettive della classe i nel set di dati specificato.

3.3 Studio di Ablazione

Presenteremo ora tutti i vari preprocessamenti dei dati che abbiamo eseguito e i risultati dei modelli relativi

3.4 Preprocessing Base

Alcune colonne avevano dei valori contenenti un ordine implicito che potevano influenzare negativamente il modello. Abbiamo rimpiazzato i codici relativi alle segueti features con i nomi delle features, così da poter utilizzare la funzione *get_dummies()* ed ottenere un *one-hot encoding*:

- Diploma_scuola_superiore
- area_geografica_scuolasuperiore
- area_geografica_residenza
- Ambito

```
features["Diploma_scuola_superiore"] = features["Diploma_scuola_superiore"].map({1: "Classico", 2: "Scientifico", 3: "Linguistico", 4: "Magistrale", 5: "Artistico", 6: "Tecnico", 7: "Professionale", 8: "Altro_italiano", 9: "Estero", 99: "Non_disponibile"})
features["area_geografica_scuolasuperiore"] = features["area_geografica_scuolasuperiore"].map({1: "Emilia_romagna", 2: "Nord", 3: "Centro", 4: "Sud_isole", 5: "Estero", 99: "Non_disponibile"})
features["area_geografica_residenza"] = features["area_geografica_residenza"].map({1: "Emilia_romagna", 2: "Nord", 3: "Centro", 4: "Sud_isole", 5: "Estero", 99: "Non_disponibile"})
features["Ambito"] = features["Ambito"].map({1: "Economia", 2: "Farmacia", 3: "Giurisprudenza", 4: "Ingegneria", 5: "Lingue", 6: "Medicina", 7: "Veterinaria", 8: "Psicologia", 9: "Scienze", 10: "Scienze_agroalimanetari", 11: "Scienze_educazione_formazione", 12: "Scienze_motorie", 13: "Scienze_politiche", 14: "Scienze_statistiche", 15: "Sociologia", 16: "Studi_umanistici"})
```

Abbiamo sostituito la data di nascita creando una nuova colonna con l'età dello studente, tramite una differenza tra la data di nascita e l'anno della coorte associato a quello studente.
Abbiamo ritenuto importante l'età ai fini dell'addestramento principalmente per la natura del problema, poichè potremmo supporre che uno studente che inizia l'università ad un'età maggiore sia più propenso ad abbandonare se incontra delle difficoltà. Abbiamo sostituito i valori **NaN** della feature *voto_scuola_superiore* con la media. Inoltre sono state eliminate le tuple contente almeno un NaN e droppate le colonne non utili ai fini dell'addestramento:

- ID_Studente: poichè si tratta semplicemente di un valore crescente che indica il numero della tupla. Come vedremo in futuro utilizzarlo può creare un bias nel modello.
- Classe_ISEE: poichè c'è la colonna Merito_ISEE contentente già il valore dell'ISEE
- DataNascita: poichè verrà sostituita con l'età dello studente al primo anno di corso
- Coorte: poichè si tratta di un valore appartenente all'insieme 2016,2017,2018 inutile ai fini dell'addestramento.

3.4.1 Divisione del dataset

Abbiamo diviso il dataset in traing e test relativamente 80% e 20% poichè tramite varie ricerche in letteratura è uno dei rapporti più utilizzati per il training di un modello. La scelta di effettuare una divisione in percentuali 80-20 è stata trovata in letteratura in un articolo pubblicato da Gholamy et Al. [3].

3.5 Momento dell'iscrizione

Per quanto riguarda il modello predittivo degli studenti al momento dell'iscrizione abbiamo eliminato dal dataset la colonna OFA_Superati poichè al momento dell'iscrizione di uno studente non sappiamo ancora se questo supererà i propri OFA o meno.

Evaluating algorithms

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.91	0.99	0.95	0.91	0.99	0.95	0.91	1.00	0.95	4426
1	0.23	0.02	0.04	0.35	0.03	0.06	0.00	0.00	0.00	431
Accuracy	0.9065266625488985			0.9112620959439983			0.91476219			4857
Macro AVG	0.57	0.51	0.50	0.63	0.51	0.51	0.46	0.50	0.48	4857
Weighted AVG	0.85	0.91	0.87	0.86	0.91	0.87	0.83	0.91	0.87	4857

Tabella 3.1: P = Precision, R = Recall e F1 = F1-score

Come si vede i risultati in accuracy sono ottimi, arrivando ad un 90% circa in tutti i modelli. Se si va ad analizzare peró i risultati in precision, recall e f1-score degli 1, quindi degli studenti che droppano, si può notare come queste siano vicine allo zero, mostrando come il risultato alto in accuracy sia legato al corretto riconoscimento degli studenti che non abbandonano gli studi. Per ovviare a questo problema, da questo momento in poi bilanceremo il dataset 50-50 tra studenti che abbandonano e non.

3.6 Momento dell'Iscrizione Bilanciato

Notando come i risultati ottenuti precedemente abbiano dei risultati molto bassi per quanto riguarda la classe dei Dropout, abbiamo cambiato strategia e deciso di andare a bilanciare il dataset con un bilanciamento 50-50. Abbiamo bilanciato il dataset con approccio under-sampling pareggiando il numero di dropout e non dropout. Abbiamo optato per l'under-sampling e non per l'over-sampling poichè crediamo che andare a generare dati sintetici non sia adatto data la natura del problema.
Abbiammo eseguito diverse prove con due tipi di bilanciamento diversi:

- il primo fatto da noi, dove andiamo a prendere i primi n dropout della classe 1 e della classe 0 (Codice 3.1)
- il secondo utilizzando la classe RandomUnderSampler di imblearn (Codice 3.2)

```
#BILANCIAMENTO 1
dropout = pd.DataFrame()
noDropout = pd.DataFrame()

for ind in features.index:
```

```
    studente = pd.DataFrame(features.loc[[ind]])
    if features["Abbandoni"][ind] == 1:
        dropout = pd.concat([studente, dropout], ignore_index=True, axis = 0)
    else:
        noDropout = pd.concat([studente, noDropout], ignore_index=True, axis = 0)

#Balancing: I'm going to get as many non-dropouts as there are dropouts
final = pd.DataFrame()
for ind in dropout.index:
    studente_drop = pd.DataFrame(dropout.loc[[ind]])
    studente_noDrop = pd.DataFrame(noDropout.loc[[ind]])
    final = pd.concat([studente_drop, final], ignore_index=True, axis = 0)
    final = pd.concat([studente_noDrop, final], ignore_index=True, axis = 0)

features = final
```

Listing 3.1: Primo bilanciamento "a mano"

```
#BILANCIAMENTO 2
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.under_sampling import RandomUnderSampler

#Resampling data
rus = RandomUnderSampler(sampling_strategy = "majority", random_state=42)
X_res, y_res = rus.fit_resample(X, y)
```

Listing 3.2: Secondo bilanciamento tramite RandomUnderSampler

Quindi d'ora in poi mostreremo due tabelle dei risultati:

1. Con il primo bilanciamento
2. Con il secondo bilanciamento

3.6.1 Risultati

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.62	0.73	0.67	0.68	0.63	0.66	0.59	0.60	0.60	437
1	0.64	0.51	0.57	0.63	0.68	0.65	0.56	0.55	0.56	401
Accuracy	0.6252983293556086			0.6539379474940334			0.5763723150357996			838
Macro AVG	0.63	0.62	0.62	0.65	0.65	0.65	0.58	0.58	0.58	838
Weighted AVG	0.63	0.63	0.62	0.66	0.65	0.65	0.58	0.58	0.58	838

Tabella 3.2: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.64	0.71	0.68	0.69	0.63	0.66	0.68	0.60	0.64	455
1	0.61	0.52	0.56	0.60	0.67	0.64	0.58	0.66	0.62	383
Accuracy	0.6276849642004774			0.6479713603818615			0.6276849642004774			838
Macro AVG	0.62	0.62	0.62	0.65	0.65	0.65	0.63	0.63	0.63	838
Weighted AVG	0.63	0.63	0.62	0.65	0.65	0.65	0.63	0.63	0.63	838

Tabella 3.3: P = Precision, R = Recall e F1 = F1-score

3.7 Momento dell'iscrizione bilanciato con ISEE

Ci siamo accorti, che la colonna che presentava più valori NaN e che di conseguenza ci faceva perdere più tuple era la colonna dell'ISEE. La colonna Classe_ISEE presentava tra i vari valori possibili, i valori 0 e 1, dove lo 0 rappresenta l'ISEE non disponibile (coorte del 2016 per il quale questo valore non è stato raccolto) e l'1 rappresentava *ISEE non presentato.* Ci siamo interrogati quindi sulla probabile motivazione legata alla non presentazione dell'ISEE e una delle possibili spiegazioni, oltre ad una semplice dimenticanza, è legata al fatto che l'ISEE sia alto a tal punto da non garantire nessun vantaggio economico in termini di tasse universitarie da pagare. Tramite delle ricerche sui portali

dell'UniBo abbiamo scoperto come oltre i 70.000€ non ci sia più nessuna agevolazione fiscale ma bisogna pagare l'importo massimo. Quindi, oltre ai procedimenti precedenti sono stati sostituiti i valori **NaN** della colonna *Merito_ISEE* in corrispondenza di una colonna *Classe_ISEE* con valore 1, con la media degli ISEE oltre i 70000 euro, in modo tale da cercare di perdere meno tuple possibili dopo aver droppato quelle con almeno un NaN.

```
rich =  features[features['Merito_ISEE'] > 70000]
rich = rich['Merito_ISEE']
mean_rich = rich.mean()
features = features[features['Classe_ISEE'] != 0]
features['Merito_ISEE'] = features['Merito_ISEE'].fillna(mean_rich)
```

3.7.1 Risultati

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.68	0.71	0.70	0.66	0.68	0.67	867
1				0.71	0.67	0.69	0.68	0.65	0.66	883
Accuracy			...	0.6931428571428572			0.666285714285			1750
Macro AVG			...	0.69	0.69	0.69	0.67	0.67	0.67	1750
Weighted AVG			...	0.69	0.69	0.69	0.67	0.67	0.67	1750

Tabella 3.4: P = Precision, R = Recall e F1 = F1-score

In questo e tutti i modelli in cui recuperiamo una maggior quantità di dati andando a riempire i valori NaN con la media degli ISEE oltre i 70.000, non siamo stati in grado di utilizzare il modello SVM su questo dataset per mancanza di potenza di calcolo.

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.67	0.67	0.67	0.57	0.61	0.59	883
1				0.67	0.66	0.66	0.57	0.52	0.54	867
Accuracy			...	0.664			0.5674285714			1750
Macro AVG			...	0.66	0.66	0.66	0.57	0.57	0.57	1750
Weighted AVG			...	0.66	0.66	0.66	0.57	0.57	0.57	1750

Tabella 3.5: P = Precision, R = Recall e F1 = F1-score

3.8 Fine del primo anno

Siamo andati poi a valutare l'impatto che i cfu ottenuti durante tutto l'anno avessero. Abbiamo quindi per ogni studente aggiunto ai dati precedenti due colonne contenenti una la somma totale dei crediti ottenuti alla fine del primo anno di corso e l'altra la media dei voti ottenuti a fine anno di corso.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.90	0.90	0.90	0.91	0.90	0.90	0.93	0.88	0.90	437
1	0.89	0.90	0.89	0.89	0.90	0.90	0.87	0.92	0.90	401
Accuracy	0.8973747016706444			0.9009546539379475			0.8949880668257757			838
Macro AVG	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	838
Weighted AVG	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	0.90	838

Tabella 3.6: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.90	0.91	0.90	0.91	0.90	0.91	0.90	0.90	0.90	455
1	0.89	0.88	0.88	0.88	0.89	0.89	0.88	0.88	0.88	383
Accuracy	0.8949880668257757			0.8973747016706444			0.8914081145584726			838
Macro AVG	0.89	0.89	0.89	0.90	0.90	0.90	0.89	0.89	0.89	838
Weighted AVG	0.89	0.89	0.89	0.90	0.90	0.90	0.89	0.89	0.89	838

Tabella 3.7: P = Precision, R = Recall e F1 = F1-score

3.9 Fine del primo anno con ISEE

Abbiamo poi anche in questo caso, recuperato le tuple con ISEE al preprocessing precedente.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.88	0.92	0.90	0.86	0.89	0.88	867
1				0.92	0.88	0.90	0.89	0.86	0.87	883
Accuracy			...	0.9022857142857142			0.8754285714285714			1750
Macro AVG			...	0.90	0.90	0.90	0.88	0.88	0.88	1750
Weighted AVG			...	0.90	0.90	0.90	0.88	0.88	0.88	1750

Tabella 3.8: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.89	0.88	0.89	0.88	0.86	0.87	883
1				0.88	0.89	0.89	0.86	0.88	0.87	867
Accuracy			...	0.8868571428571429			0.8725714285714286			1750
Macro AVG			...	0.89	0.89	0.89	0.87	0.87	0.87	1750
Weighted AVG			...	0.89	0.89	0.89	0.87	0.87	0.87	1750

Tabella 3.9: P = Precision, R = Recall e F1 = F1-score

3.10 Fine del primo semestre

Successivamente abbiamo voluto valutare l'impatto degli esami ottenuti al primo semestre. Siamo andati quindi ad aggiungere le stesse due colonne aggiunte nel preprocessamento precedente, considerando però solo CFU e voti ottenuti fino al 1/03.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.77	0.84	0.80	0.79	0.80	0.79	0.79	0.83	0.81	437
1	0.81	0.72	0.76	0.78	0.77	0.77	0.80	0.79	0.79	401
Accuracy	0.7863961813842482			0.7852028639618138			0.7923627684964201			838
Macro AVG	0.79	0.78	0.78	0.78	0.78	0.78	0.79	0.79	0.79	838
Weighted AVG	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	0.79	838

Tabella 3.10: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.79	0.83	0.81	0.80	0.79	0.80	0.81	0.81	0.81	455
1	0.78	0.73	0.76	0.76	0.77	0.76	0.77	0.77	0.77	383
Accuracy	0.7852028639618138			0.7828162291169452			0.7911694510739857			838
Macro AVG	0.79	0.78	0.78	0.78	0.78	0.78	0.79	0.79	0.79	838
Weighted AVG	0.79	0.79	0.78	0.78	0.78	0.78	0.79	0.79	0.79	838

Tabella 3.11: P = Precision, R = Recall e F1 = F1-score

3.11 Fine del primo semestre con ISEE

Vediamo a questo punto l'impatto che avremmo aggiungendo anche in questo caso le tuple con l'isee.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.79	0.83	0.81	0.75	0.85	0.79	867
1				0.82	0.78	0.80	0.83	0.72	0.77	883
Accuracy			...	0.8062857142857143			0.7828571428571428			1750
Macro AVG			...	0.81	0.81	0.81	0.78	0.78	0.78	1750
Weighted AVG			...	0.81	0.81	0.81	0.78	0.78	0.78	1750

Tabella 3.12: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0				0.79	0.80	0.79	0.75	0.82	0.78	883
1				0.79	0.78	0.78	0.80	0.72	0.76	867
Accuracy			...	0.788			0.7725714285714286			1750
Macro AVG			...	0.79	0.79	0.79	0.78	0.77	0.77	1750
Weighted AVG			...	0.79	0.79	0.79	0.77	0.77	0.77	1750

Tabella 3.13: P = Precision, R = Recall e F1 = F1-score

3.12 Extra

Come si può notare la differenza tra i risultati delle due versioni dei bilanciamenti è molto lieve. In alcuni dei nostre primi risultati, avevamo notato come ci fosse una sostanziale differenza tra queste due diverse tecniche di bilanciamento, con il nostro bilanciamento che, nel caso in cui si consideravano tutti gli esami arrivava ad accuratezze del 95% circa. Questo era dettato dal fatto che utilizzavamo nel nostro modello la colonna *IDStudente* che, nel nostro bilanciamento, venendo selezionata in tuple contigue influenzava molto di più il modello rispetto al bilanciamento tramite *RandomUnderSampler*. Questo spiega il perchè droppiamo la colonna IDStudente ed il perchè abbiamo deciso di mostrare comunque i risultati di entrambi i bilanciamenti per quanto simili.

Evaluating algorithms (Primo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.91	0.99	0.95	0.92	0.99	0.96	0.85	0.95	0.90	437
1	0.99	0.90	0.94	0.99	0.91	0.95	0.94	0.81	0.87	401
Accuracy	0.9463007159904535			0.9522673031026253			0.8866348448687351			838
Macro AVG	0.95	0.94	0.94	0.96	0.95	0.95	0.89	0.88	0.89	838
Weighted AVG	0.95	0.95	0.95	0.95	0.95	0.95	0.89	0.89	0.89	838

Tabella 3.14: P = Precision, R = Recall e F1 = F1-score

Evaluating algorithms (Secondo Bilanciamento)

	SVM			RF			LR			
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**	**Support**
0	0.65	0.60	0.63	0.71	0.63	0.66	0.62	0.56	0.59	455
1	0.57	0.62	0.59	0.61	0.69	0.65	0.53	0.58	0.55	383
Accuracy	0.609785202863918			0.6551312649164678			0.5715990453460621			838
Macro AVG	0.61	0.61	0.61	0.66	0.66	0.65	0.57	0.57	0.57	838
Weighted AVG	0.61	0.61	0.61	0.66	0.66	0.66	0.58	0.57	0.57	838

Tabella 3.15: P = Precision, R = Recall e F1 = F1-score

3.12.1 Stacking Algorithm

Abbiamo applicato lo Stacking Algorithm ai tre momenti con il secondo bilanciamento e senza la sostituzione dei NaN degli ISEE (altrimenti non avremmo potuto inserire SVM), definendolo in questo modo:

```
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, StackingClassifier

def get_stacking():
  # define the base models
  level0 = list()
  level0.append(('rf', RandomForestClassifier(n_estimators=207,
    random_state=42)))
  level0.append(('svm', SVC(C=0.01, kernel='linear')))
  level0.append(('lr', LogisticRegression()))
  # define meta learner model
  level1 = LogisticRegression()
  # define the stacking ensemble
  model = StackingClassifier(estimators=level0, final_estimator=level1, cv
    =5)
  return model
```

Listing 3.3: Stacking definition

Come si può vedere dal codice 3.3 abbiamo applicato al:

- **Livello 0:** Random Forest, SVM e Logistic Regression (i nostri tre modelli)
- **Livello 1:** Logistic Regression (di default)

Momento dell'Iscrizione Bilanciato

Evaluating algorithm (Secondo Bilanciamento)

	Precision	Recall	F1-score	Support
0	0.69	0.66	0.67	455
1	0.62	0.65	0.63	383
Accuracy	0.6551312649164678			838
Macro AVG	0.65	0.65	0.65	838
Weighted AVG	0.66	0.66	0.66	838

Fine del primo anno

Evaluating algorithm (Secondo Bilanciamento)

	Precision	Recall	F1-score	Support
0	0.91	0.89	0.90	455
1	0.88	0.89	0.88	383
Accuracy	0.8926014319809069			838
Macro AVG	0.89	0.89	0.89	838
Weighted AVG	0.89	0.89	0.89	838

Fine del primo semestre

Evaluating algorithm (Secondo Bilanciamento)

	Precision	Recall	F1-score	Support
0	0.80	0.80	0.80	455
1	0.76	0.77	0.76	383
Accuracy	0.7828162291169452			838
Macro AVG	0.78	0.78	0.78	838
Weighted AVG	0.78	0.78	0.78	838

3.13 Studio di comparazione

Il nostro riferimento principale per comparare i risultati è il lavoro di Del Bonifro et Al. [2]. Confrontando il modello che ha i risultati migliori, quindi il nostro modello alla fine del primo anno con il modello *Basic + ALR + CC* di [2] possiamo notare come otteniamo un'accuracy migliore di circa 2 punti percentuali. Anche per quanto riguarda la Recall otteniamo un miglioramento rispettamente di 0.01 per il modello SVM e di 0.04 per il modello RF.

Capitolo 4

Discussione e conclusioni

4.1 Discussione dei risultati

I risultati ottenuti ci mostrano come sia possibile predire l'abbandono di uno studente con affidabilità diverse in momenti diversi del primo anno di corso. Come prevedibile, è più facile predire l'abbandono di uno studente al passare del tempo, proprio per questo otteniamo risultati migliori nella predizione a fine del primo anno, risultati che peggiorano se consideriamo la fine del primo semestre e risultati ancora meno precisi al momento dell'iscrizione. Tra il modello predittivo a fine del primo anno e quello a fine primo semestre c'è una differenza di accuracy di circa l'11% con circa la stessa differenza anche nella recall dei dropout. La differenza in accuratezza tra il modello al momento dell'iscrizione ed il modello alla fine del primo semestre è di circa il 15% un valore non indifferente, mentre la recall varia di circa 10 punti percentuali. Un'altro risultato che possiamo notare è che non in tutti i casi un utilizzo di un maggior numero di dati (recuperando tuple riempiendo i NaN dell'ISEE come spiegato in precedenza) porta all'ottenimento di risultati migliori, poichè come possiamo vedere nel modello a fine del primo anno, con l'utilizzo del secondo bilanciamento i risultati peggiorano invece che migliorare. Possiamo quindi dire che il modello rispetta le attese, poichè è triviale capire che più dati relativi all'andamento scolastico dello studente (nel nostro caso il numero di CFU superati e il superamento o meno degli OFA) possa chiarire quello che è la possibile scelta di uno studente relativamente all'abbandono scolastico.
Infine utilizzando l'approccio dell'Ensamble Learning non abbiamo ottenuto un miglioramento considerevole delle performance. Specifichiamo però che questo è stato un primo approccio a questa metodologia e il miglioramento di esso lo lasciamo come sviluppo futuro.

4.2 Limiti del metodo

Sicuramente un limite del nostro modello è la perdita di dati. Dover bilanciare un dataset fortemente sbilanciato come quello utilizzato in questo lavoro porta ad una grande perdita di dati e di conseguenza informazioni che potevano migliorare/peggiorare i risultati ottenuti. Un'altro importante limite del nostro modello è che si ottengono i risultati migliori alla fine del primo anno di corso, quando ormai, potrebbe essere troppo tardi per aiutare lo studente.

4.3 Lavori futuri

In questo lavoro abbiamo applicato un primo approccio ad una tecninca di Ensamble Learning, quale lo Stacking Algorithm, e lasciamo sicuramente come sviluppo futuro la rifinitura dell'approccio utilizzato che porterà sicuramente al miglioramento dei risultati ottenuti.
Per continuare questo lavoro sicuramente si potrebbe andare a rendere più specifico il modello rispetto ad alcune caratteristiche come il corso di studi o la scuola di appartenenza, per andare a valutare con più precisione il tasso e le motivazioni di abbandono legate ad ogni singolo corso di studio o ambito di studio. Anche l'utilizzo di nuove tecniche come l'utilizzo delle NB o del kNN come in [1] è un possibile sviluppo futuro. Un'altro possibile sviluppo futuro riguarda l'integrazione di alcuni dati che non vengono presi in considerazione in questo studio, come la situazione familiare o l'eventuale condizione di studente-lavoratore, che possono sicuramente incidere sul percorso di studi di un ragazzo.

Bibliografia

[1] Agrusti, F., Mezzini, M., and Bonavolontà, G. Deep learning approach for predicting university dropout: A case study at Roma Tre University. *Journal of e-Learning and Knowledge Society 16*, 1 (2020), 44–54.

[2] Bonifro, F. D., Gabbrielli, M., Lisanti, G., and Zingaro, S. P. Artificial Intelligence in Education, 21st International Conference, AIED 2020, Ifrane, Morocco, July 6–10, 2020, Proceedings, Part I. 129–140.

[3] Gholamy, A., Kreinovich, V., and Kosheleva, O. Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation.

[4] Kadar, M., Sarraipa, J., Guevara, J. C., and Restrepo, E. G. y. An integrated approach for fighting dropout and enhancing students' satisfaction in higher education. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion* (New York, NY, USA, 2018), DSAI 2018, Association for Computing Machinery, p. 240–247.

[5] Zingaro, S., Del Zozzo, A., Del Bonifro, F., and Gabbrielli, M. Predictive models for effective policy making against university dropout. *Form@re - Open Journal per la formazione in rete 20*, 3 (Dec. 2020), 165–175.