UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

Dipartimento di Informatica

Corso di laurea in Informatica

Caso di Studio per l'esame di Ingegneria della Conoscenza

Sistema intelligente per l'assegnazione del Corporate Credit Rating alle aziende

Progetto di:
Mattia Curri (758306) m.curri8@studenti.uniba.it

Link Repository:
https://github.com/mattiacurri/ProgettoICON

Anno Accademico **2023-2024**

Indice

1 Introduzione

Nell'attuale panorama economico globale, le valutazioni delle aziende svolgono un ruolo cruciale nel determinare il loro status finanziario e creditizio. Le agenzie di rating forniscono valutazioni e raccomandazioni ai mercati finanziari, agli investitori e alle stesse aziende stesse, influenzando direttamente le decisioni di investimento e i flussi di capitale. In questo contesto, l'applicazione di tecniche di apprendimento supervisionato si presenta come un'opportunità significativa per migliorare e automatizzare il processo di valutazione aziendale.
Il progetto ha l'obiettivo di predire il Corporate Credit Rating, l'opinione di un'agenzia indipendente sulla probabilità che una società adempia pienamente ai propri obblighi finanziari alla scadenza.
Come riporta [2], ogni agenzia ha un proprio sistema di rating, ma sono tutti simili. Ad esempio, Standard and Poor's utilizza "AAA" per la massima qualità del credito con il rischio di credito più basso, "AA" per la migliore, seguita da "A" e "BBB" per un credito soddisfacente. Questi rating sono considerati "investment grade", il che significa che la società valutata ha un livello di qualità richiesto da molte istituzioni. Tutto ciò che è inferiore a "BBB" è considerato speculativo o peggio, fino a un rating "D", che indica "spazzatura".
Il progetto utilizza i dati forniti e adottati da [4], base di partenza per l'implementazione del sistema, a cui è stato integrato [5], che utilizza un dataset differente e meno ricco di esempi, ma che comunque propone degli interessanti spunti di partenza per affrontare il problema. Inoltre, [4] propone come sviluppo futuro quello di trattare il problema come un problema multiclasse invece che binario e di considerare più settori, cosa che è stata fatta in questo studio, seppure non tenendo conto delle date dei rating.

1.1 Elenco argomenti di interesse

- **Apprendimento Supervisionato** [6]: Valutare le Predizioni, Alberi di Decisione, Cross Validation, Modelli compositi: Boosting e Bagging, *Oversampling, undersampling, tecniche miste, valutare le predizioni in presenza di sbilanciamento delle classi*

- **Apprendimento Probabilistico** [7]: 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):

- **Rating Agency**: l'agenzia di rating che ha assegnato il rating
- **Corporation**: la società valutata
- **Rating**: il rating assegnato
- **Rating Date**: la data in cui è stato assegnato il rating
- **CIK**: il CIK (Central Index Key) della società, ID univoco
- **Binary Rating**: il rating binario assegnato a seguito dello studio di [4]
- **Ticker**: il simbolo utilizzato per identificare la società in borsa
- **SIC Code**: codice utilizzato per classificare le attività economiche
- **Sector**: il settore di appartenenza della società, in [4] sono stati valutati soltanto gli esempi con settore "BusEq" (Business Equipment), qui verranno considerati tutti
- **Current Ratio**: rapporto tra attività correnti e passività correnti
- **Long-term Debt / Capital**: rapporto tra debito a lungo termine e capitale
- **Debt Equity Ratio**: quanto un'azieda è propensa a chiedere prestiti invece di chiedere finanziamenti agli azionisti
- **Net Profit Margin**: margine di profitto netto
- **Gross Margin**: margine di profitto lordo
- **Operation Margin**: profito dopo aver coperto i costi di produzione
- **EBIT Margin**: profito prima delle tasse
- **Asset Turnover**: rapporto tra vendite e attività totali
- **EBITDA Margin**: profito prima delle tasse, ammortamenti e svalutazioni
- **Pre-Tax Profit Margin**: profito prima delle tasse
- **ROE - Return on Equity**: profito netto su capitale proprio, ovvero l'efficienza dei soldi investiti
- **ROA - Return on Assets**: profito netto su attività totali
- **ROI - Return on Investment**: profito netto su investimenti
- **Return On Tangible Equity**: profito netto su capitale proprio tangibile
- **Operating Cash Flow Per Share**: flusso di cassa operativo per azione
- **Free Cash Flow Per Share**: flusso di cassa libero per azione

Il dataset è stato preprocessato come segue.
Si è scelto di seguire la strada di [4], ovvero quella di utilizzare un misto tra conoscenze di dominio e correlazione delle feature (utilizzando il coefficiente di Pearson) per ridurre il numero di feature. Sono state rimosse poi le feature *Rating Agency*, *Corporation*, *CIK*, *Binary Rating*, *Ticker*, *SIC Code*, in quanto non rilevanti per il problema in esame. Sono stati rimossi anche *Corporation* e *Ticker* (non seguendo quindi la suddivisione di esperimenti di [5]): si è deciso di affrontare il problema indipendentemente dalla società ma soltanto con uno spaccato di una società rappresentata dai valori numerici, economici e settoriali. *Rating Date* è stato trasformato in *Year Month Day*, featuredalla forma *YYYYMMDD*, in modo da poterlo utilizzare come feature numerica. E' stata poi applicata una tecnica di *one-hot encoding* sulle feature *Rating Agency* e *Sector*.
Per quanto rigurda **Rating**, il dataset presentava tutti i rating in formato testuale, pertanto si è deciso di mappare i rating a valori numerici. Prima di fare ciò, si è deciso di ridurre il numero di classi: per ogni valore di Rating, come ad esempio "A", una società può ricevere come rating sfaccettature come "A+" o "A-", risultando quindi in circa 22 classi. I dati a disposizione non permetterebbero di realizzare un modello che riesca a predire esattamente il rating di una società, pertanto si è deciso di ridurre il numero di classi a 4, mappando i rating come segue:

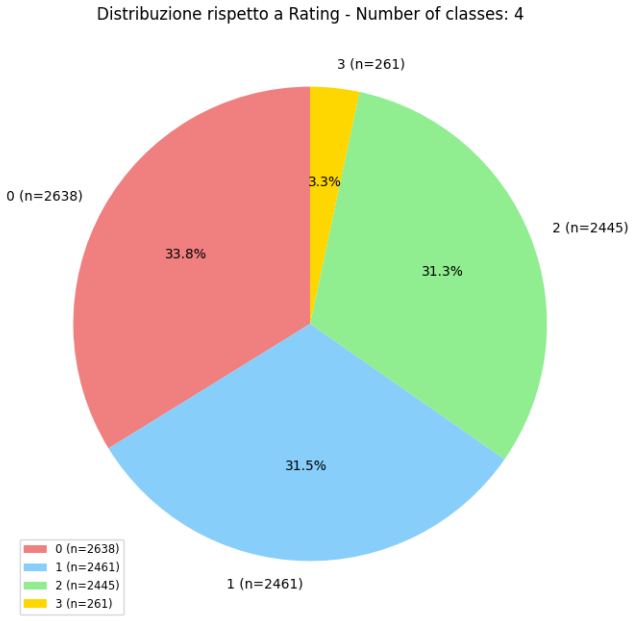
- **AAA** (Rischio minimo): 0
- **Da AA+ a A-** (Rischio basso): 0
- **BBB+, BBB, BBB-** (Rischio medio): 1
- **Da BB+ a B-** (Rischio alto): 2
- **Da CCC+ a C-** (Rischio molto alto): 3
- **D+, D, D-** (Default): 3

Il rischio molto alto e il default sono stati uniti, così come rischio minimo e rischio basso, per mancanza di dati per le classi più estreme. More on that later.
Infine, è stata applicata una normalizzazione *MinMaxScaler* ai dati numerici (escluse le colonne one-hot encoded e il rating), in modo da avere tutti i dati in un range compreso tra 0 e 1, tranne per *Asset Turnover* a cui è stata applicata la *PowerTransformer* perchè presentava una distribuzione molto sbilanciata.
Il **dataset finale** è così composto:

- **Rating Agency**: l'agenzia di rating che ha assegnato il rating (*one-hot encoded*)
- **Year Month Day**: la data in cui è stato assegnato il rating (con formato *YYYYMMDD*)
- **Current Ratio**: rapporto tra attività correnti e passività correnti (*normalizzato*)
- **Debt Equity Ratio**: quanto un'azieda è propensa a chiedere prestiti invece di chiedere finanziamenti agli azionisti (*normalizzato*)
- **Net Profit Margin**: margine di profitto netto (*normalizzato*)
- **Gross Margin**: margine di profitto lordo (*normalizzato*)
- **Asset Turnover**: rapporto tra vendite e attività totali (*normalizzato*)

- **EBITDA Margin**: profito prima delle tasse, ammortamenti e svalutazioni (*normalizzato*)
- **ROE - Return on Equity**: profito netto su capitale proprio, ovvero l'efficienza dei soldi investiti (*normalizzato*)
- **ROA - Return on Assets**: profito netto su attività totali (*normalizzato*)
- **ROI - Return on Investment**: profito netto su investimenti (*normalizzato*)
- **Return On Tangible Equity**: profito netto su capitale proprio tangibile (*normalizzato*)
- **Operating Cash Flow Per Share**: flusso di cassa operativo per azione (*normalizzato*)
- **Free Cash Flow Per Share**: flusso di cassa libero per azione (*normalizzato*)
- **Sector**: il settore di appartenenza della società (*one-hot encoded*)

Sono state quindi rimosse usando come discriminante il dominio *Long-term Debt / Capital*, *Operation Margin*, *Pre-Tax Profit Margin* e *EBIT Margin*.
A questo punto, dando un'occhiata ai dati, possiamo notare un netto sbilanciamento delle classi, in particolare quella di rischio molto alto.



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

3 Apprendimento Supervisionato

3.1 Modelli e metodologie utilizzate

Modelli adottati. Per qusto progetto ho deciso di adottare principalmente quattro modelli di apprendimento automatico:

- **Decision Tree**: 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

- **RandomForest**: 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)

- **XGBoost** [1]: utilizza un approccio di boosting basato sul gradiente che combina diverse versioni di alberi decisionali deboli per creare un modello complessivo più potente

- **LightGBM** [3]: altra libreria di boosting basata sul gradiente progettata per massimizzare l'efficienza computazionale. Si distingue per la sua velocità di addestramento e di previsione, grazie all'utilizzo di un algoritmo di divisione foglia-per-foglia e ad altre ottimizzazioni che minimizzano la memoria e la computazione necessarie.

Si è scelto di non usare modelli come ad esempio la regressione logistica, in quanto il rating può non essere direttamente correlato con le features a nostra disposizione (a titolo di esempio [8]), e quindi la linearità del modello non si presta bene. In alcuni esperimenti alcuni di questi modelli appena presentati sono stati sostituiti, in tal caso verrà specificato nella sezione apposita.

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.

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 $1 + \log_2(n)$, dove n è il numero di osservazioni **del training set**. In questo caso, per garantire un numero di fold non troppo elevato mantenendo comunque la regola, si è scelto di adottare la regola usando la seguente formula: $\frac{1 + \log_2(n)}{3}$.

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:

Classificatore LightGBM.

- **learning_rate**: Velocità con cui il modello impara dai dati durante il processo di addestramento. Un learning rate più basso richiede più iterazioni per raggiungere la convergenza, ma può portare a una migliore generalizzazione e a modelli più stabili. D'altra parte, un learning rate più alto potrebbe accelerare il processo di addestramento. **Range di valori:** [0.01, 0.05, 0.1].
- **max_depth**: Massima profondità dell'albero. **Range di valori:** [2, 5, 10].
- **n_estimators**: Numero di alberi. **Range di valori:** [50, 100, 200].
- **lambda**: Parametro di regolarizzazione L2. Un valore più alto di lambda aumenta la forza della regolarizzazione, riducendo così il rischio di overfitting. **Range di valori:** [0.01, 0.1, 0.5].
- **num_leaves**: Numero massimo di foglie per albero. **Range di valori:** [5, 15].
- **min_gain_to_split**: Guadagno minimo richiesto per eseguire uno split. **Range di valori:** [0.1].

Classificatore 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"].

Classificatore 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].

Classificatore XGBoost.

- **learning_rate**: Velocità con cui il modello impara dai dati durante il processo di addestramento. Un learning rate più basso richiede più iterazioni per raggiungere la convergenza, ma può portare a una migliore generalizzazione e a modelli più stabili. D'altra parte, un learning rate più alto potrebbe accelerare il processo di addestramento. **Range di valori:** [0.01, 0.05, 0.10].
- **max_depth**: Massima profondità degli alberi utilizzati. **Range di valori:** [5, 10, 20].
- **n_estimators**: Numero di alberi utilizzati. **Range di valori:** [20, 50, 100].
- **lambda**: Parametro di regolarizzazione L2. Un valore più alto di lambda aumenta la forza della regolarizzazione, riducendo così il rischio di overfitting. **Range di valori:** [0.01, 0.1, 0.5].

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:

- **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.

- **Cohen's Kappa** [9]: è una misura della concordanza tra le predizioni di un classificatore e i veri valori delle classi, correggendo l'accordo che potrebbe verificarsi solo per caso. Questa metrica tiene conto dell'equilibrio tra le classi e fornisce una valutazione più accurata rispetto alla semplice accuratezza. Il valore di Cohen's Kappa varia da -1 a 1, dove 1 indica una perfetta concordanza tra le predizioni e i veri valori, 0 indica concordanza casuale e valori negativi indicano concordanza peggiore di quella casuale

- **Geometric Mean**: è la radice del prodotto della sensibilità per classe. Questa misura cerca di massimizzare l'accuratezza su ciascuna delle classi, mantenendo queste accuratezze bilanciate. Per i problemi multiclasse è la radice n-esima del prodotto della sensibilità per ciascuna classe. Se almeno una classe non viene riconosciuta dal classificatore, la geometric mean sarà 0, pertanto ci servirà per dare soprattutto un *lower bound* di confronto per i nostri modelli.

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: un modello con una varianza e una deviazione standard più basse è preferibile, in quanto indica che le predizioni sono più coerenti e affidabili.

3.2 Primo esperimento

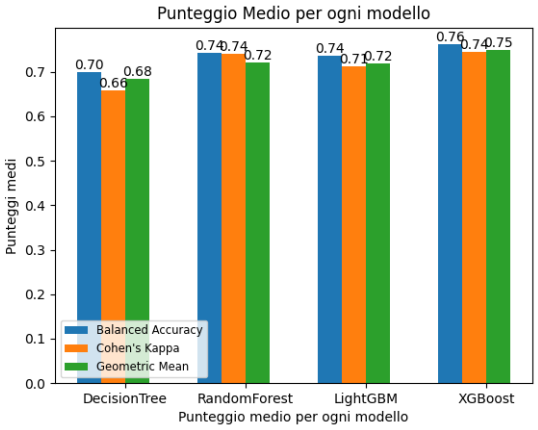
Nel primo esperimento si è provato ad allenare i modelli proposti in precedenza sugli esempi del dataset senza alcuna manipolazione successiva a quelle raccontate in precedenza. Sono state utilizzate le metriche e i modelli proposti in precedenza.

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

Parametro	**Valore**
LightGBM__learning_rate	0.1
LightGBM__max_depth	10
LightGBM__n_estimators	200
LightGBM__lambda	0.1
LightGBM__num_leaves	15
LightGBM__min_gain_to_split	0.1
LightGBM__verbose	0
XGBoost__learning_rate	0.1
XGBoost__max_depth	10
XGBoost__n_estimators	100
XGBoost__lambda	0.1
DecisionTree__criterion	log_loss
DecisionTree__max_depth	40
DecisionTree__min_samples_split	5
DecisionTree__min_samples_leaf	2
RandomForest__n_estimators	100
RandomForest__max_depth	20
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	2
RandomForest__criterion	log_loss

Tabella 1: Parametri del modello

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



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

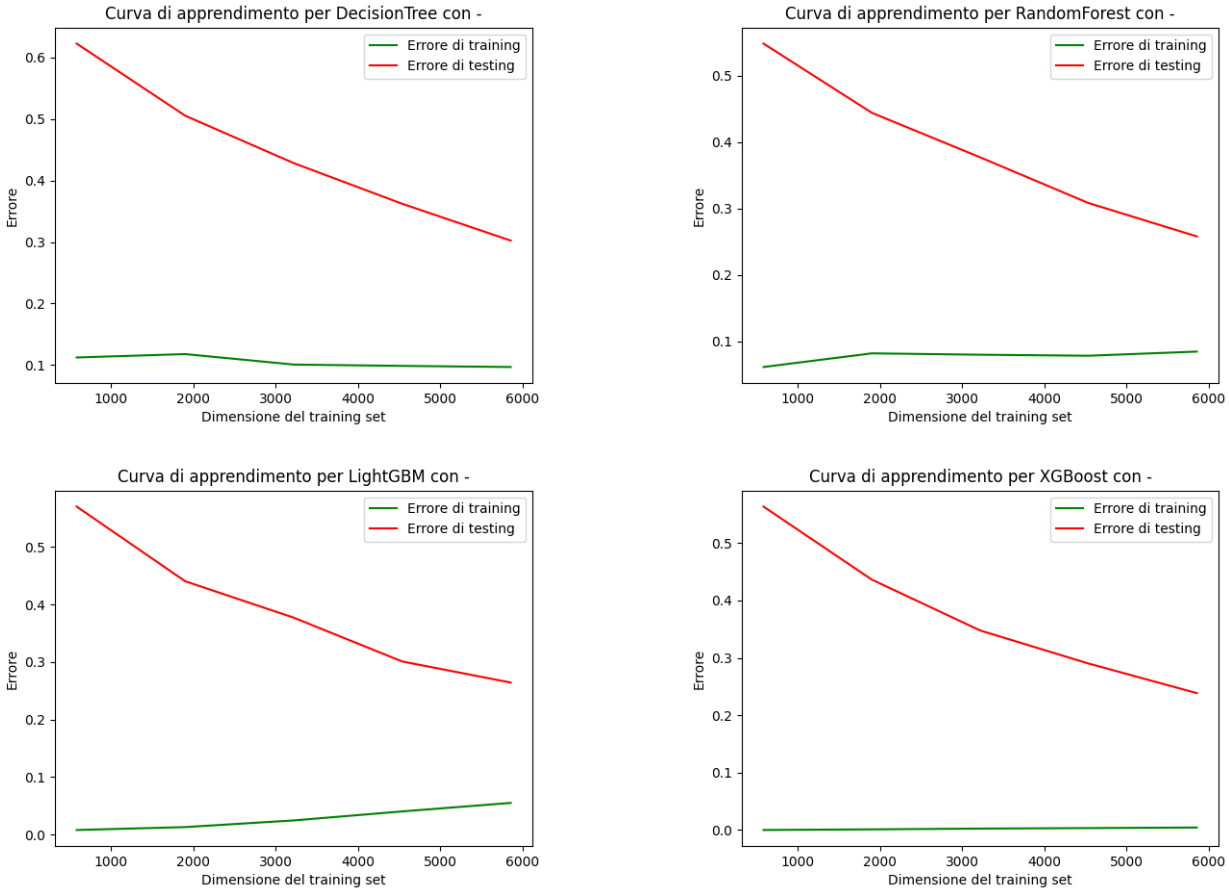


Figura 1: Curve di apprendimento.

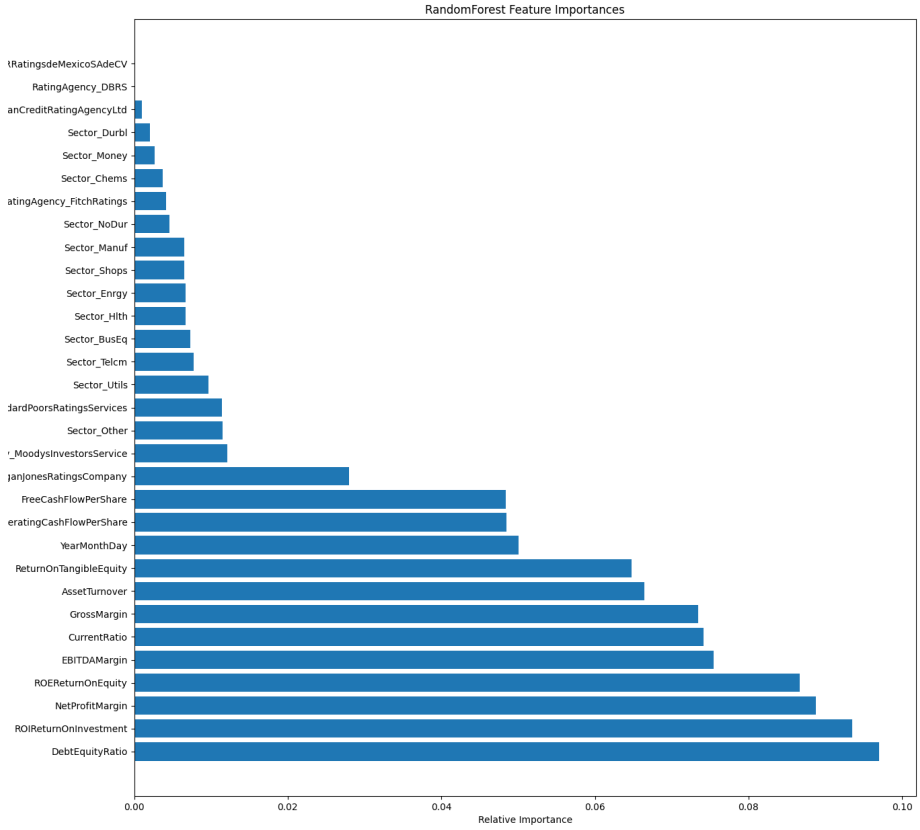
Modello	Train Error Std	Test Error Std
LightGBM	0.003677832760964418	0.014729799395024678
XGBoost	0.0017806001233474666	0.014258802458827638
DecisionTree	0.004798633588628468	0.015296962637326389
RandomForest	0.004364244206597904	0.014128073154668751

Tabella 2: Deviazione standard dell'errore per i Modelli

Modello	Train Error Var	Test Error Var
LightGBM	1.3526453817623153e-05	0.00021696699021766939
XGBoost	3.1705367992650135e-06	0.00020331344755986913
DecisionTree	2.3026884317913326e-05	0.0002339970659277595
RandomForest	1.9046627494823375e-05	0.00019960245106367185

Tabella 3: Varianza degli Errori per i Modelli

Feature importance. E' stata raccolta anche l'informazione riguardo quali feature abbiano contribuito di più in ogni modello per le predizioni.
Questo è stato fatto nel primo esperimento, in modo da utilizzare le feature maggiormente predittive per i modelli considerati migliori per costruire la rete bayesiana. Si riporta di seguito solo il grafico relativo al Random Forest, per non appesantire la trattazione, gli altri sono disponibili nel repository:



Valutazione. I primi risultati sono incoraggianti, soprattutto per i modelli basati su boosting e bagging, un pò meno per il decision tree. 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 **Decision Tree** non sembra dare segni di overfitting quanto quasi più di underfitting, dato che l'errore di training si stabilizza dopo circa metà training set, ma l'errore di testing è ancora in discesa ed è un pò lontano dall'errore di traning. Il **Random Forest** presenta un piccolo picco nell'errore di training dopo circa 2000 esempi, stessa cosa dopo altri 2000, ma comunque sono picchi poco significativi, anche qui l'errore di testing è abbastanza lontano, quindi non si può parlare di overfitting. Il **LightGBM** mostra dei possibili primi segnali di overfitting: l'errore di training inizia ad aumentare col tempo, l'errore di testing inizia a scendere abbastanza velocemente. L'**XGBoost** non presenta problemi, riesce ad essere allenato senza problemi, ma l'errore di testing è comunque ancora abbastanza alto, quindi non si può parlare di overfitting. In conclusione, i modelli non presentano segni di overfitting, ma il LightGBM potrebbe presentare dei segni di overfitting in futuro.

3.3 Secondo esperimento: class weights vs BalancedRF

La tecnica del class weight assegna pesi differenti alle classi durante il processo di addestramento del modello. In particolare, si attribuiscono pesi maggiori alle classi minoritarie e pesi minori alle classi maggioritarie. Questo consente al modello di dare maggiore importanza alle istanze delle classi minoritarie durante l'addestramento. In questo modo, il modello sarà in grado di apprendere meglio le caratteristiche delle classi minoritarie e di migliorare le

prestazioni su queste classi.
I pesi delle classi possono essere calcolati in diversi modi. Uno dei metodi comuni è utilizzare l'inverso della frequenza delle classi nel dataset, che è la tecnica che andremo ad utilizzare.

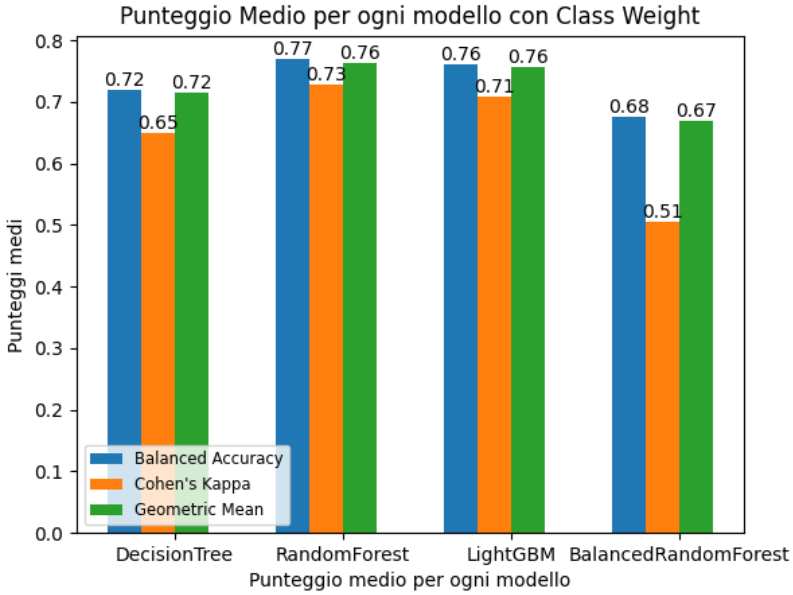
Per questo esperimento, andremo a usare tre modelli, tagliando fuori l'XGBoost, in quanto non supporta nativamente la possibilità di assegnare un peso alle classi.
In alternativa si utilizzerà un *BalancedRandomForest*, che si differenzia dal classico *Random Forest* per il fatto che estrarrà un campione su cui allenare ogni singolo *decision tree* dalla classe di minoranza e campionerà con sostituzione lo stesso numero di campioni dalla classe di maggioranza. E' un modello che si presta bene proprio per classificazione con classi sbilanciate, pertanto lo testeremo contro dei modelli impostati con un **bias** verso le classi minoritarie.

Iperparametri. Per il BalancedRandomForest lo spazio di ricerca degli iperparametri scelto è il medesimo dei Random Forest. Di seguito si riportano gli iperparametri ottenuti dalla *GridSearchCV* per ogni modello:

Parametro	**Valore**
BalancedRandomForest__criterion	entropy
BalancedRandomForest__n_estimators	200
BalancedRandomForest__max_depth	20
BalancedRandomForest__min_samples_split	2
BalancedRandomForest__min_samples_leaf	2
BalancedRandomForest__sampling_strategy	all
BalancedRandomForest__replacement	True
LightGBM__learning_rate	0.1
LightGBM__max_depth	10
LightGBM__n_estimators	200
LightGBM__lambda	0.5
LightGBM__num_leaves	15
LightGBM__min_gain_to_split	0.1
LightGBM__verbose	0
LightGBM__class_weight	balanced
DecisionTree__criterion	entropy
DecisionTree__max_depth	40
DecisionTree__min_samples_split	2
DecisionTree__min_samples_leaf	2
DecisionTree__class_weight	balanced
RandomForest__n_estimators	200
RandomForest__max_depth	20
RandomForest__min_samples_split	10
RandomForest__min_samples_leaf	2
RandomForest__criterion	log_loss
RandomForest__class_weight	balanced

Tabella 4: Parametri del modello

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



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

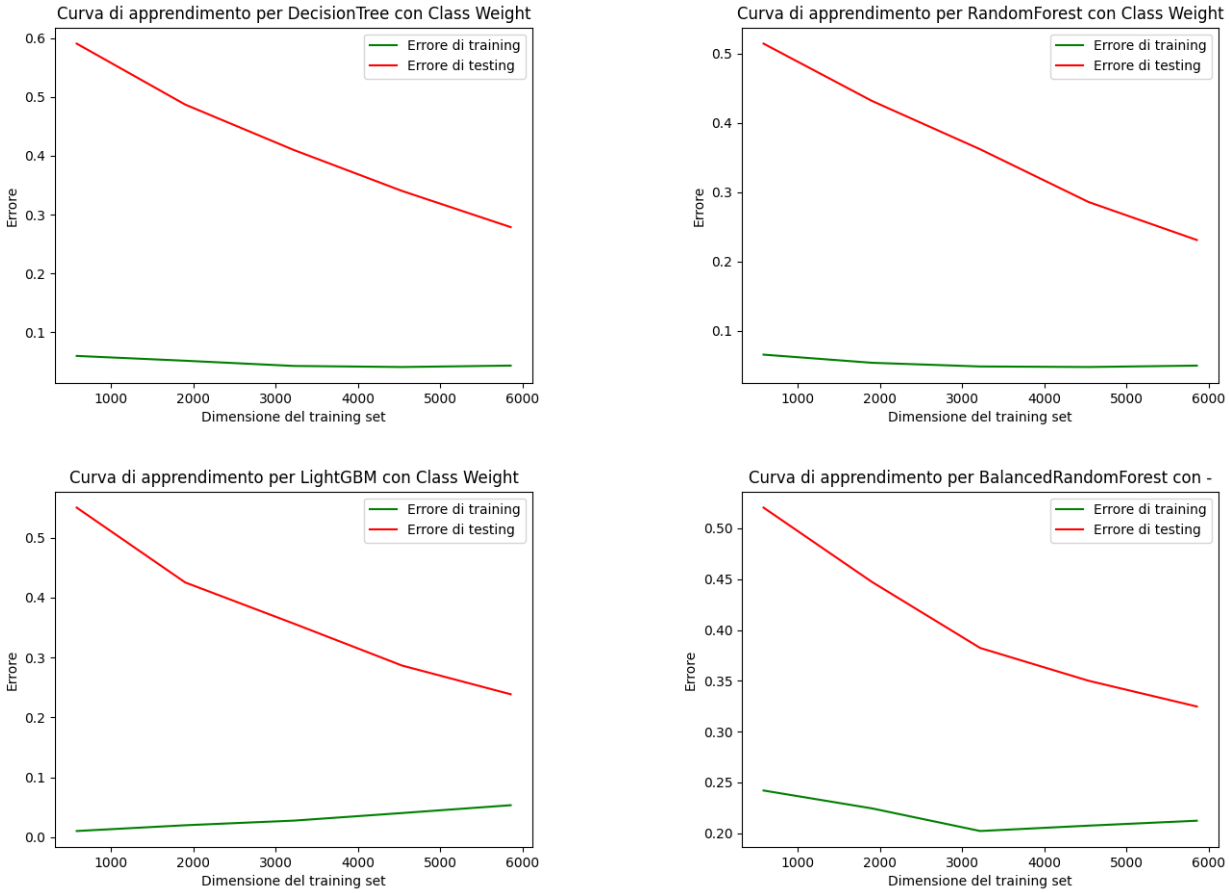


Figura 2: Curve di apprendimento.

Modello	Train Error Std	Test Error Std
LightGBM	0.0026561375826418482	0.016525860583798364
BalancedRandomForest	0.004339002154983959	0.009953134681109565
DecisionTree	0.0015879855371825314	0.015329002342180082
RandomForest	0.0018445912002196091	0.02129919916812779

Tabella 5: Deviazione standard dell'errore per i Modelli

Modello	Train Error Var	Test Error Var
LightGBM	7.055066857922481e-06	0.0002731040680351404
BalancedRandomForest	1.8826939700955435e-05	9.906488998030602e-05
DecisionTree	2.5216980663008927e-06	0.00023497831280656247
RandomForest	3.4025166959276185e-06	0.00045365588520357555

Tabella 6: Varianza degli Errori per i Modelli

Valutazione. A un primo impatto possiamo vedere come il BalancedRandomForest non è riuscito a mantenere le aspettative, soprattutto se confrontato con il suo "gemello", il Random Forest classico.
Rispetto al primo esperimento, le metriche di tutti gli altri modelli sono leggermente migliorate, ma siamo molto vicini ai range di varianza e deviazione standard del primo esperimento. Il Decision Tree sembra essersi stabilizzato ancora di più, il Random Forest presenta un errore di training decisamente più lineare, il LightGBM non presenta miglioramenti.

3.4 Terzo esperimento: oversampling

L'oversampling 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, 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.
Andremo a confrontare lo SMOTE con i risultati degli altri esperimenti e con quelli dell'ADASYN, un'altra tecnica di oversampling.
ADASYN genera nuovi esempi per le istanze delle classi minoritarie che hanno un alto grado di disomogeneità, ovvero che sono il più possibile differenti dagli esempi della classe maggioritaria per caratteristiche. Questa tecnica a differenza della precedente non riesce a garantire il perfetto ribilanciamento del dataset, ma ad ogni modo riesce quasi del tutto a risanarlo tranne nei casi in cui gli esempi della classe minoritaria sono sparsi e immersi in esempi di classe maggioritaria.
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. Come sviluppo futuro si potrebbe pensare di fare una sorta di *GridSearchCV* per trovare il miglior set di parametri per queste tecniche.

Attenzione. Si è prestata particolare attenzione nell'applicazione delle tecniche descritte nel seguito del capitolo. Infatti un errore comune è quello di utilizzare tecniche di oversampling, undersampling o miste su tutto il dataset, andando quindi ad allenare il modello sui dati sintetici, ma anche ad usare un test set con dati sintetici, che non rispecchia la realtà. Per questo motivo, si applicano le tecniche di oversampling soltanto sul training set, e non sul test set. Questo potrebbe portare a miglioramenti che non risultano entusiasmanti, ma è comunque la scelta migliore per valutare i nostri modelli.

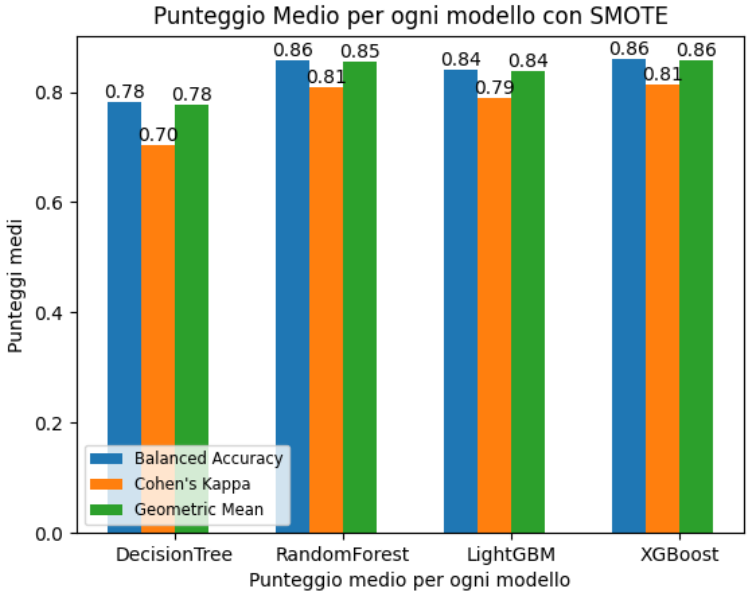
3.4.1 SMOTE

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

Parametro	**Valore**
LightGBM__learning_rate	0.1
LightGBM__max_depth	10
LightGBM__n_estimators	200
LightGBM__lambda	0.01
LightGBM__num_leaves	15
LightGBM__min_gain_to_split	0.1
XGBoost__learning_rate	0.1
XGBoost__max_depth	10
XGBoost__n_estimators	100
XGBoost__lambda	0.01
DecisionTree__criterion	gini
DecisionTree__max_depth	20
DecisionTree__min_samples_split	5
DecisionTree__min_samples_leaf	2
RandomForest__n_estimators	200
RandomForest__max_depth	20
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	2
RandomForest__criterion	entropy

Tabella 7: Parametri del modello

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



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

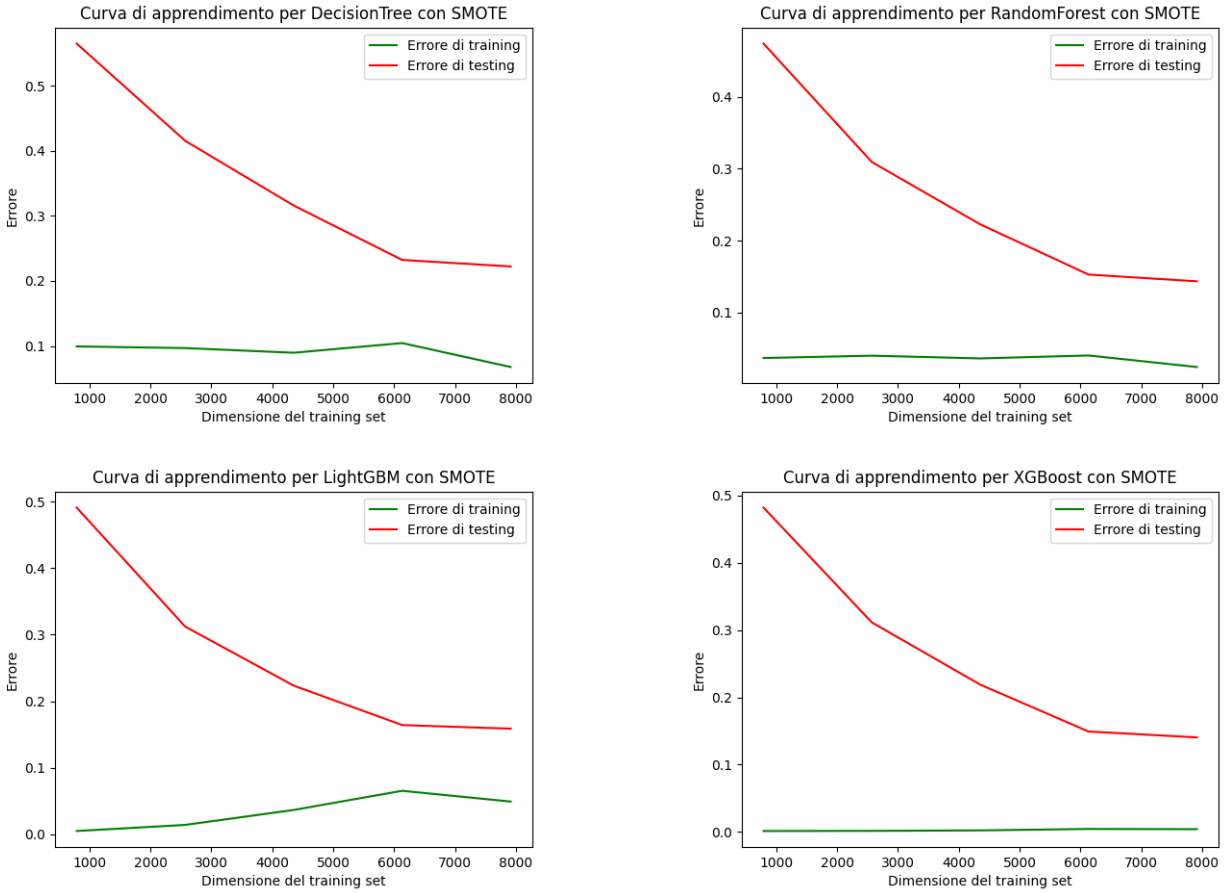


Figura 3: Curve di apprendimento.

Modello	Train Error Std	Test Error Std
LightGBM	0.0013274976900467363	0.005557462071090911
XGBoost	0.0010996943494845943	0.007626860052066837
DecisionTree	0.005071533109955392	0.006148834960506031
RandomForest	0.000739498264938402	0.004926394385656445

Tabella 8: Deviazione standard dell'errore per i Modelli

Modello	Train Error Var	Test Error Var
LightGBM	1.7622501170794206e-06	3.088538467161408e-05
XGBoost	1.209327662288345e-06	5.816899425381296e-05
DecisionTree	2.5720448085373803e-05	3.7808171371541206e-05
RandomForest	5.468576838469069e-07	2.426936164302734e-05

Tabella 9: Varianza degli Errori per i Modelli

Valutazione. I risultati sono decisamente migliori rispetto ai due esperimenti precedenti, 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 notiamo come l'XGBoost si sia comportato bene, stessa cosa per il Random Forest, mentre sorgono dubbi sull'efficacia del Decision Tree e del LightGBM, in particolare quest'ultimo sembra raggiungere una configurazione ideale tra i 6000 e gli 8000 esempi.

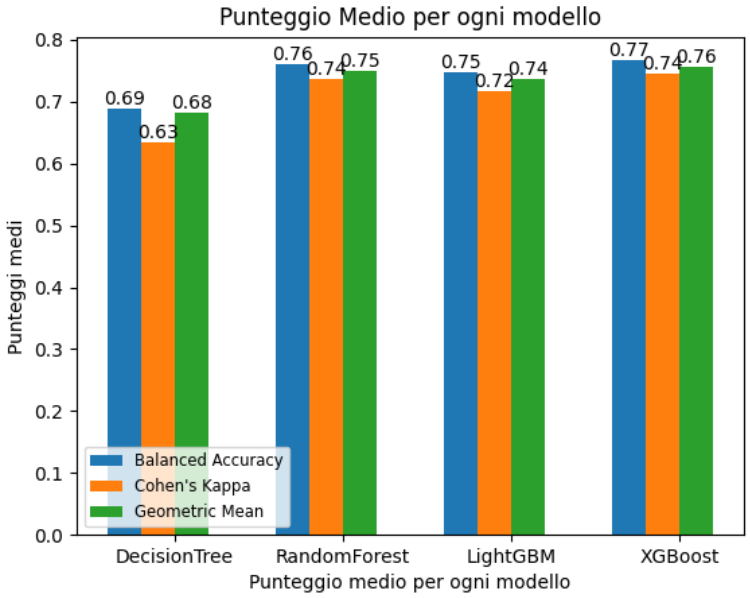
3.4.2 ADASYN

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

Parametro	Valore
LightGBM__learning_rate	0.1
LightGBM__max_depth	10
LightGBM__n_estimators	200
LightGBM__lambda	0.1
LightGBM__num_leaves	15
LightGBM__min_gain_to_split	0.1
XGBoost__learning_rate	0.1
XGBoost__max_depth	20
XGBoost__n_estimators	100
XGBoost__lambda	0.01
DecisionTree__criterion	gini
DecisionTree__max_depth	40
DecisionTree__min_samples_split	5
DecisionTree__min_samples_leaf	2
RandomForest__n_estimators	200
RandomForest__max_depth	20
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	2
RandomForest__criterion	log_loss

Tabella 10: Parametri del modello

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



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

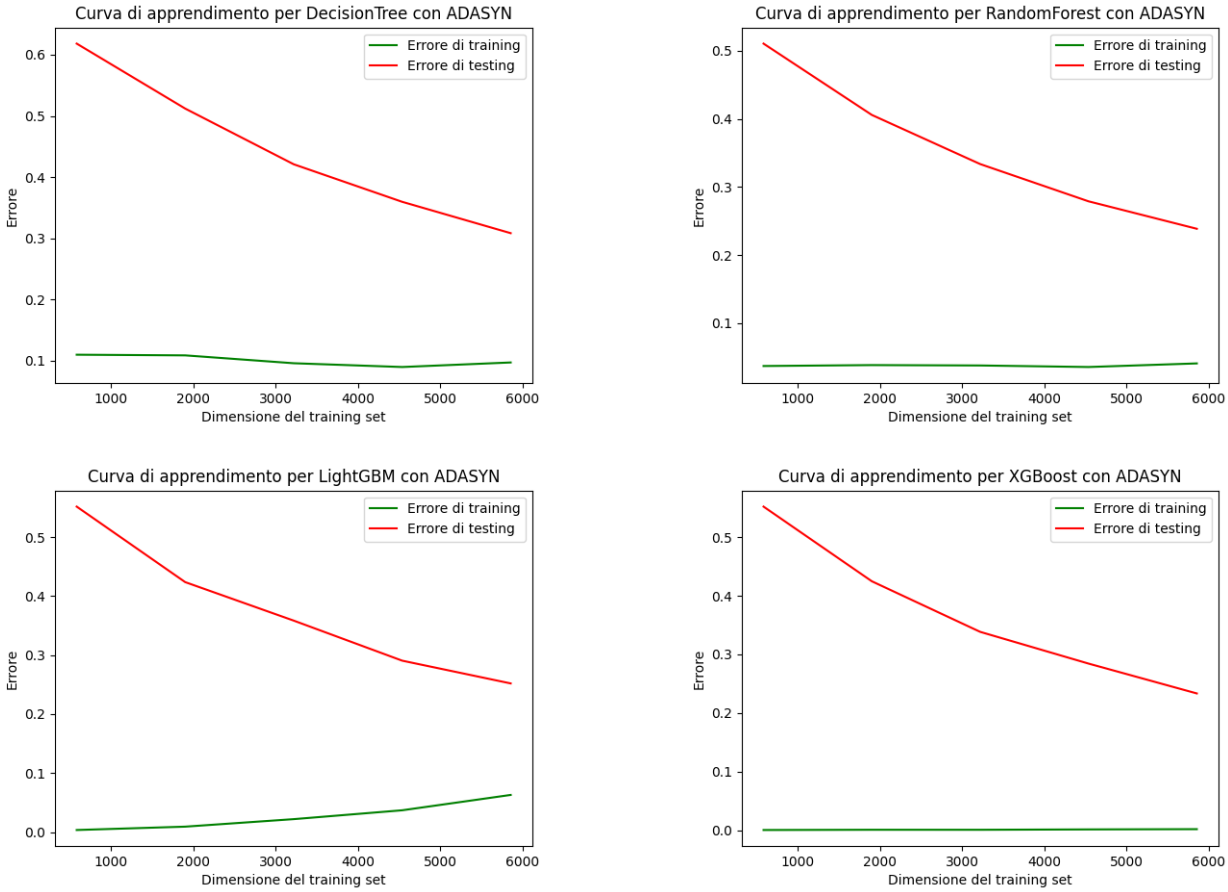


Figura 4: Curve di apprendimento.

Modello	Train Error Std	Test Error Std
LightGBM	0.0038361336791755867	0.014834571234512225
XGBoost	0.0010135798069650331	0.01775432088891302
DecisionTree	0.0050366494920735345	0.02674348515555034
RandomForest	0.0028043557998990395	0.01884405548317221

Tabella 11: Deviazione standard dell'errore per i Modelli

Modello	Train Error Var	Test Error Var
LightGBM	1.4715921604505222e-05	0.00022006450371181758
XGBoost	1.0273440250872736e-06	0.00031521591022649323
DecisionTree	2.5367838106004596e-05	0.0007152139982651415
RandomForest	7.86441145242738e-06	0.00035509842705287254

Tabella 12: Varianza degli Errori per i Modelli

Valutazione. I punteggi sono simili agli esperimenti precedenti, ma non migliori dello SMOTE.
Random Forest e XGBoost risultano molto stabili, l'XGBoost sembra avere avere varianza e deviazione standard migliori, mentre LightGBM ancora una volta è in difficoltà. Il Decision Tree si comporta benino, ma risultando comunque il peggiore tra i modelli.

3.5 Quarto esperimento: tecniche miste

Non si è fatto uso di tecniche di undersampling, ovvero di tecniche che riducono il numero di esempi della classe maggioritaria, in quanto dopo un tentativo con un algoritmo di questo tipo (*ClusterCentroids*, basato su *K-Means*), i risultati erano imprensentabili, portando a scartare del tutto questa tecnica. In alternativa si proverà a utilizzare delle tecniche miste di oversampling e undersampling: SMOTEEEN e SMOTETomek, tecniche che applicano prima lo SMOTE per fare oversampling, per poi "pulire" il dataset, rimuovendo *noise* e osservazioni "troppo facili da classificare", con tecniche di undersampling (EEN e Tomek).
Tomek rimuove quegli esempi di classi differenti che sono molto vicini tra loro, poichè saranno difficili da classificare e non aiutano l'algoritmo a trovare una discriminante.
EEN rimuove gli esempi di classe maggioritaria che sono molto vicini a quelli di classe minoritaria o gli esempi dove tutti o quasi i vicini sono di classi differenti, poichè potrebbero essere classificati erroneamente.

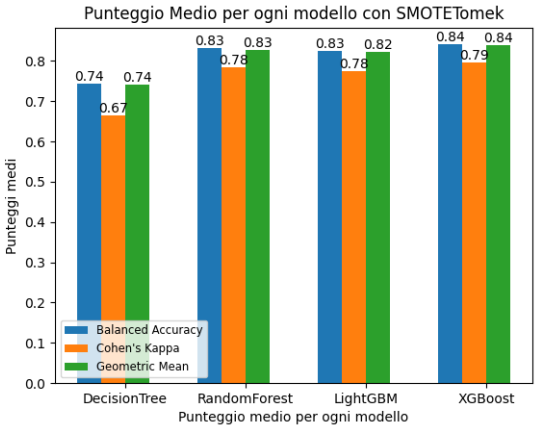
3.5.1 SMOTETomek

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

Parametro	**Valore**
LightGBM__learning_rate	0.1
LightGBM__max_depth	10
LightGBM__n_estimators	200
LightGBM__lambda	0.01
LightGBM__num_leaves	15
LightGBM__min_gain_to_split	0.1
XGBoost__learning_rate	0.1
XGBoost__max_depth	10
XGBoost__n_estimators	100
XGBoost__lambda	0.01
DecisionTree__criterion	gini
DecisionTree__max_depth	20
DecisionTree__min_samples_split	2
DecisionTree__min_samples_leaf	2
RandomForest__n_estimators	200
RandomForest__max_depth	20
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	2
RandomForest__criterion	log_loss

Tabella 13: Parametri del modello

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



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

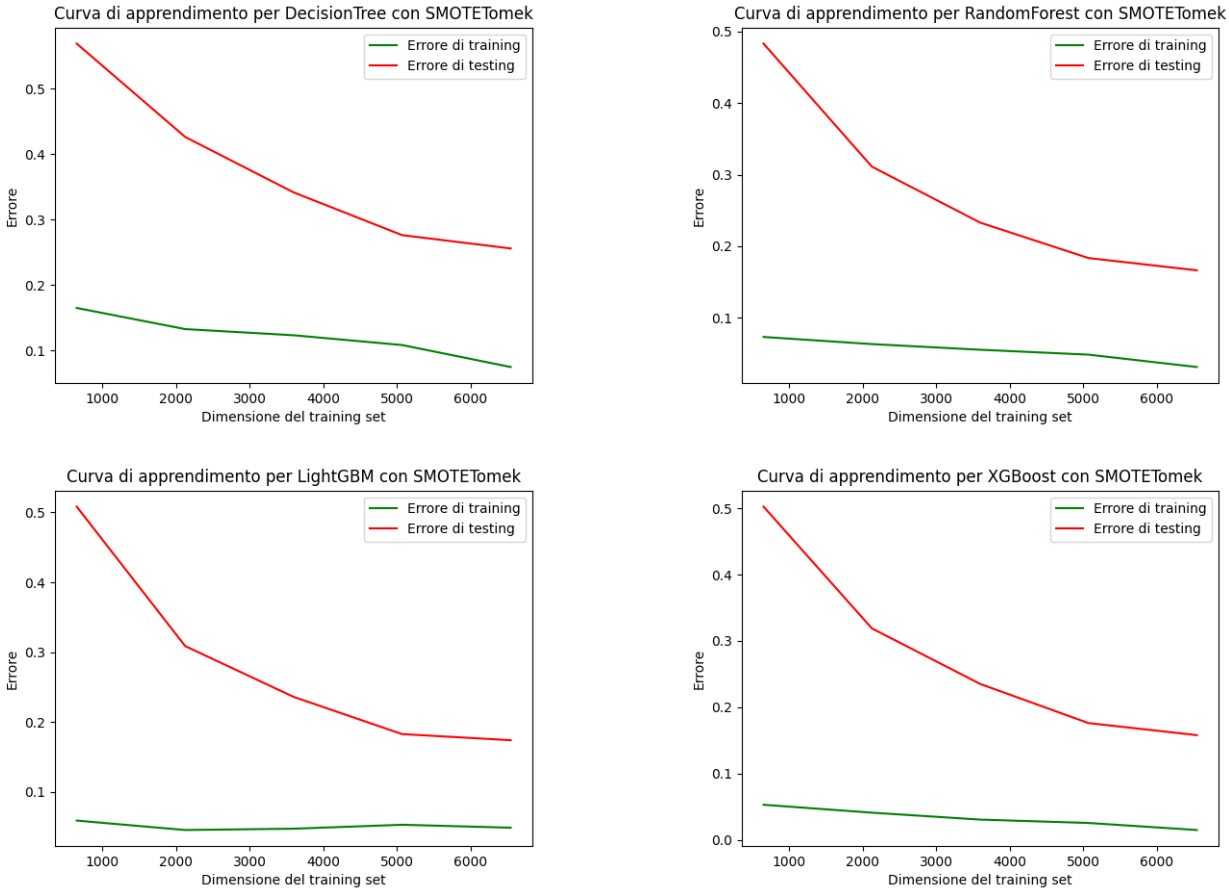


Figura 5: Curve di apprendimento.

Modello	Train Error Std	Test Error Std
LightGBM	0.0023555215150484524	0.009308465169567823
XGBoost	0.0007603005596334415	0.007147550348338646
DecisionTree	0.004106118784835311	0.010978248717168952
RandomForest	0.001156335935796055	0.005023366391826634

Tabella 14: Deviazione standard dell'errore per i Modelli

Modello	Train Error Var	Test Error Var
LightGBM	5.548481607856157e-06	8.664752381305731e-05
XGBoost	5.780569409789242e-07	5.1087475982035904e-05
DecisionTree	1.6860211475177415e-05	0.00012052194489602173
RandomForest	1.3371127964133382e-06	2.523420990653334e-05

Tabella 15: Varianza degli Errori per i Modelli

Valutazione. I risultati sono simili all'esperimento con SMOTE.
Varianza e deviazione standard sono nella maggior parte dei casi maggiori rispetto all'esperimento con SMOTE e in generale i modelli sembrano aver bisogno di più dati per apprendere, notiamo come l'errore di training non si stabilizza in nessuno dei modelli.

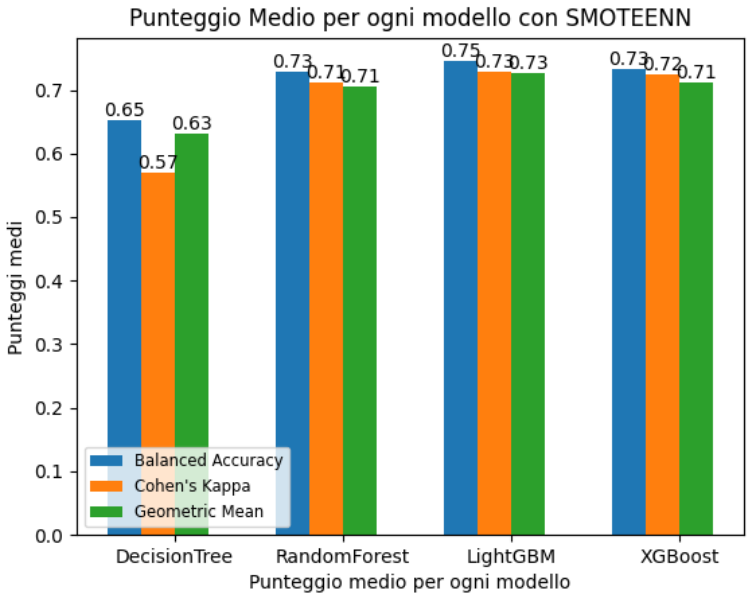
3.5.2 SMOTEENN

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

Parametro	Valore
LightGBM__learning_rate	0.05
LightGBM__max_depth	5
LightGBM__n_estimators	200
LightGBM__lambda	0.1
LightGBM__num_leaves	15
LightGBM__min_gain_to_split	0.1
XGBoost__learning_rate	0.1
XGBoost__max_depth	20
XGBoost__n_estimators	100
XGBoost__lambda	0.1
DecisionTree__criterion	log_loss
DecisionTree__max_depth	40
DecisionTree__min_samples_split	2
DecisionTree__min_samples_leaf	2
RandomForest__n_estimators	100
RandomForest__max_depth	20
RandomForest__min_samples_split	2
RandomForest__min_samples_leaf	2
RandomForest__criterion	gini

Tabella 16: Parametri del modello

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



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

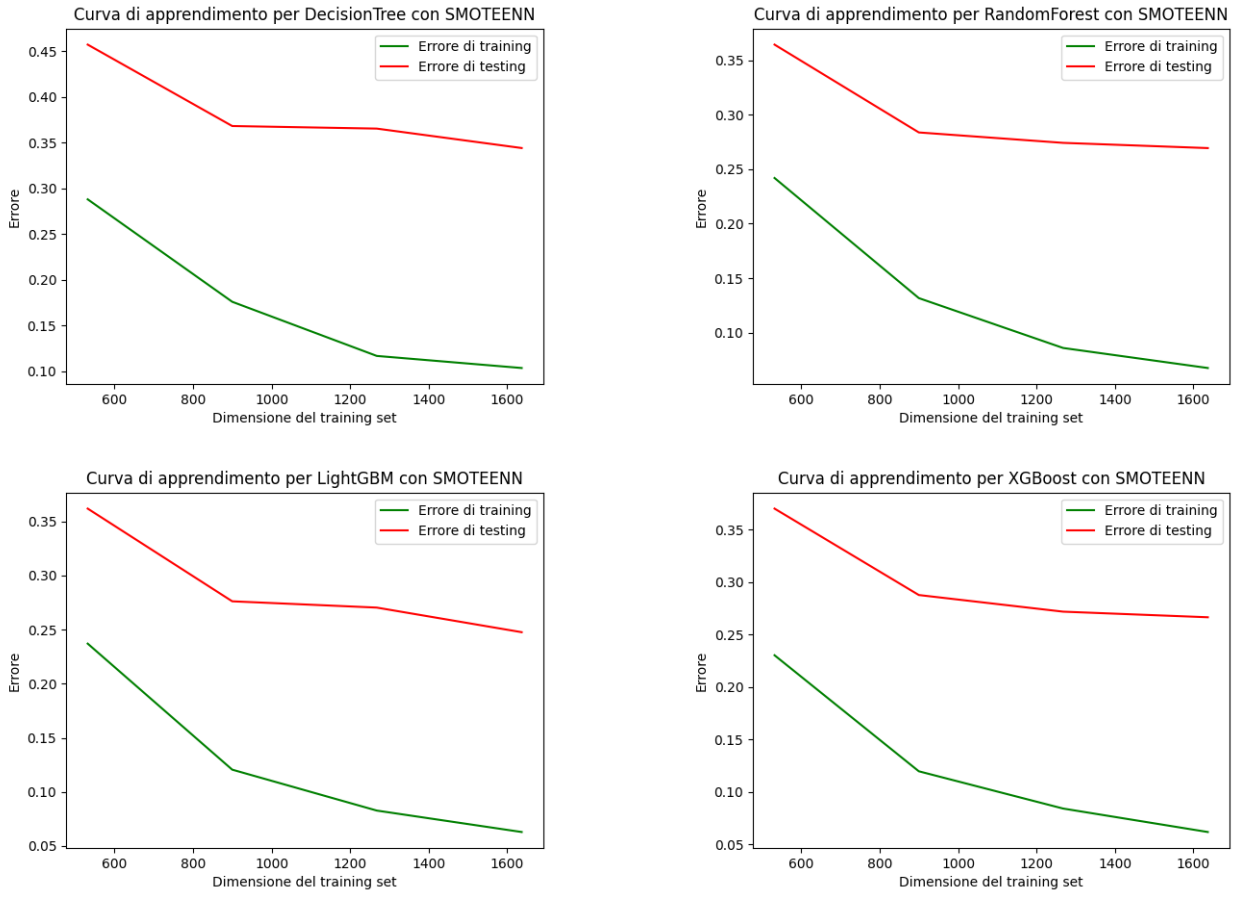


Figura 6: Curve di apprendimento.

Modello	Train Error Std	Test Error Std
LightGBM	0.00686144408303901	0.02657900599318599
XGBoost	0.007076483753736052	0.025883027531329383
DecisionTree	0.00943035445191201	0.020224378283450403
RandomForest	0.00802437542204398	0.03194225646631045

Tabella 17: Deviazione standard dell'errore per i Modelli

Modello	Train Error Var	Test Error Var
LightGBM	4.707941490467104e-05	0.0007064435595858169
XGBoost	5.007662231689027e-05	0.0006699311141875547
DecisionTree	8.893158508869667e-05	0.00040902547695210025
RandomForest	6.439060091390349e-05	0.0010203077481595517

Tabella 18: Varianza degli Errori per i Modelli

Valutazione. I risultati sono al pari dei primi esperimenti, se non inferiori.
Dalle curve di apprendimento si può facilmente notare come i modelli non siano riusciti a stabilizzarsi, e la varianza e la deviazione standard sono molto alte, segno che i modelli non sono riusciti a generalizzare bene. Il numero di esempi non era sufficiente.

3.6 Sommario

In generale, i risultati avuti sono accettabili. Non vi è un modello che spicca rispetto a tutti, ma ciò che possiamo affermare con certezza è che l'esperimento con SMOTE è quello globalmente andato meglio. I modelli che si sono distinti sono stati:

- l'XGBoost del primo esperimento, simile per risultati, deviazione standard e varianza al Random Forest del primo esperimento ma con una curva di apprendimento migliore;
- il Random Forest del secondo esperimento;
- il Random Forest e l'XGBoost del terzo esperimento con SMOTE, risultati i migliori globalmente;
- l'XGBoost del terzo esperimento con ADASYN;
- l'XGBoost del quarto esperimento con SMOTETomek.

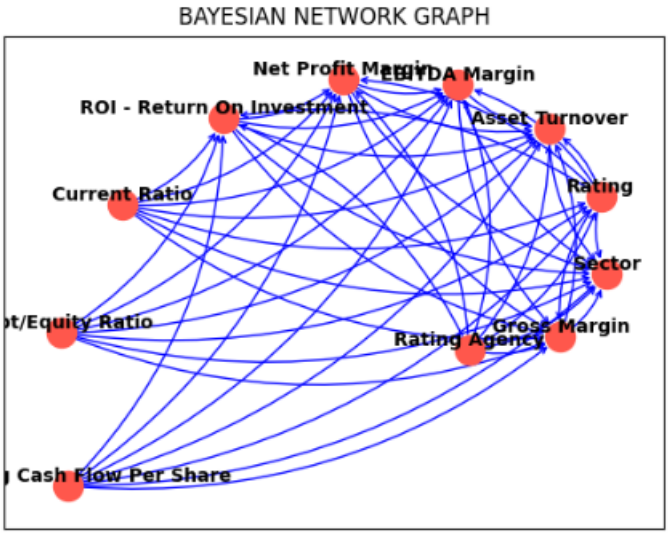
Un possibile approccio futuro può essere quello di combinare i modelli che si sono distinti per verificare se i risultati sono migliorabili.

4 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.

4.1 Apprendimento della struttura

Date le risorse computazionali a disposizione, si è sfruttata la *feature importance* degli esperimenti precedenti per selezionare le variabili da mantenere per apprendere la struttura della rete bayesiana. Le feature che verranno utilizzate sono le seguenti: *Rating Agency, Rating, Sector, Current Ratio, Debt/Equity Ratio, Gross Margin, EBITDA Margin, Net Profit Margin, Asset Turnover, ROI - Return On Investment, Operating Cash Flow Per Share.*
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*:

```
CPT of Rating Agency:
+-------------------------------------------------+-------------+
| Rating Agency(DBRS)                             | 0.0033312   |
+-------------------------------------------------+-------------+
| Rating Agency(Egan-Jones Ratings Company)       | 0.362076    |
+-------------------------------------------------+-------------+
| Rating Agency(Fitch Ratings)                    | 0.0611147   |
+-------------------------------------------------+-------------+
| Rating Agency(HR Ratings de Mexico S.A. de C.V.)| 0.000640615 |
+-------------------------------------------------+-------------+
| Rating Agency(Japan Credit Rating Agency,Ltd.)  | 0.00281871  |
+-------------------------------------------------+-------------+
| Rating Agency(Moody's Investors Service)        | 0.209609    |
+-------------------------------------------------+-------------+
| Rating Agency(Standard & Poor's Ratings Services)| 0.36041    |
+-------------------------------------------------+-------------+
```

Non si è riuscito ad ottenere in tempi ragionevoli la stampa della CPT di una variabile avente dei parents, dato che la rete è molto complessa.

4.2 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.
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.

```
                Rating Agency  Debt/Equity Ratio  \
0  Egan-Jones Ratings Company               4.0

   Operating Cash Flow Per Share  Asset Turnover  EBITDA Margin  \
0                            1.0             2.0            3.0

   Net Profit Margin  ROI - Return On Investment Sector  Current Ratio  \
0                3.0                            3.0  BusEq            0.0

   Gross Margin
0           3.0
  Rating:  0
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
+-----------+---------------+
| Rating    |   phi(Rating) |
+===========+===============+
| Rating(0) |        1.0000 |
+-----------+---------------+
| Rating(1) |        0.0000 |
+-----------+---------------+
| Rating(2) |        0.0000 |
+-----------+---------------+
| Rating(3) |        0.0000 |
+-----------+---------------+
```

Figura 7: Esempio generato e classificato

```
--------- ORA SENZA DEBT/EQUITY RATIO ----------
Finding Elimination Order: : 100%|██████████| 1/1 [00:00<00:00,  2.48it/s]
Eliminating: Debt/Equity Ratio: 100%|██████████| 1/1 [00:00<00:00, 997.69it/s]
+-----------+---------------+
| Rating    |   phi(Rating) |
+===========+===============+
| Rating(0) |        1.0000 |
+-----------+---------------+
| Rating(1) |        0.0000 |
+-----------+---------------+
| Rating(2) |        0.0000 |
+-----------+---------------+
| Rating(3) |        0.0000 |
+-----------+---------------+
```

Figura 8: Esempio generato e classificato senza *Debt/Equity Ratio*

```
                Rating Agency  Debt/Equity Ratio  \
0  Egan-Jones Ratings Company                4.0

   Operating Cash Flow Per Share  Asset Turnover  EBITDA Margin  \
0                            0.0             2.0            2.0

   Net Profit Margin  ROI - Return On Investment Sector  Current Ratio  \
0                3.0                               3.0  Other            0.0

   Gross Margin
0           2.0
  Rating:  3
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
+-----------+---------------+
| Rating    |   phi(Rating) |
+===========+===============+
| Rating(0) |        0.0000 |
+-----------+---------------+
| Rating(1) |        0.0000 |
+-----------+---------------+
| Rating(2) |        0.4000 |
+-----------+---------------+
| Rating(3) |        0.6000 |
+-----------+---------------+
```

Figura 9: Distribuzione di probabilità ampia

```
                Rating Agency  Debt/Equity Ratio  \
0  Egan-Jones Ratings Company                4.0

   Operating Cash Flow Per Share  Asset Turnover  EBITDA Margin  \
0                            1.0             0.0            3.0

   Net Profit Margin  ROI - Return On Investment Sector  Current Ratio  \
0                3.0                               3.0  Enrgy            0.0

   Gross Margin
0           4.0
  Rating:  1
Finding Elimination Order: : : 0it [00:00, ?it/s]
0it [00:00, ?it/s]
+-----------+---------------+
| Rating    |   phi(Rating) |
+===========+===============+
| Rating(0) |        0.2917 |
+-----------+---------------+
| Rating(1) |        0.5000 |
+-----------+---------------+
| Rating(2) |        0.2083 |
+-----------+---------------+
| Rating(3) |        0.0000 |
+-----------+---------------+
```

Figura 10: Distribuzione di probabilità molto ampia

Come è possibile notare la rete è in grado di fare inferenza sulla variabile *Rating* anche se non sono note delle variabili, come nel nostro caso di *Debt/Equity Ratio* in *Figura* 8.
E' anche possibile notare che non sempre la rete riesce ad assegnare una stima esatta come in *Figura* 10. Alcune volte riesce a stimare una probabilità vicina come in *Figura* 9, altre volte invece la distribuzione di probabilità è molto più ampia come in *Figura* 10, riuscendo però comunque a fornire la probabilità più alta alla classe corretta.

4.3 Valutazione

E' possibile 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 0.59, 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 tra le variabili.

4.4 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:

```
infer = VariableElimination(bayesianNetwork)
query_report(infer, variables=['Debt/Equity Ratio'], evidence={'Rating': 3},
             desc='Data la osservazione che una azienda è molto rischiosa qual è la distribuzione di probabilità '
                  'per Debt/Equity Ratio')
query_report(infer, variables=['Debt/Equity Ratio', 'Operating Cash Flow Per Share'], evidence={'Rating': 3},
             desc='Data la osservazione che una azienda è molto rischiosa qual è la distribuzione di probabilità '
                  'congiunta di Debt/Equity Ratio e Operating Cash Flow Per Share')
```

Figura 11: Esempio di query

- Data l'osservazione che una azienda è molto rischiosa qual è la distribuzione di probabilità per *Debt/Equity Ratio*
- Data l'osservazione che una azienda è molto rischiosa qual è la distribuzione di probabilità congiunta di *Debt/Equity Ratio* e *Operating Cash Flow Per Share*

Debt/Equity Ratio	phi(Debt/Equity Ratio)
Debt/Equity Ratio(0.0)	0.0026
Debt/Equity Ratio(2.0)	0.0038
Debt/Equity Ratio(3.0)	0.0017
Debt/Equity Ratio(4.0)	0.9919

Figura 12: Risultato prima query

Operating Cash Flow Per Share	Debt/Equity Ratio	phi(Operating Cash Flow Per Share,Debt/Equity Ratio)
Operating Cash Flow Per Share(0.0)	Debt/Equity Ratio(0.0)	0.0000
Operating Cash Flow Per Share(0.0)	Debt/Equity Ratio(2.0)	0.0001
Operating Cash Flow Per Share(0.0)	Debt/Equity Ratio(3.0)	0.0000
Operating Cash Flow Per Share(0.0)	Debt/Equity Ratio(4.0)	0.0637
Operating Cash Flow Per Share(1.0)	Debt/Equity Ratio(0.0)	0.0026
Operating Cash Flow Per Share(1.0)	Debt/Equity Ratio(2.0)	0.0038
Operating Cash Flow Per Share(1.0)	Debt/Equity Ratio(3.0)	0.0017
Operating Cash Flow Per Share(1.0)	Debt/Equity Ratio(4.0)	0.9273
Operating Cash Flow Per Share(2.0)	Debt/Equity Ratio(0.0)	0.0000
Operating Cash Flow Per Share(2.0)	Debt/Equity Ratio(2.0)	0.0000
Operating Cash Flow Per Share(2.0)	Debt/Equity Ratio(3.0)	0.0000
Operating Cash Flow Per Share(2.0)	Debt/Equity Ratio(4.0)	0.0003
Operating Cash Flow Per Share(4.0)	Debt/Equity Ratio(0.0)	0.0000
Operating Cash Flow Per Share(4.0)	Debt/Equity Ratio(2.0)	0.0000
Operating Cash Flow Per Share(4.0)	Debt/Equity Ratio(3.0)	0.0000
Operating Cash Flow Per Share(4.0)	Debt/Equity Ratio(4.0)	0.0006

Figura 13: Risultato seconda query

Da query di questo tipo possiamo ottenere informazioni utili per trarre delle conclusioni, ad esempio che un'azienda molto rischiosa ha molto probabilmente un *Debt/Equity Ratio* alto, e che quindi tende ad indebitarsi. Dalla seconda query notiamo che se un'azienda è molto rischiosa allora questa ha molto probabilmente un *Debt/Equity Ratio* alto e un *Operating Cash Flow Per Share* basso.

4.5 Sommario

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

5 Conclusioni

E' stato condotto uno studio sull'applicazione di tecniche di apprendimento supervisionato per classificare il *Corporate Credit Rating* delle aziende. E' stata poi 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 state soddisfacenti, e nonostante la poca quantità di dati a disposizione, la rete bayesiana sembra rappresentare bene il dominio.

6 Sviluppi futuri

Oltre a quelli già proposti, si potrebbero considerare i seguenti sviluppi futuri:

- Provare a studiare le singole classi originali, avendo a disposizione più dati o con modelli come le reti neurali.
- Fare uno studio sul *noise* dei rating, dovuto a fattori esterni da quelli utilizzati
- Fare *feature engineering* con altre informazioni del dominio
- Fare *feature selection* utilizzando tecniche più avanzate, e quindi non solo basandosi su correlazioni e conoscenza del dominio
- Sviluppo di un'interfaccia per poter utilizzare il modello in produzione

Riferimenti bibliografici

[1] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. ACM.

[2] Investopedia. Corporate credit rating: What it is, how it works, 2024. Online; accessed 13-February-2024.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[4] Ravi Makwana, Dhruvil Bhatt, Kirtan Delwadia, Agam Shah, and Bhaskar Chaudhury. How to get investment grade rating in the age of explainable ai? *Available at SSRN 4163283*, 2022.

[5] Cuong V Nguyen, Sanjiv R Das, John He, Shenghua Yue, Vinay Hanumaiah, Xavier Ragot, and Li Zhang. Multimodal machine learning for credit modeling. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1754–1759. IEEE, 2021.

[6] David Poole and Alan Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, Cambridge, UK, 3 edition, 2023. Online; accessed 13-February-2024.

[7] David Poole and Alan Mackworth. *Artificial Intelligence: Foundations of Computational Agents*. Cambridge University Press, Cambridge, UK, 3 edition, 2023. Online; accessed 13-February-2024.

[8] Il Fatto Quotidiano. Moody's multata per i suoi conflitti di interessi. assegnati rating a società in cui i soci dell'agenzia avevano partecipazioni oltre il 10percento, 2024. Online; accessed 13-February-2024.

[9] Wikipedia. Kappa di cohen — wikipedia, l'enciclopedia libera, 2019. [Online; in data 14-febbraio-2024].