Dipartimento di Informatica

Corso di laurea in Informatica

Caso di Studio per l'esame di Ingegneria della Conoscenza

Dropout Detective

Sistema intelligente per la previsione di abbandono o conclusione accademica di uno studente

Progetto di:
Natale Mastrogiacomo (758384) n.mastrogiacomo4@studenti.uniba.it

Link Repository:
https://github.com/GameRule17/DropoutDetective

Anno Accademico **2023-2024**

Indice

1 Introduzione

Garantire l'accesso universale a un'istruzione di qualità è una priorità chiave a livello globale. Tuttavia, secondo il rapporto indipendente per la Commissione Europea, l'abbandono scolastico rimane un problema molto presente a causa di una varietà di fattori sociali, economici e demografici. Anche nel paese più di successo (Danimarca), solo circa l'80% degli studenti completa i propri studi, mentre in Italia questa percentuale è solo del 46% [1]. Il progetto, date una serie di informazioni su uno studente estratte da [2], ha come obiettivo principale quello di predire il suo potenziale abbandono agli studi o, alternativamente, una futura conclusione con successo degli stessi.

1.1 Elenco argomenti di interesse

- **Programmazione Logica** [3]: Rappresentazione Relazionale, Linguaggio Relazionale a Regole, Query con Variabili.
- **Apprendimento Supervisionato** [4]: Valutare le Predizioni, Alberi Decisionali, Random Forest, Regressione Logistica, Cross Validation, *Oversampling, Ensemble Model con VotingClassifier, valutare le predizioni in presenza di sbilanciamento delle classi.*
- **Reti Neurali e Deep Learning** [5]: Feedforward Neural Networks, Multilayer Perceptron.
- **Apprendimento Probabilistico** [6]: Rete Bayesiana, Apprendimento della Struttura, Query, Generazione di nuovi Samples, *Valutazione.*

In *corsivo* gli argomenti extra trattati.

2 Preprocessing dei dati

Il dataset presenta le seguenti feature (in rosso la feature target):

Feature	**Descrizione**
Marital status	Lo stato civile dello studente. (Categorico)
Application mode	Il metodo di candidatura utilizzato dallo studente.
Application order	Momento in cui lo studente ha fatto domanda.
Course	Il corso seguito dallo studente.
Daytime/evening attendance	Se lo studente frequenta le lezioni di giorno o di sera.
Previous qualification	La qualifica ottenuta dallo studente prima di iscriversi all'università.
Nacionality	La nazionalità dello studente.
Mother's qualification	La qualifica della madre dello studente.
Father's qualification	La qualifica del padre dello studente.
Mother's occupation	L'occupazione della madre dello studente.
Father's occupation	L'occupazione del padre dello studente.
Displaced	Se lo studente è uno sfollato.
Educational special needs	Se lo studente ha bisogni educativi speciali.
Debtor	Se lo studente ha dei debiti non ancora pagati.
Tuition fees up to date	Se lo studente è in regola con il pagamento delle tasse universitarie.
Gender	Il sesso dello studente.
Scholarship holder	Se lo studente è un borsista.
Age at enrollment	L'età dello studente al momento dell'iscrizione.
International	Se lo studente è uno studente internazionale.
Curricular units 1st sem (credited)	Il numero di unità curriculari che lo studente ha completato e per le quali ha ricevuto crediti accademici nel primo semestre.
Curricular units 1st sem (enrolled)	Il numero di unità curriculari alle quali lo studente è formalmente iscritto nel primo semestre.
Curricular units 1st sem (evaluations)	Il numero di unità curriculari per le quali lo studente ha ricevuto una valutazione o un voto nel primo semestre.
Curricular units 1st sem (approved)	Il numero di unità curriculari che lo studente ha superato con successo nel primo semestre. Significa che lo studente ha ottenuto un voto sufficiente per passare queste unità.

Tabella 1: Descrizione delle colonne del dataset degli studenti (Parte 1).

Feature	Descrizione
Curricular units 2nd sem (credited)	Il numero di unità curriculari che lo studente ha completato e per le quali ha ricevuto crediti accademici nel secondo semestre.
Curricular units 2nd sem (enrolled)	Il numero di unità curriculari alle quali lo studente è formalmente iscritto nel secondo semestre.
Curricular units 2nd sem (evaluations)	Il numero di unità curriculari per le quali lo studente ha ricevuto una valutazione o un voto nel secondo semestre.
Curricular units 2nd sem (approved)	Il numero di unità curriculari che lo studente ha superato con successo nel secondo semestre. Significa che lo studente ha ottenuto un voto sufficiente per passare queste unità.
Target	La variabile dipendente per la predizione dell'abbandono o successo accademico. (Categorico)

Tabella 2: Descrizione delle colonne del dataset degli studenti (Parte 2).

Il dataset è stato inizialmente preprocessato come segue, ulteriori operazioni di processing dei dati sono state eseguite a termine della fase di feature engineering. Sono quindi state eseguite le seguenti operazioni per garantire la qualità e la coerenza dei dati:

1. **Rinominazione di alcune colonne con nomi errati o troppo esplicativi**: Alcune colonne nel dataset presentavano nomi con errori ortografici o nomi troppo descrittivi che rendevano il codice meno leggibile. Per risolvere questo problema, le colonne sono state rinominate, ad esempio, da *Nacionality* a *Nationality* e da *Age at enrollment* a *Age*.

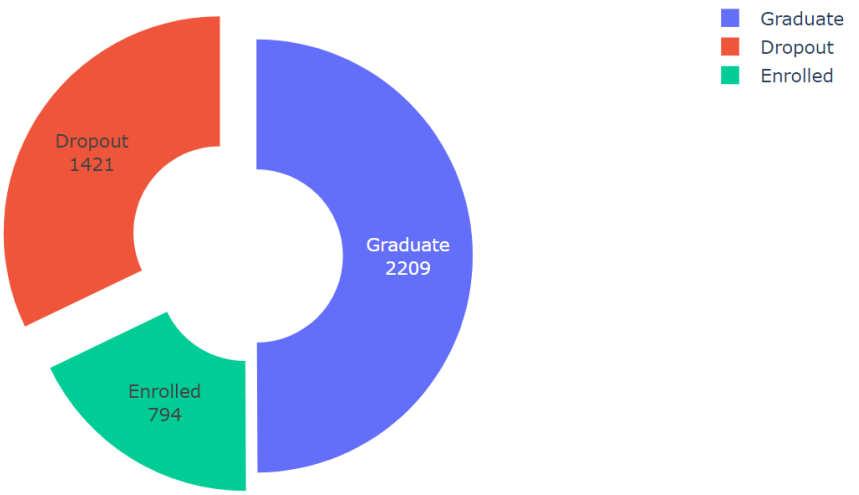
2. **Rimozione dai nomi delle colonne del genitivo sassone**: Per semplificare i nomi delle colonne, è stato rimosso il genitivo sassone (i.e., "'s"). Questo è stato fatto anche per agevolare le operazioni di feature engineering che saranno successivamente descritte.

3. **Verifica della presenza di valori nulli**: È stata effettuata una verifica per identificare la presenza di valori nulli nel dataset. La presenza di questi avrebbe potuto causare problemi nelle operazioni successive.

4. **Verifica della presenza di valori duplicati**: È stata effettuata una verifica per identificare la presenza di valori duplicati nel dataset. La presenza di avrebbe potuto distorcere i risultati.

5. **Codifica della feature Target**: La colonna *Target* è l'unico campo non numerico nel dataset e contiene tre valori distinti. Per facilitare l'analisi, i valori sono stati codificati utilizzando un dizionario, mappando ciascun valore a un numero intero: *Dropout* a 0, *Enrolled* a 1, e *Graduate* a 2.

A questo punto, dando un'occhiata ai dati, possiamo notare un netto sbilanciamento delle istanze della feature Target, in particolare quella con valore *Enrolled*.



Pertanto, quello che stiamo per affrontare è un problema di classificazione multiclasse sbilanciato.

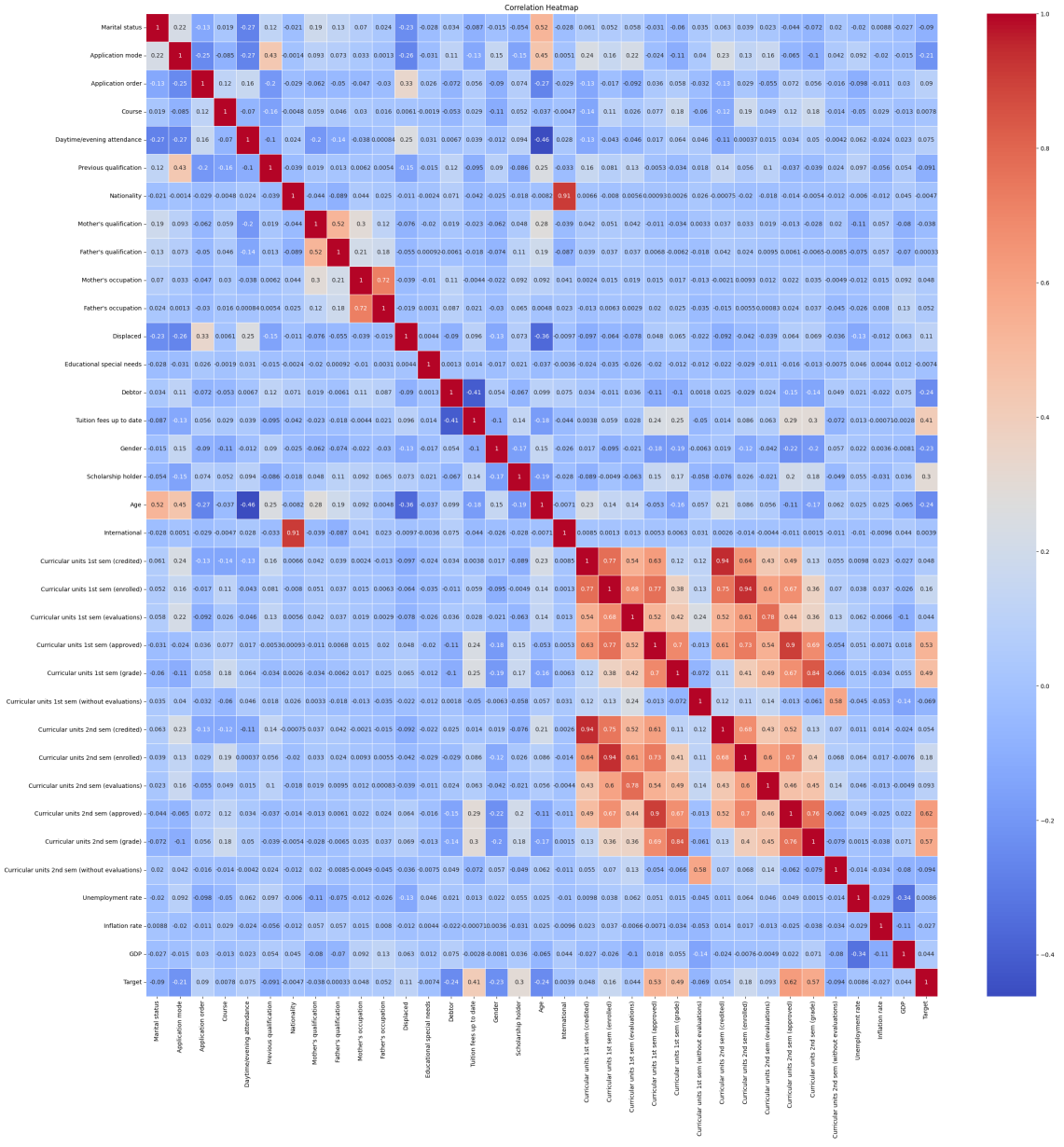
3 Programmazione Logica

3.1 Analisi delle Feature

Abbiamo inizialmente estratto una heatmap di correlazione con l'obiettivo di visualizzare le correlazioni tra diverse variabili del dataset. Ricordiamo che i coefficienti di correlazione variano da -1 a 1:

1. 1 indica una correlazione positiva perfetta.
2. -1 indica una correlazione negativa perfetta.
3. 0 indica nessuna correlazione.

I colori della heatmap variano dal blu (correlazione negativa) al rosso (correlazione positiva), passando per il bianco (nessuna correlazione). Abbiamo quindi scoperto come le feature con alta corrispondenza sono quelle rappresentanti le varie unità curriculari, fra loro combinate. Ad esempio *Curricular units 1st sem (credited)* ha una correlazione molto positiva con *Curricular units 2nd sem (credited)* pari a *0.94*



In generale, a seguito dell'analisi della heatmap sono state scoperte le seguenti correlazioni fra le feature del dataset:

1. Prestazioni Accademiche: Le prestazioni accademiche degli studenti (misurate tramite approvazioni e voti) sono strettamente correlate tra i semestri. Migliorare le performance in un semestre potrebbe avere effetti positivi anche sul semestre successivo.

2. Regolarità nei Pagamenti: Gli studenti in regola con i pagamenti delle tasse universitarie tendono ad avere un migliore stato finanziario e meno debiti, il che può influenzare positivamente il loro successo accademico.

3. Impatto delle Variabili Demografiche: Le variabili demografiche hanno un impatto minore sulle prestazioni accademiche rispetto alle variabili che misurano direttamente l'interazione con il curriculum scolastico.

3.2 Feature Engineering via Query Prolog

Sulla base delle informazioni ottenute in sezione 3.1, si è deciso di implementare una base di conoscenza con fatti prelevati dal dataset in modo da, attraverso l'uso di linguaggio relazionale a regole (nel nostro caso Prolog), estrarre nuove feature più esplicative. Questo approccio consente di esprimere facilmente relazioni complesse e regole di inferenza, migliorando la qualità e la profondità delle analisi sui dati degli studenti.
Si è scelto principalmente di utilizzare la libreria *pyswip*, la quale consente di chiamare query Prolog tramite Python. La stessa consente infatti di creare una nuova istanza di Prolog per eseguire operazioni di inferenza logica sui dati. Successivamente è stata definita una funzione *facts_from_dataframe* che converte ogni riga del dataframe in un fatto Prolog. Ogni fatto rappresenta uno studente e le sue caratteristiche, rendendo i dati strutturati in modo relazionale per essere utilizzati da Prolog.

In seguito un'altra funzione *new_features_extraction* utilizza regole logiche in Prolog per creare nuove feature. Le nuove regole definite sono frutto dell'analisi effettuata in Sezione 3.1 e includono:

- **Studenti Finanziariamente Stabili**: Una regola che identifica gli studenti che non hanno debiti e sono in regola con il pagamento delle tasse universitarie.

```
financially_stable(ID) :-
    student(ID, Debtor, FeesUpToDate),
    (Debtor =:= 0, FeesUpToDate =:= 1).
```

- **Interazione tra le unità curricolari del primo e secondo semestre approvate**: Una regola che calcola l'interazione tra il numero di unità curricolari approvate nel primo e nel secondo semestre.

```
interaction_cu_1st_2nd_approved(ID, Result) :-
    student(ID, CU1stApproved, CU2ndApproved),
    Result is CU1stApproved * CU2ndApproved.
```

- **Interazione tra le unità curricolari del primo e secondo semestre valutate**: Una regola che calcola l'interazione tra i voti delle unità curricolari del primo e del secondo semestre.

```
interaction_cu_1st_2nd_grade(ID, Result) :-
    student(ID, CU1stGrade, CU2ndGrade),
    Result is CU1stGrade * CU2ndGrade.
```

- **Totale delle unità curricolari approvate**: Una regola che somma il numero totale di unità curricolari approvate nei due semestri.

```
total_cu_approved(ID, Result) :-
    student(ID, CU1stApproved, CU2ndApproved),
    Result is CU1stApproved + CU2ndApproved.
```

- **Media dei voti delle unità curricolari**: Una regola che calcola la media dei voti delle unità curricolari dei due semestri.

```
total_cu_grade(ID, Result) :-
    student(ID, CU1stGrade, CU2ndGrade),
    Result is (CU1stGrade + CU2ndGrade) / 2.
```

Al termine della definizione, le regole definite Prolog sono state eseguite per ottenere i nuovi attributi per ogni studente. Le query con variabili sono state utilizzate per estrarre i risultati delle regole logiche applicate. I risultati delle query sono stati convertiti in un formato più utile e inseriti nel dataframe originale come nuove colonne. Questo ha permesso di arricchire il dataset con nuove informazioni derivate dalle regole logiche applicate.
A seguito dell'esecuzione delle operazioni di Feature Engineering sono quindi inserite le seguenti feature nel Dataset:

Feature
Financially Stable
Interaction_CU_1st_2nd_Approved
Interaction_CU_1st_2nd_Grade
Total_CU_Approved
Total_CU_Grade

Tabella 3: Nuove Feature nel Dataset.

3.3 Correlazioni con Feature Target

Effettuiamo una analisi dell'importanza delle nuove feature nel dataset nei confronti del Target in modo da determinare quali colonne rimuovere. Generalmente, si sceglierà di rimuovere quelle con bassa correlazione con la variabile *Target*. Più specificatamente, abbiamo impostato una soglia di correlazione e rimosso le colonne con coefficienti di correlazione al di sotto di tale soglia. In questo caso, si è scelto di rimuovere le colonne con un coefficiente di correlazione assoluto inferiore a 0.1 (ovviamente questa soglia può essere regolata in base alle necessità). Di seguito, alcune colonne che abbiamo considerato per la rimozione, rappresentate anche in Tabella 4 di colore rosso:

- **Father's occupation**: Con correlazione **0.051702**.
- **Mother's occupation**: Con correlazione **0.048424**.
- **Curricular units 1st sem (credited)**: Con correlazione **0.048150**.

- **Curricular units 1st sem (evaluations)**: Con correlazione **0.044362**.
- **GDP**: Con correlazione **0.044135**.
- **Unemployment rate**: Con correlazione **0.008627**.
- **Course**: Con correlazione **0.007841**.
- **International**: Con correlazione **0.003934**.
- **Father's qualification**: Con correlazione **0.000329**.
- **Nationality**: Con correlazione **-0.004740**.
- **Educational special needs**: Con correlazione **-0.007353**.
- **Inflation rate**: Con correlazione **-0.026874**.
- **Mother's qualification**: Con correlazione **-0.038346**.

Queste colonne hanno valori di correlazione assoluta bassi e potrebbero non fornire un potere predittivo significativo per la tua variabile *Target*. Tuttavia, saranno poi eseguiti degli esperimenti sia con che senza queste colonne per vedere se si ottiene una differenza significativa nelle prestazioni dei modelli.

Sono inoltre state rimosse anche le colonne utilizzate per la realizzazione di feature più esplicative, come definito nella Sezione 3.2, riducendo la multicollinearità. Esse sono rappresentate in Tabella 4 con colore giallo:

- Curricular units 2nd sem (approved).
- Curricular units 2nd sem (grade).
- Curricular units 1st sem (approved).
- Curricular units 1st sem (grade).
- Tuition fees up to date.
- Debtor.

Feature	**Correlazione**
Target	1.000000
Curricular units 2nd sem (approved)	0.624157
Interaction_CU_1st_2nd_Grade	0.590804
Total_CU_Approved	0.590362
Curricular units 2nd sem (grade)	0.566827
Total_CU_Grade	0.550356
Curricular units 1st sem (approved)	0.529123
Curricular units 1st sem (grade)	0.485207
Tuition fees up to date	0.409827
Interaction_CU_1st_2nd_Approved	0.409278
Financially Stable	0.382936
Scholarship holder	0.297595
Curricular units 2nd sem (enrolled)	0.175847
Curricular units 1st sem (enrolled)	0.155974
Displaced	0.113986
Curricular units 2nd sem (evaluations)	0.092721
Application order	0.089791
Daytime/evening attendance	0.075107
Curricular units 2nd sem (credited)	0.054004
Father occupation	0.051702
Mother occupation	0.048424
Curricular units 1st sem (credited)	0.048150
Curricular units 1st sem (evaluations)	0.044362
GDP	0.044135
Unemployment rate	0.008627
Course	0.007841
International	0.003934
Father qualification	0.000329
Nationality	-0.004740
Educational special needs	-0.007353
Inflation rate	-0.026874
Mother's qualification	-0.038346
Curricular units 1st sem (without evaluations)	-0.068702
Marital status	-0.089804
Previous qualification	-0.091365
Curricular units 2nd sem (without evaluations)	-0.094028
Application mode	-0.212025
Gender	-0.229270
Debtor	-0.240999
Age	-0.243438

Tabella 4: Correlazioni delle feature con Target.

Per concludere la sezione, riportiamo in Tabella 5 la lista delle nuove feature del dataset a seguito dell'esecuzione di tutte le operazioni sopra citate.

Feature
Interaction_CU_1st_2nd_Grade
Total_CU_Approved
Total_CU_Grade
Interaction_CU_1st_2nd_Approved
Financially Stable
Scholarship holder
Curricular units 2nd sem (enrolled)
Curricular units 1st sem (enrolled)
Displaced
Curricular units 2nd sem (evaluations)
Application order
Daytime/evening attendance
Curricular units 2nd sem (credited)
Curricular units 1st sem (without evaluations)
Marital status
Previous qualification
Curricular units 2nd sem (without evaluations)
Application mode
Gender
Age
Target

Tabella 5: Nuove feature del Dataset.

4 Apprendimento Supervisionato

4.1 Modelli e metodologie utilizzate

Modelli adottati. Per qusto progetto si è deciso di adottare principalmente tre modelli di apprendimento automatico, più un quarto solo in uno specifico esperimento. Tutti sono stati presentati e studiati durante il corso:

- **Albero Decisionale**: classificatore strutturato ad albero in cui le foglie rappresentano le classi di appartenenza (o le probabilità di appartenenza a tali classi) mentre la radice e i nodi interni rappresentano delle condizioni sulle feature di input. A seconda se tali condizioni sono rispettate o meno, verrà seguito un percorso piuttosto che un altro e alla fine si arriverà alla classe di appartenenza.

- **Random Forest**: classificatore che si ottiene creando tanti Decision Tree. Il valore in output si ottiene mediando sulle predizioni di ogni albero. appartenente alla foresta (tecnica di bagging)

- **Regressione Logistica**: modello che stima la probabilità che una determinata osservazione appartenga alla classe positiva utilizzando una funzione logistica della combinazione lineare delle variabili indipendenti. È un metodo efficace e ampiamente utilizzato per problemi di classificazione.

- **Rete Neurale Multilayer Perceptron**: modello costituito da strati multipli di nodi in un grafo diretto, con ogni strato completamente connesso al successivo. Eccetto che per i nodi in ingresso, ogni nodo è un neurone (elemento elaborante) con una funzione di attivazione lineare.

Train Test Split. Per valutare le performance dei modelli, si è diviso il dataset in due parti: una parte per l'addestramento e una parte per la validazione. Si è scelto di adottare una divisione 80-20 **stratificata**, in modo da mantenere la stessa distribuzione delle classi nel training set e nel test set. La suddivisione stratificata infatti assicura che entrambi i set contengano una proporzione simile di campioni per ciascuna classe presente nel dataset originale. Questo è particolarmente importante quando si affrontano problemi di classificazione, come nel nostro caso, in cui le classi non sono bilanciate nel dataset originale. La stratificazione aiuta a garantire che il modello venga addestrato su una varietà di esempi rappresentativi di ciascuna classe e che le prestazioni del modello siano valutate su un test set che riflette accuratamente la distribuzione delle classi nel mondo reale.

Cross-Validation. Per trovare i migliori iperparametri per ogni modello e per la fase di valutazione si è adottata una *Repeated Stratified K-fold cross validation*, in cui la cross validation viene ripetuta per m volte mantenendo nei fold la distribuzione originaria delle classi. Il numero K di fold è stato definito per ogni esperimento dalla regola di Sturges, una delle regole empiriche per la scelta del numero di classi in un istogramma. La regola di Sturges suggerisce di utilizzare un numero di classi (*fold*) pari a $1 + 3.322 \cdot \log_{10}(n)$, dove n è il numero di osservazioni **del training set**.

Ricerca degli iperparametri. Per la ricerca degli iperparametri migliori si è adottata una *GridSearchCV* con una *Repeated Stratified K-fold Cross-Validation*. Di seguito la descrizione degli iperparametri per ogni modello e i valori testati:

DecisionTree

- **criterion**: Misura la qualità dello split effettuato sui nodi. **Range di valori:** ["gini", "entropy", "log_loss"].
  Descrizioni:
  - **gini**: Misura la "purezza" della divisione dei dati, più precisamente quanto spesso un elemento viene classificato in modo sbagliato
  - **entropy**: Misura la quantità di disordine nei dati. Minimizzare l'entropia significa massimizzare l'information gain
  - **log_loss**: Logarithmic loss, indicata quando l'output corrisponde a una probabilità piuttosto che a un valore di classe.
- **max_depth**: Massima profondità dell'albero. **Range di valori:** [5, 10, 20, 40].
- **min_samples_split**: Numero minimo di campioni per suddividere un nodo. **Range di valori:** [2, 5, 10, 20].
- **min_samples_leaf**: Numero minimo di campioni per essere foglia. **Range di valori:** [2, 5, 10, 20].
- **splitter**: Strategia di suddivisione del nodo. **Range di valori:** ["best"].

RandomForest

- **criterion**: Criterio di suddivisione. **Range di valori:** ["gini", "entropy", "log_loss"].
  Descrizioni:
  - **gini**: Misura la "purezza" della divisione dei dati, più precisamente quanto spesso un elemento viene classificato in modo sbagliato
  - **entropy**: Misura la quantità di disordine nei dati. Minimizzare l'entropia significa massimizzare l'information gain
  - **log_loss**: Logarithmic loss, indicata quando l'output corrisponde a una probabilità piuttosto che a un valore di classe.
- **n_estimators**: Numero di alberi nella foresta. **Range di valori:** [10, 100, 200].
- **max_depth**: Massima profondità dell'albero. **Range di valori:** [5, 10, 20].
- **min_samples_split**: Il numero minimo di esempi necessari affinché possa essere utilizzato un criterio di split. **Range di valori:** [2, 5, 10].
- **min_samples_leaf**: Il numero minimo di esempi per poter creare una foglia. **Range di valori:** [2, 5, 10].

LogisticRegression

- **penalty**: Tipo di norma per la regolarizzazione. **Range di valori:** ["l1", "l2", "elasticnet", "none"].
  Descrizioni:
  - **l1**: Regolarizzazione L1 (lasso), favorisce i modelli sparsi.
  - **l2**: Regolarizzazione L2 (ridge), limita i valori assoluti dei coefficienti.
  - **elasticnet**: Combinazione di regolarizzazione L1 e L2, con parametro di mixing α per il peso tra le due regolarizzazioni.
  - **none**: Nessuna regolarizzazione.
- **C**: Inverso della forza di regolarizzazione. Valori più alti indicano una regolarizzazione più debole. **Range di valori:** [0.1, 1, 10].
- **solver**: Algoritmo utilizzato per l'ottimizzazione della funzione obiettivo. **Range di valori:** ["newton-cg", "lbfgs", "liblinear", "sag", "saga"].
  Descrizioni:
  - **newton-cg**: Metodo Newton-Conjugate Gradient.
  - **lbfgs**: Limited-memory Broyden–Fletcher–Goldfarb–Shanno.
  - **liblinear**: Liblinear, ottimizzazione con line search.
  - **sag**: Stochastic Average Gradient.
  - **saga**: Successive Approximation of Generalized Gradient.

MLPClassifier

- **hidden_layer_sizes**: L'i-esimo elemento rappresenta il numero di neuroni nell'i-esimo strato nascosto. **Intervallo di valori:** [(10,10,10), (20,50,20), (50,)].
- **activation**: Funzione di attivazione per lo strato nascosto. **Intervallo di valori:** ['tanh', 'relu'].
  - **tanh**: La funzione tangente iperbolica, restituisce f(x) = tanh(x).
  - **relu**: La funzione unità rettificata lineare, restituisce f(x) = max(0, x).
- **solver**: Ottimizzazione dei pesi. **Intervallo di valori:** ['sgd', 'adam'].
  - **sgd**: Discesa del gradiente stocastica.
  - **adam**: Il risolutore Adam, un ottimizzatore basato sul gradiente stocastico.
- **alpha**: Parametro di penalità L2 (termine di regolarizzazione). **Intervallo di valori:** [0.0001, 0.01].
- **learning_rate**: Velocità del passo nella discesa del gradiente. **Intervallo di valori:** ['constant', 'adaptive'].
  - **constant**: Il tasso di apprendimento è costante come definito da 'learning_rate_init'.
  - **adaptive**: Il tasso di apprendimento viene ridotto quando l'apprendimento si stabilizza.

Riproducibilità. Per garantire la riproducibilità dei risultati, si è scelto di fissare il *seed* per la generazione di numeri casuali al numero *42*.

Valutazione. La parte di valutazione è stata fatta utilizzando una *cross_val_score*, ovvero un metodo che valuta sempre utilizzando la Cross Validation i modelli addestrati. Per la valutazione delle performance del sistema ho deciso di utilizzare le seguenti metriche:

- **Accuracy**: La percentuale di previsioni corrette su tutte le previsioni effettuate. È una misura semplice che calcola la proporzione di tutte le istanze correttamente classificate rispetto al totale delle istanze. Tuttavia, può essere fuorviante in presenza di un forte sbilanciamento tra le classi, poiché un classificatore che predice sempre la classe più frequente può avere un'accuracy alta ma scarsa capacità di riconoscere le classi meno frequenti.

- **F1 Score**: La media armonica tra precision e recall. Fornisce un bilanciamento tra le due metriche, utile quando si desidera un compromesso tra precisione e capacità di recupero delle istanze positive. L'F1 score è particolarmente utile in presenza di classi sbilanciate, poiché combina sia la precision che il recall in un'unica metrica. È definito come:
$$\text{F1 Score} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- **Precision**: La proporzione di istanze correttamente identificate come positive rispetto al totale delle istanze identificate come positive. In altre parole, tra tutte le istanze che il modello prevede come positive, quante sono realmente positive. È particolarmente utile quando il costo di un falso positivo è alto. È definita come:
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall**: La proporzione di istanze correttamente identificate come positive rispetto al totale delle istanze effettivamente positive. In altre parole, tra tutte le istanze che sono realmente positive, quante sono state correttamente identificate dal modello. È particolarmente utile quando il costo di un falso negativo è alto. È definita come:
$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **Balanced Accuracy**: È una versione modificata dell'accuratezza standard che tiene conto dello sbilanciamento delle classi nei dati. Calcola la media delle accuratezze per ciascuna classe, fornendo quindi una valutazione equilibrata delle prestazioni del modello su tutte le classi. Utilizzare un'accuracy normale sarebbe fuorviante, al classificatore potrebbe bastare predire sempre la classe più frequente per ottenere un'accuracy alta, ma questo non significa che il classificatore sia buono. La balanced accuracy è definita come la media delle sensibilità per ciascuna classe.

La *GridSearchCV* è stata allenata nel trovare gli iperparametri che massimizzano la *Balanced accuracy*. Sono state poi prese in considerazione **varianza** e **deviazione standard** per ogni modello: entrambe queste metriche misurano la dispersione dei dati. In particolare, la varianza indica quanto le predizioni del modello si discostano dalla media delle predizioni stesse, mentre la deviazione standard è la radice quadrata della varianza e fornisce una misura della dispersione delle predizioni rispetto alla media.

Un modello con una varianza e una deviazione standard più basse è preferibile perché indica che le predizioni del modello sono più coerenti e meno sparse. In altre parole, il modello è più stabile e le sue prestazioni sono meno soggette a variazioni casuali. Questo significa che le sue predizioni sono affidabili e che il modello è meno incline a overfitting, ovvero a catturare rumore nei dati di addestramento piuttosto che le caratteristiche generali dei dati. Un modello con bassa varianza e deviazione standard è quindi considerato più robusto e performante, capace di generalizzare meglio su nuovi dati non visti.

4.2 Primo esperimento: Standard

Nel primo esperimento si è provato ad allenare i modelli proposti in precedenza sugli esempi del dataset con le sole manipolazioni raffigurate nel Capitolo 2, inerenti il preprocessing iniziale dei dati.

Iperparametri. Di seguito si riportano gli iperparametri ottenuti dalla *GridSearchCV* per ogni modello:

Parametro	**Valore**
DecisionTree__criterion	gini
DecisionTree__max_depth	40
DecisionTree__min_samples_split	10
DecisionTree__min_samples_leaf	20
RandomForest__n_estimators	100
RandomForest__max_depth	20
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	2
RandomForest__criterion	gini
LogisticRegression__penalty	l2
LogisticRegression__C	10
LogisticRegression__solver	sag

Tabella 6: Migliori parametri dei modelli.

Risultati. Di seguito si riportano i risultati ottenuti per ogni metrica e per ogni modello:

Modello	**Accuracy**	**F1 Score**	**Precision**	**Recall**	**Bal. Accuracy**
Decision Tree	0.7480	0.7358	0.7309	0.7480	0.6619
Random Forest	0.7661	0.7510	0.7503	0.7661	0.6723
Logistic Regression	0.7582	0.7426	0.7392	0.7582	0.6643

Tabella 7: Confronto Metriche di Performance per Dataset Originale

Curve di apprendimento. Di seguito si riportano le curve di apprendimento per ogni modello, con tanto di deviazione standard e varianza:

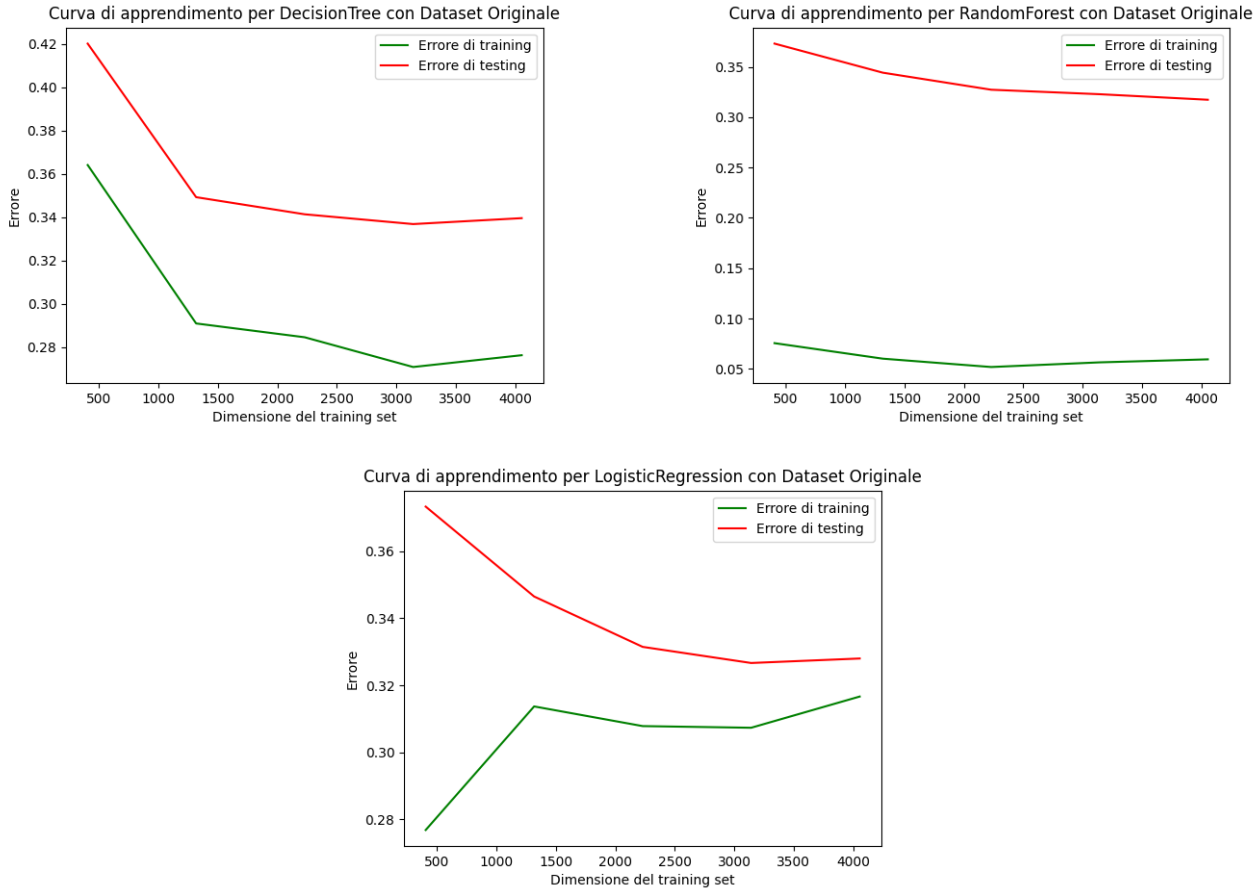


Figura 1: Curve di apprendimento Dataset Originale.

Modello	**Train Error Std**	**Test Error Std**
Decision Tree	0.0069919	0.0236521
Random Forest	0.0028087	0.0209585
Logistic Regression	0.0035906	0.0208664

Tabella 8: Confronto Deviazione Standard dell'Errore per i Modelli

Modello	**Train Error Var**	**Test Error Var**
Decision Tree	0.0000489	0.0005594
Random Forest	0.0000079	0.0004393
Logistic Regression	0.0000129	0.0004354

Tabella 9: Confronto Varianza degli Errori per i Modelli

Valutazione. I primi risultati sono abbastanza incoraggianti. Le metriche non ci danno informazioni particolarmente utili riguardante un possibile overfitting o underfitting, però ci fanno capire come già di base i modelli si comportano abbastanza bene.

Il grafico di apprendimento per il **Decision Tree** mostra che l'errore di training si stabilizza rapidamente, suggerendo che il modello apprende velocemente dai dati di addestramento. Tuttavia, l'errore di testing continua a diminuire senza stabilizzarsi del tutto, indicando che il modello potrebbe essere sottoaddestrato (underfitting). La discrepanza tra l'errore di

training e l'errore di testing suggerisce che il modello potrebbe essere capace di generalizzare ancora meglio sui dati non visti.
Per il **Random Forest**, l'errore di training è molto basso e tende a stabilizzarsi, mentre l'errore di testing è significativamente più alto, ma anch'esso in fase di diminuzione. La differenza tra l'errore di training e l'errore di testing suggerisce che il modello ha una buona capacità di apprendimento, ma ulteriori dati potrebbero migliorare ulteriormente la performance.
Nel grafico della **Logistic Regression**, l'errore di training diminuisce stabilmente mentre l'errore di testing segue una traiettoria simile, ma più lenta. La differenza tra i due errori diminuisce man mano che la dimensione del set di training aumenta, suggerendo che il modello sta migliorando nella generalizzazione con più dati.
In nessuno dei modelli sono presenti segni evidenti di overfitting, dato che l'errore di training non aumenta e l'errore di testing continua a scendere.

4.3 Secondo esperimento: Feature Eng.

Nel secondo esperimento addestriamo i modelli dopo aver eseguito correttamente tutte le operazioni sul dataset definite nel Capitolo 3. Verificheremo in seguito se le operazioni di Feature Engineering e la pulizia o semplificazione del dataset hanno portato a migliori prestazioni dei modelli.

Iperparametri. Di seguito si riportano gli iperparametri ottenuti dalla *GridSearchCV*, eseguita sul nuovo dataset, per ogni modello:

Parametro	**Valore**
DecisionTree__criterion	gini
DecisionTree__max_depth	10
DecisionTree__min_samples_split	20
DecisionTree__min_samples_leaf	20
RandomForest__n_estimators	200
RandomForest__max_depth	20
RandomForest__min_samples_split	5
RandomForest__min_samples_leaf	2
RandomForest__criterion	gini
LogisticRegression__penalty	l2
LogisticRegression__C	10
LogisticRegression__solver	newton-cg

Tabella 10: Migliori parametri dei modelli.

Risultati. Di seguito si riportano i risultati ottenuti per ogni metrica e per ogni modello:

Modello	**Accuracy**	**F1 Score**	**Precision**	**Recall**	**Bal. Accuracy**
Decision Tree	0.7471	0.7349	0.7308	0.7472	0.6593
Random Forest	0.7677	0.7501	0.7507	0.7677	0.6717
Logistic Regression	0.7623	0.7429	0.7426	0.7623	0.6618

Tabella 11: Confronto Metriche di Performance dopo Feature Engineering

Curve di apprendimento. Di seguito si riportano le curve di apprendimento per ogni modello, con tanto di deviazione standard e varianza:

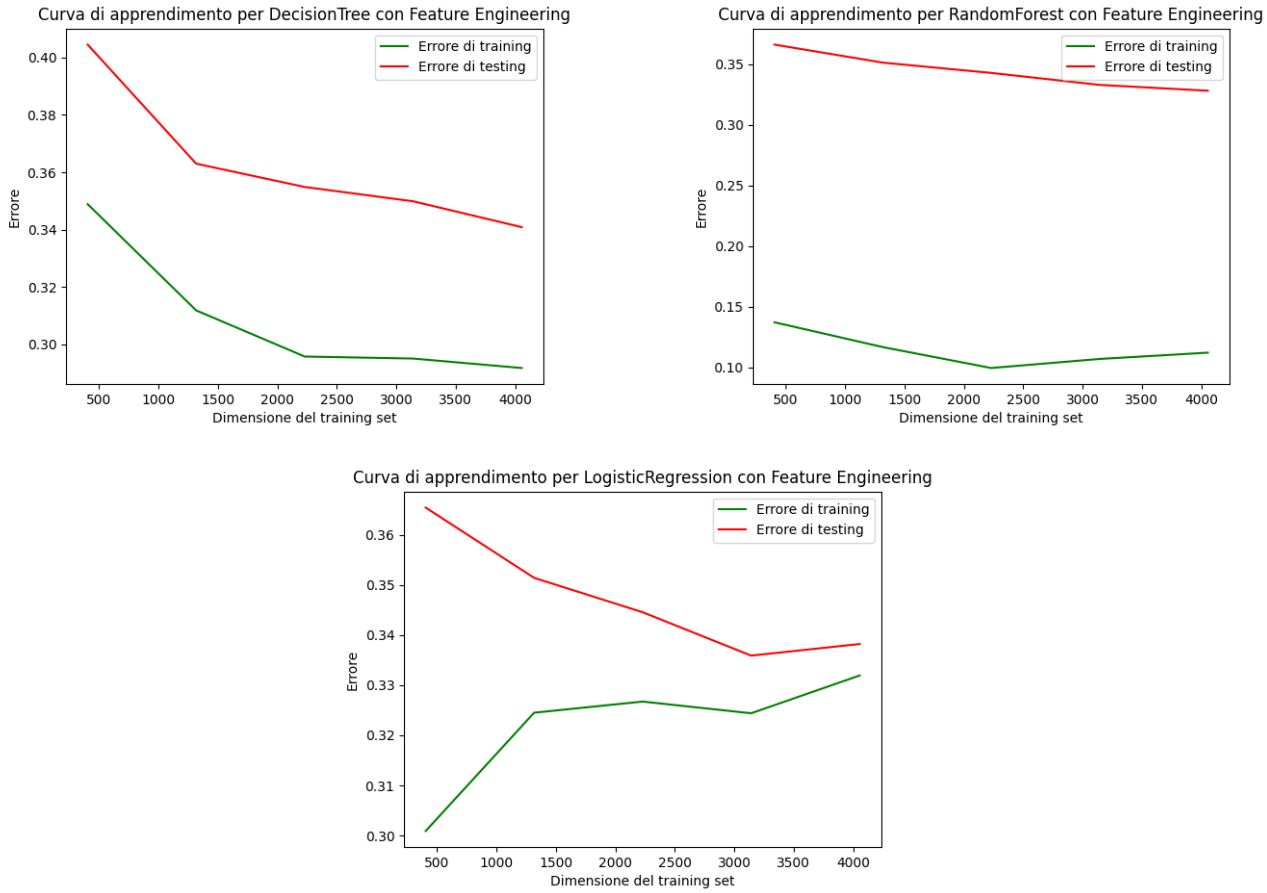


Figura 2: Curve di apprendimento dopo Feature Engineering.

Modello	**Train Error Std**	**Test Error Std**
Decision Tree	0.0069336	0.0308849
Random Forest	0.0029920	0.0188906
Logistic Regression	0.0023952	0.0149772

Tabella 12: Confronto Deviazione Standard dell'Errore per i Modelli

Modello	**Train Error Var**	**Test Error Var**
Decision Tree	0.0000481	0.0009539
Random Forest	0.0000090	0.0003569
Logistic Regression	0.0000057	0.0002243

Tabella 13: Confronto Varianza degli Errori per i Modelli

Valutazione. Dai risultati ottenuti notiamo come le operazioni di Feature Engineering abbiano avuto effetti diversi sui modelli. Il classificatore **Decision Tree** ha visto un aumento dell'errore e della varianza nei dati di test, suggerendo una minore stabilità. Al contrario, **Random Forest** e **Logistic Regression** hanno beneficiato delle operazioni di Feature Engineering, con miglioramenti (seppur leggeri) nelle metriche di performance e una maggiore

stabilità. In generale, il Classificatore su Random Forest è emerso come il modello con le migliori performance complessive dopo le operazioni di Feature Engineering, seguito dalla Logistic Regression. I risultati, seppur in certi casi migliori, sono comunque molto simili rispetto a quelli ottenuti nel Primo esperimento.

4.4 Terzo esperimento: SMOTE

In questo esperimento si decide di applicare una tecnica di oversampling. Essa consiste nell'aumentare artificialmente il numero di esempi delle classi minoritarie nel dataset di addestramento, in modo da bilanciare meglio la distribuzione delle classi. Ciò viene fatto replicando casualmente le istanze esistenti della classe minoritaria o generando nuove istanze sintetiche utilizzando tecniche come lo SMOTE [7], che testeremo.
SMOTE sintetizza nuove istanze per la classe minoritaria creando campioni artificiali lungo le linee che collegano istanze simili nello spazio delle feature. In questo modo, si cerca di mantenere la struttura dei dati originari mentre si aumenta il numero di istanze della classe minoritaria.
Gli algoritmi sono stati utilizzati con *random_state* uguale a *42*, e con *sampling_strategy* uguale a *all*, in modo da ribilanciare tutte le classi.

Iperparametri. Di seguito si riportano gli iperparametri ottenuti dalla *GridSearchCV* per ogni modello:

Parametro	**Valore**
DecisionTree__criterion	gini
DecisionTree__max_depth	10
DecisionTree__min_samples_split	2
DecisionTree__min_samples_leaf	20
RandomForest__n_estimators	200
RandomForest__max_depth	20
RandomForest__min_samples_split	5
RandomForest__min_samples_leaf	2
RandomForest__criterion	entropy
LogisticRegression__penalty	l2
LogisticRegression__C	10
LogisticRegression__solver	newton-cg

Tabella 14: Migliori parametri dei modelli.

Risultati. Di seguito si riportano i risultati ottenuti per ogni metrica e per ogni modello:

Modello	**Accuracy**	**F1 Score**	**Precision**	**Recall**	**Bal. Accuracy**
Decision Tree	0.7410	0.7422	0.7450	0.7411	0.7410
Random Forest	0.8200	0.8201	0.8223	0.8200	0.8200
Logistic Regression	0.7241	0.7240	0.7274	0.7241	0.7241

Tabella 15: Confronto Metriche di Performance dopo SMOTE

Curve di apprendimento. Di seguito si riportano le curve di apprendimento per ogni modello, con tanto di deviazione standard e varianza:

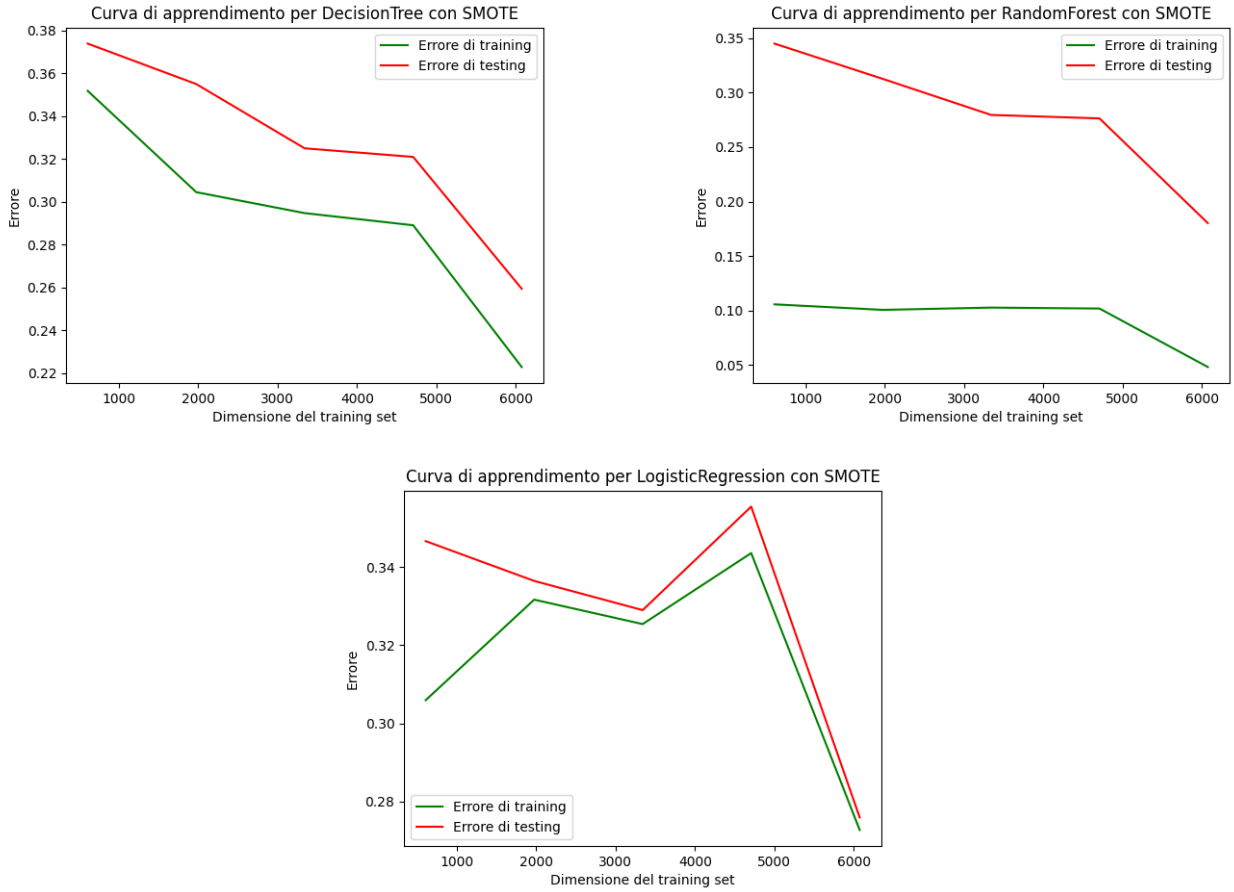


Figura 3: Curve di apprendimento con SMOTE.

Modello	**Train Error Std**	**Test Error Std**
Decision Tree	0.0030867	0.0171150
Random Forest	0.0013451	0.0139532
Logistic Regression	0.0022863	0.0166193

Tabella 16: Confronto Deviazione Standard dell'Errore per i Modelli

Modello	**Train Error Var**	**Test Error Var**
Decision Tree	0.0000095	0.0002929
Random Forest	0.0000018	0.0001947
Logistic Regression	0.0000052	0.0002762

Tabella 17: Confronto Varianza degli Errori per i Modelli

Valutazione. I risultati risultano essere migliori rispetto ai due esperimenti precedenti, sopratutto per il Random Forest, pertanto resta da controllare se vi è overfitting in qualche modello.
I valori di deviazione standard e varianza non portano a pensare a overfitting. Guardando le curve **Decision Tree** e **Random Forest** beneficiano chiaramente dell'uso di SMOTE, con entrambe le curve di errore che diminuiscono costantemente con l'aumento dei dati di training. La **Logistic Regression** mostra una variabilità maggiore nelle prestazioni, suggerendo che potrebbe non essere il modello ottimale in combinazione con SMOTE per

questo dataset.
Notiamo inoltre come, in questo esperimento, i valori di Accuracy e Balanced Accuracy coincidano. Questo è evidente dal fatto che i modelli sono stati addestrati su un dataset ribilanciato, e che le operazioni eseguite dallo SMOTE sono state eseguite correttamente.

4.5 Quarto esperimento: 2 categorie

Con l'obiettivo di migliorare ulteriormente l'accuratezza della previsione del nostro modello di apprendimento automatico, si è deciso di semplificare la variabile target in due categorie: Dropout (1) e No Dropout (0).
Adattando il dataset a questo cambiamento, trasformiamo la feature *Target* in una nuova feature binaria, unendo gli stati *Enrolled* e *Graduate* nella categoria *No Dropout*. Questa modifica mira a semplificare la previsione e a migliorare potenzialmente le prestazioni dei modelli, apprendendo unicamente una distinzione tra studenti che abbandonano e non.

Poichè anche a seguito dell'unificazione il dataset risulta essere sbilanciato, verrà comunque applicato l'algoritmo SMOTE come nell'esempio precedente.

Iperparametri. Di seguito si riportano gli iperparametri ottenuti dalla *GridSearchCV* per ogni modello:

Parametro	**Valore**
DecisionTree__criterion	gini
DecisionTree__max_depth	5
DecisionTree__min_samples_split	2
DecisionTree__min_samples_leaf	20
RandomForest__n_estimators	10
RandomForest__max_depth	10
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	5
RandomForest__criterion	entropy
LogisticRegression__penalty	l2
LogisticRegression__C	1
LogisticRegression__solver	newton-cg

Tabella 18: Migliori parametri dei modelli.

Risultati. Di seguito si riportano i risultati ottenuti per ogni metrica e per ogni modello:

Modello	**Accuracy**	**F1 Score**	**Precision**	**Recall**	**Bal. Accuracy**
Decision Tree	0.8461	0.8457	0.8492	0.8461	0.8461
Random Forest	0.8641	0.8640	0.8658	0.8641	0.8641
Logistic Regression	0.8548	0.8547	0.8560	0.8548	0.8548

Tabella 19: Confronto Metriche di Performance dopo SMOTE + Target Binario

Curve di apprendimento. Di seguito si riportano le curve di apprendimento per ogni modello, con tanto di deviazione standard e varianza:

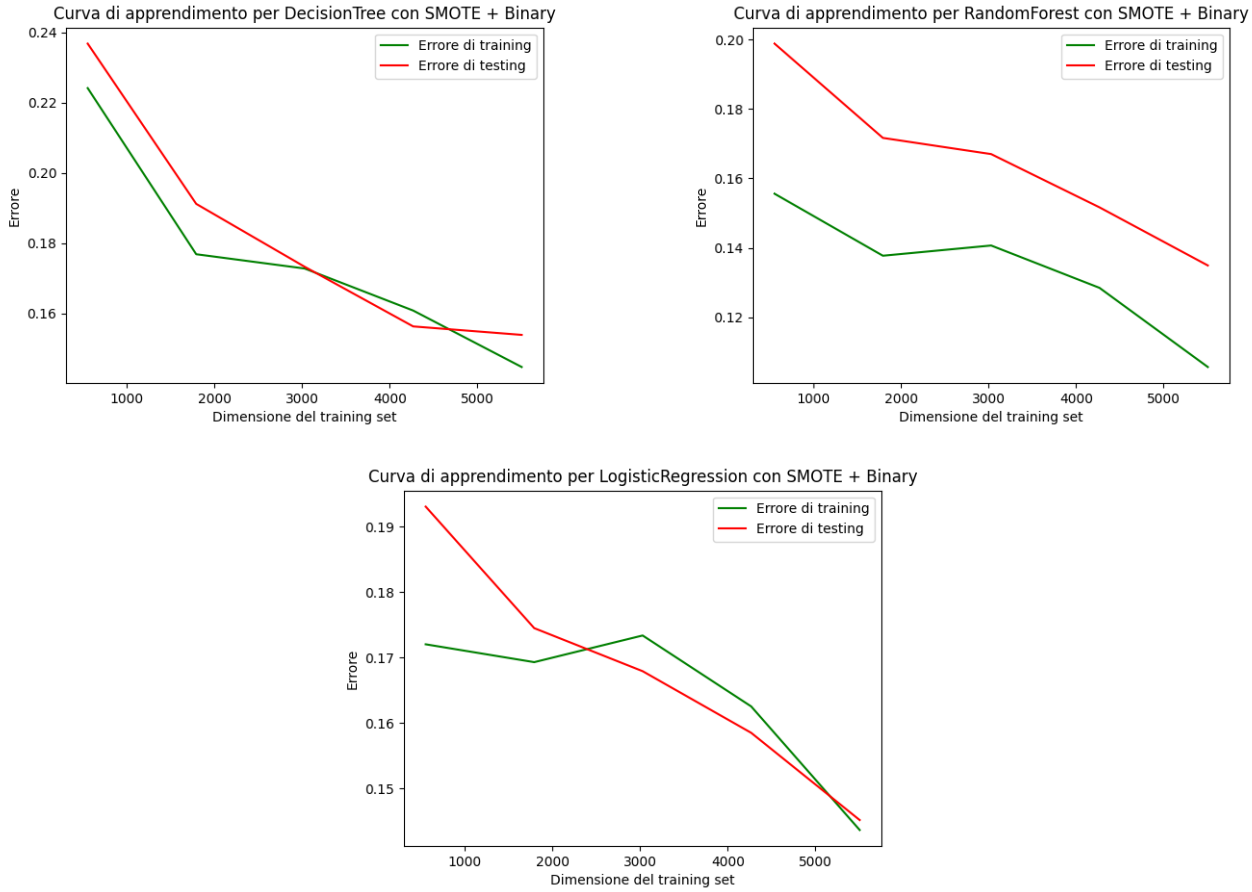


Figura 4: Curve di apprendimento con SMOTE + Target Binario.

Modello	**Train Error Std**	**Test Error Std**
Decision Tree	0.0026755	0.0141261
Random Forest	0.0027737	0.0146419
Logistic Regression	0.0016199	0.0156463

Tabella 20: Confronto Deviazione Standard dell'Errore per i Modelli

Modello	**Train Error Var**	**Test Error Var**
Decision Tree	0.0000072	0.0001995
Random Forest	0.0000077	0.0002144
Logistic Regression	0.0000026	0.0002448

Tabella 21: Confronto Varianza degli Errori per i Modelli

Valutazione. I risultati da questo ultimo esperimento sono eccellenti, considerando sopratutto quelli ottenuti inizialmente.
Per ciascuno dei grafici, l'andamento parallelo e la convergenza delle due curve indicano un buon adattamento del modello senza overfitting significativo.
In sintesi, tutti e tre i modelli beneficiano in prestazioni, il **Logistic Regression** sembra offrire il miglior equilibrio tra capacità di adattamento e generalizzazione, mentre il **Random Forest** mostra il miglior risultato in termini di riduzione dell'errore.

4.6 Quinto Esperimento: VotingClassifier

L'ultimo esperimento condotto sul dataset consiste nell'implementazione di un modello ensemble. Esso è un metodo di machine learning che utilizza più modelli individuali per produrre una singola predizione combinata. L'idea è che combinando le capacità di diversi modelli, si possano ottenere risultati migliori rispetto a quelli ottenuti da ciascun modello singolarmente. Più specificatamente, utilizzeremo un VotingClassifier con un meccanismo di soft voting, è una tecnica specifica di ensemble che utilizza il principio del voto per combinare le predizioni di vari modelli. Esistono due principali tipi di voting: hard voting e soft voting. Si sceglie, per questo esempio, un meccanismo di soft voting in cui ogni modello non vota solo per una classe, ma fornisce una distribuzione di probabilità su tutte le possibili classi. Le probabilità vengono quindi combinate (ad esempio, mediandole) e la classe con la probabilità media più alta viene scelta come predizione finale.
I modelli considerati nel VotingClassifier sono: Decision Tree, Logistic Regression e **Rete Neurale Multilayer Perceptron**. Notiamo come in questo esperimento il modello Random Forest sia stato sostituito da una Rete Neurale MLP in quanto esso risulta, per architettura, già un Ensemble Model.

Iperparametri. Di seguito si riportano gli iperparametri ottenuti dalla *GridSearchCV* per ogni modello:

Parametro	**Valore**
DecisionTree__criterion	gini
DecisionTree__max_depth	5
DecisionTree__min_samples_split	5
DecisionTree__min_samples_leaf	20
NeuralNetwork__hidden_layer_sizes	(50,)
NeuralNetwork__activation	tanh
NeuralNetwork__solver	adam
NeuralNetwork__alpha	0.01
NeuralNetwork__learning_rate	adaptive
LogisticRegression__penalty	l2
LogisticRegression__C	1
LogisticRegression__solver	newton-cg

Tabella 22: Migliori parametri dei modelli.

Risultati. Di seguito si riportano i risultati ottenuti per ogni metrica per il modello:

Modello	**Accuracy**	**F1 Score**	**Precision**	**Recall**	**Bal. Accuracy**
Voting Classifier	0.8643	0.8611	0.8629	0.8643	0.8247

Tabella 23: Confronto Metriche di Performance dopo Target Binario + MLP e SMOTE Off su Voting Classifier

Curve di apprendimento. Di seguito si riportano le curve di apprendimento per il modello, con tanto di deviazione standard e varianza:

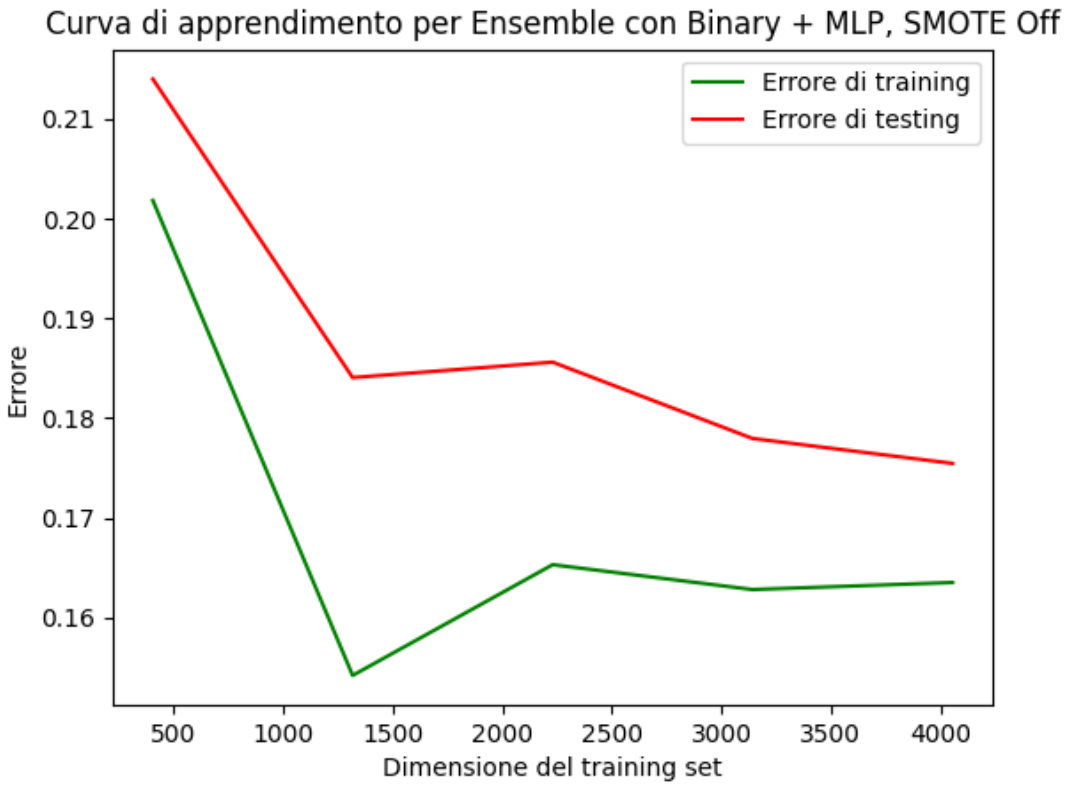


Figura 5: Curve di apprendimento con Target Binario + MLP e SMOTE Off su Voting Classifier

Modello	**Train Error Std**	**Test Error Std**
Voting Classifier	0.0036155	0.0178516

Tabella 24: Deviazione Standard dell'Errore per il Voting Classifier

Modello	**Train Error Var**	**Test Error Var**
Voting Classifier	0.0000013	0.0003186

Tabella 25: Varianza degli Errori per il Voting Classifier

Valutazione. I risultati anche da questo esperimento sono eccellenti, considerando sopratutto quelli ottenuti inizialmente.
L'andamento delle curve di apprendimento suggerisce che il modello di Ensemble è efficace e beneficia dell'aumento dei dati di training.
I risultati ottenuti, tuttavia, non si discostano rispettoa quelli ottenuti nel Quarto Esperimento.

4.7 Sommario

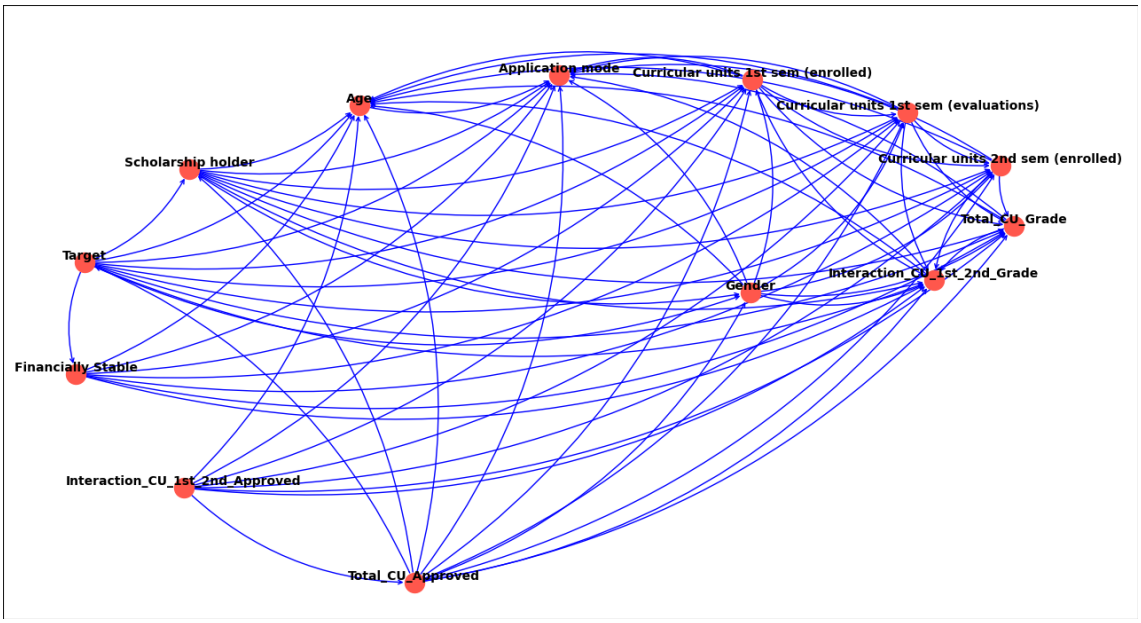
In generale, i risultati ottenuti sopratutto negli ultimi esperimenti sono molto accettabili. Il Random Forest detiene globalmente i migliori risultati in termini di riduzione dell'errore, rispetto agli altri modelli.

5 Ragionamento probabilistico e Bayesian Network

Il ragionamento probabilistico è una forma di ragionamento che sfrutta la teoria della probabilità, in particolar modo dipendenza e indipendenza tra variabili e regola di Bayes. Nel ragionamento probabilistico si assegnano probabilità a ipotesi ed eventi e si utilizzano le probabilità a posteriori per i ragionamenti. Un'applicazione del ragionamento probabilistico sono le reti bayesiane. Queste vengono rappresentate mediante grafi orientati aciclici (DAG) dove ogni nodo del grafo rappresenta una variabile e gli archi indicano le dipendenze probabilistiche tra le variabili.

5.1 Apprendimento della struttura

Date le risorse computazionali a disposizione, si è sfruttata la *feature importance* correlata con il Target trattata in Sezione 3.3. Esse saranno quindi le variabili da mantenere per apprendere la struttura della rete bayesiana. Le feature che verranno utilizzate sono le seguenti: *Application mode, Gender, Scholarship holder, Age, Curricular units 1st sem (enrolled), Curricular units 1st sem (evaluations), Curricular units 2nd sem (enrolled), Financially Stable, Interaction_CU_1st_2nd_Approved, Interaction_CU_1st_2nd_Grade, Total_CU_Approved, Total_CU_Grade, Target.*
I valori continui sono stati discretizzati mediate *KBinsDiscretizer*, la struttura è stata appresa mdiante l'utilizzo dell'*HillClimbSearch* e come *estimator* la *Maximum Likelihood*.
Di seguito si riporta la struttura della rete:



Di seguito si riporta l'esempio di una *CPD*:

- **CPD per la variabile Gender**: Essa dipende dalle variabili *Scholarship holder, Target, Total_CU_Approved*, In questo caso stiamo dicendo che P(Gender=0 | Scholarship holder = 4, Target = 4, Total_CU_Approved = 4) = 0.5. Questo significa che, dati gli esempi nel dataset, si è appreso che, se Scholarship holder = 4, Target = 4 e Total_CU_Approved = 4 allora Gender ha il 50% di probabilità di assumere quel valore.

```
+--------------------+-----+--------------------------+
| Scholarship holder | ... | Scholarship holder(4.0)  |
+--------------------+-----+--------------------------+
| Target             | ... | Target(4.0)              |
+--------------------+-----+--------------------------+
| Total_CU_Approved  | ... | Total_CU_Approved(4.0)   |
+--------------------+-----+--------------------------+
| Gender(0.0)        | ... | 0.5                      |
+--------------------+-----+--------------------------+
| Gender(4.0)        | ... | 0.5                      |
+--------------------+-----+--------------------------+
```

Non si è riuscito ad ottenere in tempi ragionevoli la stampa di CPD di variabili aventi più genitori a causa di mancanza di tempo e/o risorse di calcolo.

5.2 Generazione di sample e gestione di dati mancanti

Tramite la rete bayesiana è possibile generare nuovi sample, ovvero nuove configurazioni di variabili. Questo è utile per generare nuovi dati da utilizzare per addestrare un modello, o per fare inferenza su nuovi dati.

- Vengono sfruttate le probabilità apprese dal dataset in fase di addestramento per generare un esempio sintetico credibile:

```
Application mode
4.0
Interaction_CU_1st_2nd_Approved
0.0
Scholarship holder
0.0
Gender
0.0
Financially Stable
1
Curricular units 2nd sem (enrolled)
1.0
Total_CU_Grade
3.0
Curricular units 1st sem (enrolled)
2.0
Curricular units 1st sem (evaluations)
2.0
Interaction_CU_1st_2nd_Grade
2.0
Age
1.0
Total_CU_Approved
2.0
```

Ovviamente l'esempio randomico viene generato già discretizzato.

Dell'esempio precedentemente mostrato è stata inoltre calcolata la probabilità di appartenenza ad una delle classi Target.

- Classificazione dell'esempio:

```
+-------------+---------------+
| Target      |   phi(Target) |
+=============+===============+
| Target(0.0) |        1.0000 |
+-------------+---------------+
| Target(4.0) |        0.0000 |
+-------------+---------------+
```

Le reti bayesiane sono in grado di gestire casi in cui una variabile di input non è nota. Questo è utile in quanto i dati reali spesso presentano valori mancanti. La rete bayesiana è in grado di fare inferenza su nuovi dati, anche se non tutte le variabili sono note.
Ad esempio, è stata eliminata la feature *Gender* dall'esempio precedentemente generata.

- Classificazione dell'esempio con la feature rimossa:

```
+-------------+---------------+
| Target      |   phi(Target) |
+=============+===============+
| Target(0.0) |        0.9470 |
+-------------+---------------+
| Target(4.0) |        0.0530 |
+-------------+---------------+
```

5.3 Valutazione

Introduciamo uno score che consente di stabilire quanto bene il modello descrive i dati osservati. Questo è utile per valutare la bontà del modello. Utilizzeremo il *correlation-score* offerto da *pgmpy*, che effettua un test di correlazione tra le variabili del dataset.
Il correlation-score della nostra rete bayesiana rispetto alla sua *balanced accuracy* è circa 0.50, questo indica che la rete è in grado di catturare una certa correlazione tra le variabili, ma non è in grado di catturare tutte le dipendenze probabilistiche.

5.4 Query di esempio

Le reti bayesiane sono in grado di rispondere a query probabilistiche, ovvero di calcolare la probabilità di una variabile dato un insieme di evidenze. Di seguito si riportano due esempi di query:

- Data la osservazione che uno studente è stabile dal punto di vista finanziazio, qual è la distribuzione di probabilità per Scholarship holder (Ovvero borsista o meno)?

```
infer = VariableElimination(bn)
query_report(infer, variables=['Scholarship holder'],
evidence={'Financially Stable': 1}, desc='')
```

Risultato della query:

```
+-------------------------+----------------------------+
| Scholarship holder      |   phi(Scholarship holder) |
+=========================+============================+
| Scholarship holder(0.0) |                     0.7327 |
+-------------------------+----------------------------+
| Scholarship holder(4.0) |                     0.2673 |
+-------------------------+----------------------------+
```

Secondo i dati, è più probabile NON aver vinto una borsa di studio se si è già finanziariamente stabili.

- Data la osservazione che uno studente ha una età avanzata (valore più alto discretizzato), qual è la distribuzione di probabilità per Financially Stable (finanziariamente stabile)?

```
infer = VariableElimination(bn)
query_report(infer, variables=['Financially Stable'],
evidence={'Age': 4.0}, desc='')
```

Risultato della query:

```
+-----------------------+----------------------------+
| Financially Stable    |   phi(Financially Stable) |
+=======================+============================+
| Financially Stable(0) |                     0.3595 |
+-----------------------+----------------------------+
| Financially Stable(1) |                     0.6405 |
+-----------------------+----------------------------+
```

Secondo i dati, è più probabile essere finanziariamente stabili in età avanzata.

5.5 Sommario

Dato lo sbilanciamento delle classi, e dato il numero minore di esempi su cui è stato allenata la rete bayesiana, possiamo tutto sommato dire che la rete bayesiana creata ha delle prestazioni accettabili, e permette di eseguire delle query probabilistiche interessanti.

6 Conclusioni

Abbiamo condotto uno studio sull'applicazione di tecniche di apprendimento supervisionato per classificare l' *Abbandono Scolastico* di studenti universitari. Tale studio è stato in parte supportato da una operazione preliminare di Feature Engineering che ha consentito il miglioramento delle feature del dataset. Per concludere, è stata creata una rete bayesiana per fare inferenza su nuovi dati e per rispondere a query probabilistiche. I risultati ottenuti con le varie tecniche sperimentate sono stati soddisfacenti, e nonostante la poca quantità di dati a disposizione, la rete bayesiana sembra rappresentare decentemente il dominio.

Riferimenti bibliografici

[1] J. Quinn, "Dropout and completion in higher education in europe among students from under-represented groups," 2013. An Independent report authored for the NESET network of experts.

[2] V. Realinho, J. Machado, L. Baptista, and M. V. Martins, "Predicting student dropout and academic success," *Data*, vol. 7, no. 11, 2022.

[3] D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*. Cambridge, UK: Cambridge University Press, 3 ed., 2023.

[4] D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*. Cambridge, UK: Cambridge University Press, 3 ed., 2023.

[5] D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*. Cambridge, UK: Cambridge University Press, 3 ed., 2023.

[6] D. Poole and A. Mackworth, *Artificial Intelligence: Foundations of Computational Agents*. Cambridge, UK: Cambridge University Press, 3 ed., 2023.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, p. 321–357, June 2002.