Analisi delle Performance Scolastiche

Componenti del gruppo:

Giuditta Izzo, [MAT.758357], g.izzo2@studenti.uniba.it

Link GitHub: progetto

A.A. 2023-2024

Indice

1. Introduzione

L'obiettivo di questo progetto è sviluppare un modello in grado di prevedere le performance di uno studente in un esame non ancora svolto , sfruttando informazioni storiche relative al percorso scolastico dello studente.

Requisiti funzionali

Il progetto prevede l'applicazione di tecniche di apprendimento supervisionato e reti bayesiane, utilizzando Python e le sue librerie dedicate all'analisi dei dati e all'apprendimento automatico.

Python: Versione 3.11

IDE utilizzato: PyCharm

Librerie utilizzate:

- **pandas**: Per la manipolazione e analisi dei dati (es. caricamento, trasformazione e pulizia del dataset).
- **numpy**: Per calcoli numerici e gestione di array.
- **matplotlib**: Per la creazione di grafici e visualizzazioni.
- **networkx**: Per la creazione e visualizzazione di grafi, come la struttura di una rete bayesiana.
- **scikit-learn**: Per la creazione di modelli di machine learning, la validazione incrociata e la valutazione delle performance.
- **imblearn**: Per il bilanciamento dei dataset tramite tecniche di sovracampionamento.
- **pgmpy**: Per la creazione, gestione e inferenza su reti bayesiane.

Installazione e avvio

Aprire il progetto con l'IDE preferito. Avviare il programma partendo dal file main.py.

2. Preprocessing del Dataset

Il dataset utilizzato per questa analisi è il 'Student Performance Factors', disponibile su Kaggle, che contiene dati dettagliati sulle variabili che influenzano i risultati accademici, tra cui fattori personali, familiari e scolastici.
Dataset Link: https://www.kaggle.com/datasets/lainguyn123/student-performance-factors.

Le features nel dataset sono le seguenti:

- **Hours_Studied**: Indica il numero di ore dedicate allo studio da uno studente in una settimana. Questo dato rappresenta un fattore chiave per analizzare l'impegno scolastico e può variare da un minimo di 1 ora a un massimo di 44 ore settimanali.
- **Attendance**: Rappresenta la percentuale di lezioni frequentate dallo studente rispetto al totale. Questo valore, compreso tra il 60% e il 100%, evidenzia il livello di partecipazione scolastica.
- **Parental_Involvement**: Misura il livello di coinvolgimento dei genitori nell'educazione del figlio. I livelli sono classificati in "Low"(Basso), "Medium"(Medio) e "High"(Alto), fornendo un'idea dell'importanza attribuita alla supervisione e al supporto familiare.
- **Access_to_Resources**: Descrive la disponibilità di risorse educative. Anche questa variabile è suddivisa in livelli ("Low", "Medium", "High"), riflettendo la qualità dell'ambiente di apprendimento.

- **Extracurricular_Activities**: Indica se lo studente partecipa o meno ad attività extracurriculari. I valori possibili sono "Yes" e "No".
- **Sleep_Hours**: Specifica il numero medio di ore di sonno per notte. Un fattore importante per la salute e le performance cognitive varia tra 4 e 10 ore.
- **Previous_Scores**: Riporta il punteggio dell'ultimo esame. I valori sono compresi tra 50 e 100, fornendo una misura delle performance accademiche passate.
- **Motivation_Level**: Valuta il livello di motivazione dello studente nei confronti degli studi, classificato come "Low", "Medium" o "High".
- **Internet_Access**: Indica se lo studente ha accesso a Internet. I valori possibili sono "Yes" e "No", evidenziando il divario tecnologico.
- **Tutoring_Sessions**: Rappresenta il numero di sessioni di tutoraggio frequentate al mese, con un range da 0 a 8. Questo valore è utile per misurare il supporto aggiuntivo ricevuto.
- **Family_Income**: Classifica il livello di reddito familiare in "Low", "Medium" e "High", fornendo informazioni sul contesto socioeconomico dello studente.
- **Teacher_Quality**: Valuta la qualità degli insegnanti, classificata in "Low", "Medium" e "High". Questo dato riflette la percezione della competenza e dell'efficacia degli insegnanti.
- **School_Type**: Specifica il tipo di scuola frequentata, con due opzioni: "Public" (pubblica) o "Private" (privata).
- **Peer_Influence**: Misura l'influenza dei pari sulle performance accademiche dello studente. I valori possibili sono "Positive"(Positiva), "Neutral"(Neutrale) e "Negative"(Negativa).
- **Physical_Activity**: Indica il numero medio di ore dedicate ad attività fisiche settimanali, con un range da 0 a 6 ore. Questo dato rappresenta l'importanza data alla salute fisica.
- **Learning_Disabilities**: Specifica se lo studente presenta disabilità di apprendimento. I valori possibili sono "Yes" e "No".
- **Parental_Education_Level**: Descrive il livello più alto di istruzione raggiunto dai genitori, suddiviso in "High School"(Scuola superiore), "College"(Istruzione universitaria di primo livello) e "Postgraduate" (Studi post-laurea).
- **Distance_from_Home**: Indica la distanza tra la casa dello studente e la scuola. Può essere classificata in "Near" (vicino), "Moderate" (moderata) e "Far" (lontana).
- **Gender**: Specifica il genere dello studente, con due opzioni: "Male" (maschio) e "Female" (femmina).
- **Exam_Score**: Rappresenta il punteggio finale ottenuto dallo studente all'esame, con un range compreso tra 54 e 101.

Preprocessing del dataset

Il preprocessing dei dati è stato progettato per migliorare la qualità del dataset e garantire performance ottimali per i modelli. Di seguito, riportiamo una descrizione dettagliata delle modifiche applicate, le ragioni dietro ogni scelta e l'adattamento specifico per i vari algoritmi.

Durante il preprocessing sono state applicate le seguenti trasformazioni:

1) Eliminazione delle righe con valori nulli:
   Motivazione: Le righe contenenti valori nulli sono state eliminate dal dataset. Sebbene si potessero applicare altre strategie, come la sostituzione dei valori nulli (imputazione), l'eliminazione è stata scelta per garantire che i modelli lavorassero su dati completi e accurati.

Inoltre, la dimensione del dataset era sufficiente a tollerare la perdita di alcune righe senza compromettere la rappresentatività.

2) Encoding delle variabili categoriche:
   Motivazione: L'encoding delle variabili categoriche è fondamentale poiché i modelli di machine learning non possono elaborare stringhe. Le categorie sono state convertite in valori numerici, con mappature specifiche per ciascuna variabile:
   a. Variabili binarie (es. "Yes"/"No") → 1 e 0.
   b. Variabili ordinali (es. "Low"/"Medium"/"High") → 0, 1, 2.

3) Exam_Score ridotto a intervalli:
   Motivazione: Per semplificare l'output e renderlo compatibile con modelli come Decision Tree e Reti Bayesiane, "Exam_Score" è stato discretizzato in tre intervalli numerici. Abbiamo usato direttamente i valori 0, 1 e 2.

4) Standardizzazione delle feature numeriche (solo per Logistic Regression):
   Motivazione: La standardizzazione è stata utilizzata per Logistic Regression perché questo modello è sensibile alla scala delle feature. La standardizzazione (media = 0, deviazione standard = 1) è preferita alla normalizzazione in quanto mantiene la distribuzione della variabile. Senza standardizzazione, feature con scale più grandi potrebbero dominare.
   a. Colonne standardizzate: "Hours_Studied", "Attendance", "Previous_Scores", "Tutoring_Sessions", "Physical_Activity".

5) Raggruppamento delle feature numeriche in intervalli (solo per Bayesian Network):
   Motivazione: Le Reti Bayesiane richiedono variabili discrete per calcolare le probabilità condizionate. Le feature numeriche continue sono state discretizzate in tre intervalli utilizzando lo stesso approccio usato per "Exam_Score".
   a. Colonne raggruppate: "Hours_Studied", "Attendance", "Previous_Scores", "Tutoring_Sessions", "Physical_Activity".

Per ridurre le feature numeriche in intervalli e stato utilizzato il seguente approccio:

```
def group_target_intervals(df, column):  6 usages  new *
    """
    Raggruppa i valori di una colonna in intervalli specificati.

    :param df: DataFrame
    :param column: Nome della colonna da trasformare
    :return: DataFrame trasformato
    """
    discretizer = KBinsDiscretizer(n_bins=3,encode='ordinal', strategy='uniform')
    df[column] = discretizer.fit_transform(df[[column]])

    return df
```

Il dataset è stato diviso in training set e test set con percentuale 80-20.

3. Apprendimento Supervisionato

L'apprendimento supervisionato è una delle branche principali dell'apprendimento automatico. Questo approccio si basa sull'utilizzo di dati etichettati, dove ogni esempio è accompagnato da features di input e dall'etichetta associata (output). L'obiettivo principale è quello di addestrare un modello a prevedere correttamente l'etichetta per nuovi esempi mai visti.
In questo progetto, l'apprendimento supervisionato è stato impiegato per prevedere il valore ordinalizzato dell'Exam Score (0, 1, 2) per ciascuno studente, sulla base di variabili come ore di studio, partecipazione scolastica e altre caratteristiche correlate.

Classificazione nel contesto del progetto

L'apprendimento supervisionato applicato in questo progetto ricade nella classificazione, in cui l'output da prevedere è discreto e rappresentato da tre classi ordinali:

0: Punteggio basso.
1: Punteggio medio.
2: Punteggio alto.

Il modello impara a distinguere tra queste classi sulla base delle caratteristiche fornite.

Processo di apprendimento supervisionato

Il processo è stato articolato nelle seguenti fasi fondamentali:

- *Scelta degli iper-parametri*: Determinazione dei parametri del modello.
- *Addestramento*: Il modello è stato allenato sui dati etichettati per apprendere la relazione input-output.
- *Valutazione*: I modelli sono stati testati su un dataset separato per misurare le loro prestazioni.

Modelli applicati

Per la previsione delle classi di Exam Score, sono stati adottati tre modelli di apprendimento supervisionato:

- *Decision Tree:*
  Un albero decisionale suddivide iterativamente i dati in sottoinsiemi basati su condizioni sulle variabili di input. Ogni nodo dell'albero rappresenta una condizione logica, mentre le foglie corrispondono alle classi finali previste. Questo modello è semplice da interpretare e fornisce una chiara visualizzazione delle decisioni prese.

- *Random Forest:*
  La Random Forest è un modello di ensemble composto da più alberi decisionali. Ogni albero viene addestrato su un sottoinsieme casuale dei dati e le sue previsioni sono combinate per produrre l'output finale. Questo approccio riduce il rischio di overfitting rispetto all'uso di un singolo albero.

- *Regressione Logistica Multinomiale:*
  La regressione logistica multinomiale è stata utilizzata per calcolare la probabilità di appartenenza a ciascuna delle tre classi. La funzione softmax converte i risultati in una

distribuzione di probabilità, e la classe con la probabilità più alta è selezionata come output. Questo modello è particolarmente efficace quando i dati sono linearmente separabili.

Scelta degli iper-parametri

Gli iper-parametri sono i parametri di un modello di apprendimento automatico che devono essere definiti prima dell'inizio dell'addestramento. A differenza dei parametri interni, gli iper-parametri non vengono appresi direttamente dai dati. Essi influenzano significativamente le prestazioni del modello e determinano la sua capacità di generalizzare su nuovi dati. La selezione degli iper-parametri è quindi una delle fasi più cruciali del processo di apprendimento.

Per scegliere gli iper-parametri, in questo progetto è stata utilizzata la tecnica della K-Fold Cross Validation (CV) combinata con una GridSearch.
In questo approccio:

- Il dataset è stato suddiviso in K sottoinsiemi(fold).
- Il modello è stato addestrato su K-1 sottoinsiemi e testato sull'ultimo.
- Il processo viene ripetuto K volte, alternando il fold di test, permettendo di valutare le prestazioni del modello su dati diversi.

Le combinazioni di iper-parametri che ottenevano le migliori metriche di valutazione (es. accuratezza) sono state selezionate.

Parametri considerati e ottimizzati

- ***Decision Tree***
    - Criterion: Determina la misura utilizzata per valutare la qualità dello split nei nodi. Valori possibili:
        - *gini*: Misura la purezza dello split, ovvero quanto spesso un elemento viene classificato in modo sbagliato.
        - *entropy*: Misura la quantità di disordine nei dati, massimizzando l'informazione guadagnata.
        - *log_loss*: Una metrica che usa la log loss (o perdita logaritmica) per valutare la qualità dello split, valuta quanto accuratamente il modello stima le probabilità delle classi, penalizzando severamente previsioni confidenti ma errate.
    - Splitter: Indica la strategia per scegliere la suddivisione nei nodi. È stato utilizzato il valore predefinito best per ottenere lo split ottimale. L'albero valuta tutti i possibili punti di split e sceglie quello che massimizza il criterio di split (ad es. Gini, Entropy, ecc.)
    - Max_depth: Indica la profondità massima dell'albero.
    - Min_samples_split: Numero minimo di esempi necessari per consentire uno split.

- Min_samples_leaf: Numero minimo di esempi richiesti per una foglia.

```
decision_tree_parameters = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10,20],
    'min_samples_leaf': [1, 2, 5, 10, 20],
    'splitter': ['best']

}
```

- **RandomForest**
  La Random Forest utilizza gli stessi iper-parametri dei singoli Decision Tree, con l'aggiunta di:

  - n_estimators: Numero di alberi nella foresta. Un valore maggiore aumenta la robustezza del modello ma aumenta anche il tempo di calcolo.

```
random_forest_parameters = {
    'criterion': ['gini', 'entropy','log_loss'],
    'max_depth': [None, 5, 10],
    'n_estimators': [10, 20, 50],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 5, 10, 20],
}
```

- **Regressione Logistica Multinomiale**
  La regressione logistica multinomiale è stata implementata con un'attenzione particolare agli iper-parametri chiave, che influenzano direttamente le sue capacità predittive e la robustezza:

  - Penalty: Stabilisce il tipo di regolarizzazione per ridurre la complessità del modello e prevenire overfitting.
    - *l2*: Penalizza i pesi in base alla loro grandezza, favorendo una distribuzione più uniforme. Questo metodo è più indicato per dataset con molte feature rilevanti.

  - C: Rappresenta l'inverso della forza di regolarizzazione ($1/\lambda$). Valori di C più piccoli indicano una regolarizzazione più forte, utile per prevenire l'overfitting, ma possono anche ridurre la capacità del modello di catturare le sfumature nei dati. Valori elevati, invece, rendono il modello più flessibile, aumentando il rischio di overfitting su dati di training rumorosi.

  - Max_iter: Questo parametro definisce il numero massimo di iterazioni dell'algoritmo di ottimizzazione. È essenziale per garantire che l'algoritmo converga. Valori insufficienti possono portare a modelli non ottimali.
  - Solver: Indica l'algoritmo utilizzato per ottimizzare la funzione di costo. Le scelte principali includono:

- *lbfgs*: Acronimo di "Limited-memory Broyden–Fletcher–Goldfarb–Shanno," è un algoritmo di ottimizzazione numerica basato su metodi di derivata seconda, specificamente progettato per problemi di ottimizzazione con vincoli di memoria. È particolarmente efficace per dataset di medie dimensioni, garantendo una buona velocità di convergenza.
- *liblinear*: Utilizza tecniche di programmazione lineare ed è particolarmente efficiente per problemi con poche feature o un basso numero di campioni.

```
logistic_regression_parameters = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100],
    'penalty': ['l2'],
    'solver': ['liblinear', 'lbfgs'],
    'max_iter': [100000, 150000],

}
```

In questo progetto, la metrica utilizzata per valutare i modelli durante la K-Fold Cross Validation è stata il punteggio F1 Macro, una misura che estende il concetto di F1 score ai problemi multi-classe. L'F1 Macro calcola il punteggio F1 per ciascuna classe in modo indipendente e poi ne calcola la media aritmetica. Questo approccio assegna uguale peso a tutte le classi, indipendentemente dalla loro dimensione, risultando particolarmente utile nei dataset sbilanciati, dove alcune classi potrebbero essere rappresentate in modo sproporzionato. Anche in presenza di un dataset bilanciato tramite oversampling, l'F1 Macro garantisce una valutazione più equa delle prestazioni del modello sulle diverse classi.

```
grid_search_rf = GridSearchCV(
    RandomForestClassifier(random_state = 42),
    random_forest_parameters,
    cv = 5,
    scoring = 'f1_macro',
)
```

```
grid_search_lr = GridSearchCV(
    LogisticRegression(random_state = 42),
    logistic_regression_parameters,
    cv = 5,
    scoring = 'f1_macro',
)
```

```
    grid_search_dc = GridSearchCV(
        DecisionTreeClassifier(random_state = 42),
        decision_tree_parameters,
        cv = 5,
        scoring = 'f1_macro',
    )
```

Nel mio caso i parametri che sono stati restituiti sono:

	Modello	Parametro	Valore
0	Decision Tree	criterion	gini
1	Decision Tree	max_depth	None
2	Decision Tree	min_samples_split	2
3	Decision Tree	min_samples_leaf	5
4	Random Forest	criterion	entropy
5	Random Forest	max_depth	None
6	Random Forest	n_estimators	50
7	Random Forest	min_samples_split	2
8	Random Forest	min_samples_leaf	1
9	Logistic Regression	C	10
10	Logistic Regression	penalty	l2
11	Logistic Regression	solver	liblinear
12	Logistic Regression	max_iter	100000

Fase di addestramento e test

```
dt, p_dt = optimize_decision_tree(x_train, y_train)
rf, p_rf = optimize_random_forest(x_train, y_train)

x_resampled = standardize_data(x_train, colums_to_standardize: ['Hours_Studied', 'Attendance', 'Sleep_Hours',
                                'Previous_Scores', 'Tutoring_Sessions', 'Physical_Activity'])
lr, p_lr = optimize_logistic_regression(x_resampled, y_train)

generate_model_parameters_table(p_dt, p_rf, p_lr)

scoring_metrics = ['accuracy', 'precision_macro', 'recall_macro', 'f1_macro']
results_dt = {}
results_rf = {}
results_lr = {}
for scoring_metric in scoring_metrics:
    print(f"\n--- Evaluating {scoring_metric} ---")
    dt_score = evaluate_model(dt, x_test, y_test, scoring_metric)
    rf_score = evaluate_model(rf, x_test, y_test, scoring_metric)
    lr_score = evaluate_model(lr, x_test, y_test, scoring_metric)

    results_dt[scoring_metric] = dt_score
    results_rf[scoring_metric] = rf_score
    results_lr[scoring_metric] = lr_score
```

Problema Rilevato: UndefinedMetricWarning

Otteniamo il seguente errore:

```
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
```

Durante l'addestramento è emerso un avviso di tipo UndefinedMetricWarning. Questo errore è dovuto alla presenza di classi con pochi campioni, un problema che si verifica quando il modello non riesce a predire alcun campione appartenente a una determinata classe. Di conseguenza:

- Metriche di valutazione risultano non definite (pari a 0.0).
- Il modello fatica a generalizzare correttamente, specialmente per le classi minoritarie, riducendo drasticamente le performance complessive.

Distribuzione delle Classi Target

La distribuzione delle classi del target Exam_Score è la seguente:

```
Distribuzione delle classi di Exam_Score nel train set:
Exam_Score
0.0    3984
1.0     773
2.0      26
```

È una distribuzione fortemente sbilanciata.

Influenza degli Outlier

Gli outlier possono influenzare il metodo di discretizzazione applicato con KBinsDiscretizer, soprattutto quando si utilizza la strategia "uniform", come nel nostro caso.

La strategia "uniform" suddivide l'intervallo dei valori della variabile target in n_bins intervalli di ampiezza uguale. Tuttavia, questa suddivisione è determinata dai valori minimi e massimi della variabile.

Gli outlier, essendo valori estremamente alti o bassi rispetto alla distribuzione dei dati, ampliano artificialmente l'intervallo complessivo della variabile "Exam_Score".
Di conseguenza finiscono nei bordi degli intervalli (primo o ultimo bin), mentre i valori centrali vengono raggruppati nell'intervallo centrale. Questo porta a una distribuzione sbilanciata anche dopo la discretizzazione.

Per evitare che gli outlier influenzino negativamente la discretizzazione con KBinsDiscretizer, abbiamo deciso di rimuovere gli outlier utilizzando un metodo robusto come IQR (Interquartile Range) prima di applicare il KBinsDiscretizer.

Gestione degli outlier:

Gli outlier sono stati calcolati utilizzando l'Interquartile Range (IQR), un metodo robusto che non richiede assunzioni sulla distribuzione dei dati:

$$\boldsymbol{IQR} = Q3 - Q1$$

Dove Q1 = primo quartile (25° percentile) e Q3 = terzo quartile (75° percentile).
Limiti:

$$Limite\ inferiore = Q1 - 1.5 * IQR$$
$$Limite\ superiore = Q3 + 1.5 * IQR$$

I valori al di fuori di questi limiti sono considerati outlier.

Il metodo adottato per la Gestione degli Outlier è l'**eliminazione delle righe con outlier**.
Colonne analizzate: "Hours_Studied", "Tutoring_Sessions", "Exam_Score".

```
Il numero di outlier nella colonna '['Hours_Studied']' è: Hours_Studied    40
dtype: int64
Il numero di outlier nella colonna '['Attendance']' è: Attendance    0
dtype: int64
Il numero di outlier nella colonna '['Sleep_Hours']' è: Sleep_Hours    0
dtype: int64
Il numero di outlier nella colonna '['Previous_Scores']' è: Previous_Scores    0
dtype: int64
Il numero di outlier nella colonna '['Tutoring_Sessions']' è: Tutoring_Sessions    423
dtype: int64
Il numero di outlier nella colonna '['Physical_Activity']' è: Physical_Activity    0
dtype: int64
Il numero di outlier nella colonna '['Exam_Score']' è: Exam_Score    103
dtype: int64
```

A seguito dell'eliminazione degli outliner, eseguita sull'intero dataset, la distribuzione sul test set è la seguente:

```
Distribuzione delle classi di Exam_Score nel train set:
Exam_Score
0.0     988
1.0    2394
2.0     995
```

L'eliminazione degli outlier ha portato a una significativa riduzione del numero di campioni nella classe 0.0 e un lieve bilanciamento delle classi.
La nuova distribuzione risulta più equilibrata, permettendo ai modelli di apprendimento supervisionato di apprendere meglio le caratteristiche di ciascuna classe, inclusa la classe 2, precedentemente minoritaria.

Risultati Dopo la Gestione degli Outlier
A seguito dell'eliminazione degli outlier i modelli sono stati ricalibrati utilizzando gli stessi parametri di configurazione.

I parametri che sono stati restituiti sono:

Exam score trasformato a intevalli

	Modello	Parametro	Valore
0	Decision Tree	criterion	gini
1	Decision Tree	max_depth	10
2	Decision Tree	min_samples_split	20
3	Decision Tree	min_samples_leaf	2
4	Random Forest	criterion	entropy
5	Random Forest	max_depth	None
6	Random Forest	n_estimators	50
7	Random Forest	min_samples_split	5
8	Random Forest	min_samples_leaf	1
9	Logistic Regression	C	10
10	Logistic Regression	penalty	l2
11	Logistic Regression	solver	lbfgs
12	Logistic Regression	max_iter	100000

La fase di addestramento e test è stata ripetuta.

Fase di addestramento e test

Nessun errore si è presentato.

Valutazione delle prestazioni

La valutazione delle prestazioni del modello è una fase cruciale per comprendere quanto esso riesca a generalizzare su dati non visti.
Per valutare le performance del modello, sono state utilizzate le seguenti metriche:

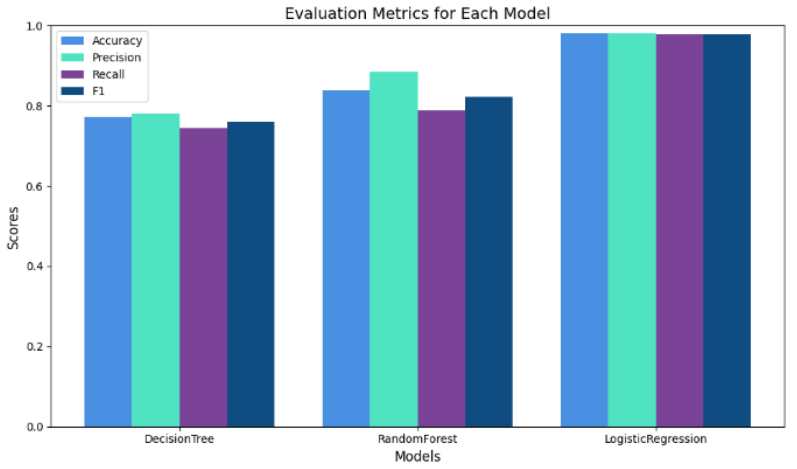
- **Accuracy**: Questa metrica rappresenta il rapporto tra il numero di previsioni corrette e il numero totale di previsioni effettuate.

- **Precision Macro**: La precisione macro è la media delle precisioni calcolate su ciascuna classe. La precisione per una classe specifica si calcola come il rapporto tra il numero di esempi correttamente classificati in quella classe e il numero totale di esempi classificati in quella classe (inclusi i falsi positivi).

- **Recall Macro**: La recall macro è la media dei valori di recall calcolati per ogni classe. La recall per una classe è definita come il rapporto tra il numero di esempi correttamente classificati in quella classe e il numero totale di esempi effettivamente appartenenti a quella classe (inclusi i falsi negativi).

- **F1 Macro**: La F1 macro è la media armonica delle F1 score calcolate per ciascuna classe. La metrica F1 score bilancia precision e recall, risultando elevata solo quando entrambi i valori sono alti. La versione macro della F1 score garantisce che tutte le classi siano considerate in modo equo, indipendentemente dalla loro rappresentazione nel dataset.

Risultati



Analisi dei risultati

Dai risultati emergono alcune osservazioni generali sulle prestazioni dei modelli analizzati:

- **Decision Tree**:

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.78      0.69      0.73       333
         1.0       0.76      0.84      0.80       779
         2.0       0.81      0.70      0.75       347

    accuracy                           0.77      1459
   macro avg       0.78      0.75      0.76      1459
weighted avg       0.77      0.77      0.77      1459
```

Accuracy intorno al 77%.
Ad esempio, per la classe 0 si osserva una precisione di circa 0.78, recall di circa 0.69 e f1-score di circa 0.73. I valori medi (macro) sono intorno a 0.78 (precisione), 0.75 (recall) e 0.76 (f1-score).
Questo indica che l'albero decisionale cattura alcuni pattern ma sbaglia in una percentuale significativa di casi.

- **Random Forest**:

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.94      0.71      0.81       333
         1.0       0.79      0.96      0.86       779
         2.0       0.92      0.70      0.79       347

    accuracy                           0.84      1459
   macro avg       0.88      0.79      0.82      1459
weighted avg       0.86      0.84      0.83      1459
```

Accuracy intorno all'84%.
Il Random Forest mostra prestazioni migliori rispetto al Decision Tree, con valori medi più alti (ad esempio, precisione macro intorno a 0.88), anche se per alcune classi il recall può essere inferiore.

- **Logistic Regression**:

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      0.98      0.98       333
         1.0       0.98      0.98      0.98       779
         2.0       0.99      0.97      0.98       347

    accuracy                           0.98      1459
   macro avg       0.98      0.98      0.98      1459
weighted avg       0.98      0.98      0.98      1459

Accuracy: 0.98
```

Accuracy intorno al 98%
l modello di Logistic Regression mostra prestazioni quasi perfette, con precisione, recall e f1-score vicini a 0.98 per ogni classe.
Questi risultati indicano che la Logistic Regression è in grado di classificare il set di test con grande successo.

Considerazioni Generali
La Logistic Regression mostra performance quasi perfette. Il Random Forest si comporta bene, mentre il Decision Tree, da solo, ha prestazioni inferiori.

Valutazione delle prestazioni con K-Fold Cross Validation e analisi dell'overfitting

In precedenza, abbiamo valutato le prestazioni tramite un test set, ma per avere una stima più affidabile del modello e individuare eventuali problematiche come l'overfitting, abbiamo adottato la K-Fold Cross Validation.
Questo approccio consente di utilizzare diverse suddivisioni del dataset, riducendo il rischio di sovra- o sotto-stima delle performance e offrendo un quadro più completo del

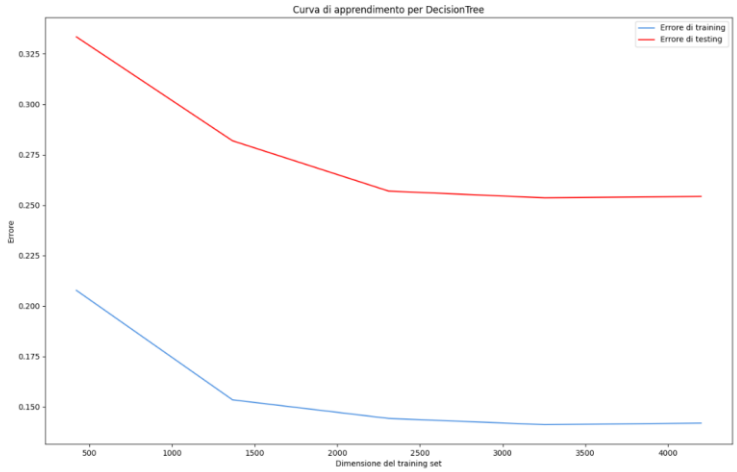
comportamento del modello. La K-Fold viene applicata su ognuna delle suddivisioni del dataset, ovvero per ogni specifica dimensione del training set scelta.
Le *curve di apprendimento* rappresentano graficamente l'andamento delle performance del modello al variare della dimensione del training set. In particolare, viene tracciata una curva per l'errore di training e una per l'errore di validazione (test). Queste curve evidenziano come il modello si comporta quando viene addestrato con quantità sempre maggiori di dati:

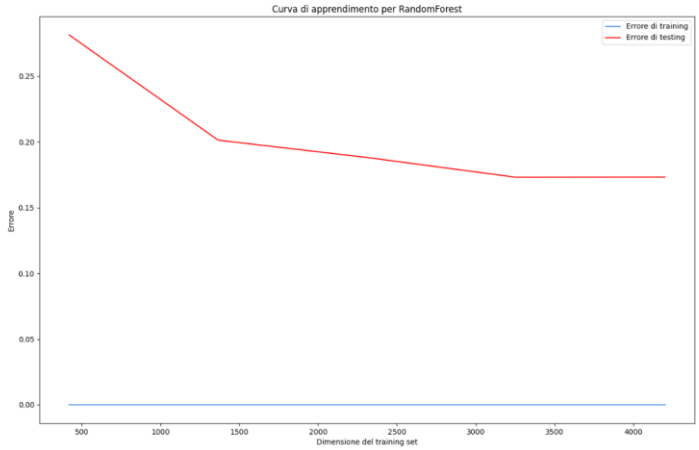
- *Curva di training*: Mostra l'errore medio ottenuto sul sottoinsieme di dati utilizzato per l'addestramento.
- *Curva di test*: Rappresenta l'errore medio calcolato sul set di validazione. In generale, man mano che aumenta la quantità di dati per l'addestramento, il modello riesce a generalizzare meglio e l'errore sul test tende a diminuire.

Per ogni suddivisione del dataset, la K-Fold Cross Validation calcola l'F1 macro.
L'analisi comparata delle due curve fornisce un'indicazione chiara del grado di overfitting:

- Un divario significativo tra l'errore di training (molto basso) e l'errore di test (relativamente alto) indica che il modello si adatta eccessivamente ai dati di addestramento, compromettendo la capacità di generalizzazione.
- Se le due curve convergono verso valori simili, significa che il modello generalizza bene, bilanciando l'ottimizzazione sia sul training che sul test.

L'overfitting rappresenta una delle sfide principali nell'apprendimento automatico. Garantire che il modello generalizzi bene significa ottimizzare non solo le prestazioni sui dati di addestramento, ma anche la sua capacità di fare previsioni affidabili su dati mai visti.





Dai risultati delle curve di apprendimento emergono alcune osservazioni generali sulle prestazioni dei modelli analizzati:

- **Decision Tree**:
  - *Errore di Training e Testing*:
    La curva di apprendimento del Decision Tree mostra che l'errore di training è vicino a zero, suggerendo che il modello è altamente adatto ai dati di training. Tuttavia, l'errore di testing è significativamente più alto, indicando una tendenza all'overfitting.
    L'errore di testing diminuisce gradualmente con l'aumentare della dimensione del training set, ma la distanza tra l'errore di training e quello di testing rimane notevole.

- **Random Forest**:
  - *Errore di Training e Testing*:
    La curva di apprendimento mostra un errore di training estremamente basso, vicino a zero, con un errore di testing più basso rispetto al Decision Tree. Ciò suggerisce che il modello ha una capacità migliore di generalizzare.
    Tuttavia, l'errore di testing non raggiunge livelli ottimali, mantenendo una certa distanza rispetto all'errore di training.
    Overfitting nei modelli: Il Decision Tree e il Random Forest mostrano segni di overfitting, seppur con intensità diversa.
    Sarebbe utile esplorare tecniche di bilanciamento come oversampling .

Sovracampionamento con Random Oversampling

Prima del sovracampionamento, la variabile target "Exam_Score" presentava la seguente distribuzione:

- Low (0): 988.
- Medium (1): 2394.
- High (2): 995.

Questo squilibrio porta i modelli a favorire la classe Medium, ignorando le classi Low e High, meno rappresentate.
Per affrontare questo problema è stato scelto il Random Oversampling, una tecnica che aumenta la rappresentatività delle classi minoritarie duplicando in modo casuale i campioni esistenti, senza introdurre nuovi dati o errori.
Dopo l'applicazione del Random Oversampling, le classi Low e High sono state bilanciate, raggiungendo la stessa numerosità della classe Medium.

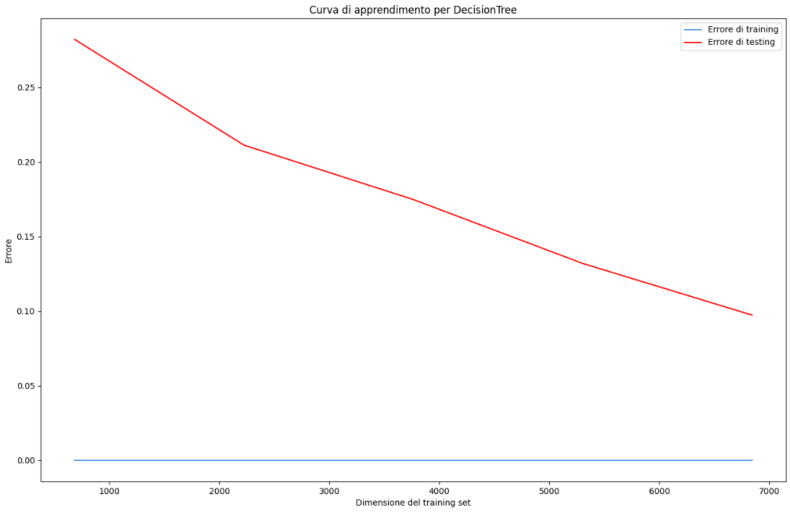
Exam_Score	
0.0	2535
1.0	2535
2.0	2535

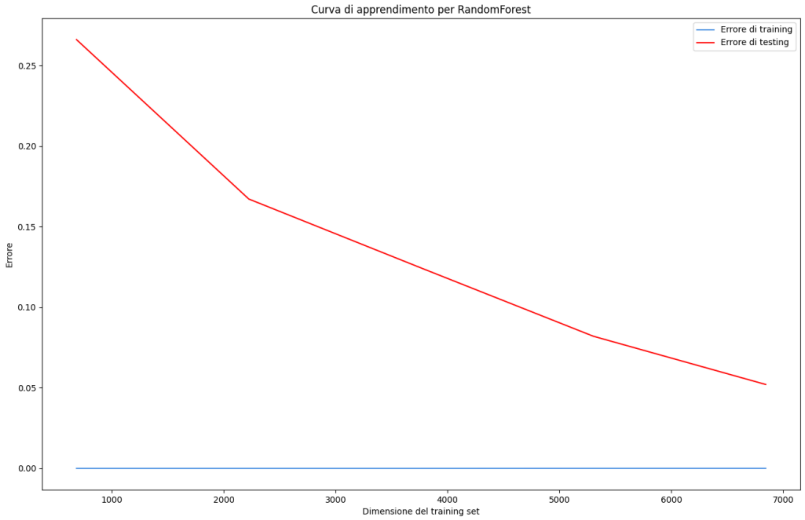
Questo bilanciamento permette ai modelli di apprendimento di generalizzare meglio e trattare tutte le classi in modo equo, migliorando la performance complessiva in fase di test.
È stato scelto il random oversampling poiché abbiamo per la maggior parte features categoriche.

Iper-parametri restituiti

	Modello	Parametro	Valore
0	Decision Tree	criterion	entropy
1	Decision Tree	max_depth	None
2	Decision Tree	min_samples_split	2
3	Decision Tree	min_samples_leaf	1
4	Random Forest	criterion	gini
5	Random Forest	max_depth	None
6	Random Forest	n_estimators	50
7	Random Forest	min_samples_split	2
8	Random Forest	min_samples_leaf	1

Valutazione delle prestazioni





Dai risultati delle curve di emergono alcune osservazioni generali sulle prestazioni dei modelli analizzati:

- **Decision Tree**:
  - *Errore di Training e Testing*:
    La curva di apprendimento del Decision Tree dopo il random oversampling evidenzia una diminuzione dell'errore di testing con l'aumentare dei dati di training. Tuttavia, l'errore di training rimane pressoché nullo, indicando che il modello si adatta perfettamente ai dati di training.
    Questo comportamento potrebbe segnalare una propensione all'overfitting, benché mitigato dal random oversampling che aumenta la rappresentatività del dataset.

Il Decision Tree è migliorato dopo il random oversampling, mostrando una riduzione dell'errore di testing e una migliore generalizzazione. Tuttavia, l'errore di training estremamente basso evidenzia possibile overfitting.

- **Random Forest**:

- *Errore di Training e Testing*:
  La curva di apprendimento del Random Forest dopo il random oversampling mostra un netto miglioramento dell'errore di testing, che si riduce progressivamente con l'aumentare dei dati di training.
  L'errore di training rimane nullo indicando che il modello è ancora incline all'overfitting .

  Considerazioni Generali

  - *Decision Tree*: L'overfitting può essere mitigato regolando parametri come max_depth, min_samples_split, e min_samples_leaf.

  - Random Forest: Si può ulteriormente ottimizzare regolando parametri come n_estimators. Vale la pena affinare i parametri per garantire robustezza e generalizzazione ottimali.

  Ottimizzazione dei Modelli
  Dopo aver analizzato i risultati precedenti, dove le problematiche di overfitting erano ancora evidenti nei modelli Decision Tree e Random Forest, abbiamo modificato i parametri per migliorare la generalizzazione e ridurre la complessità dei modelli.

  Parametri per il Decision Tree

```
decision_tree_parameters = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [5, 10, 15],
    'min_samples_split': [10, 20, 50],
    'min_samples_leaf': [5, 10, 20, 50],
    'splitter': ['best']

}
```

  Questi parametri sono stati scelti per:

- Ridurre la profondità massima (max_depth) e controllare la crescita eccessiva dell'albero.
- Aumentare i campioni minimi richiesti per uno split (min_samples_split) e nelle foglie (min_samples_leaf), rendendo il modello meno sensibile ai rumori nei dati.

Parametri per il Random Forest:

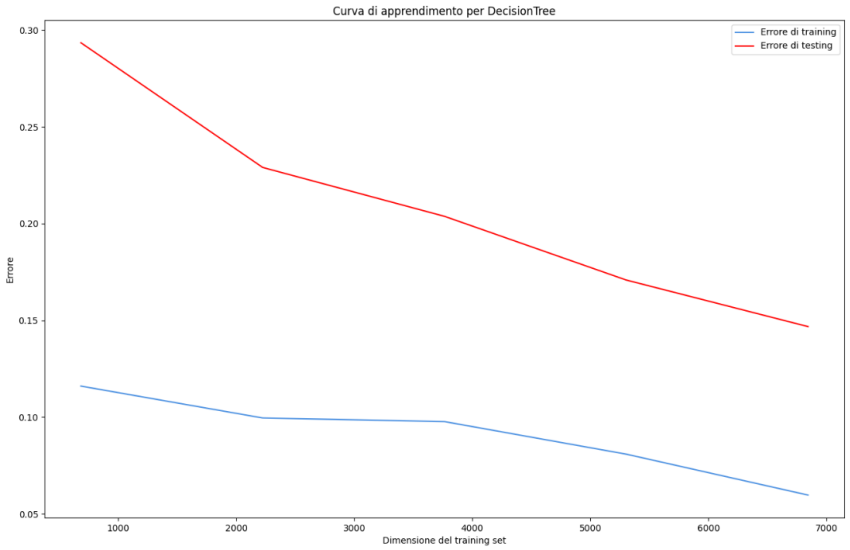
```
random_forest_parameters = {
    'criterion': ['gini', 'entropy','log_loss'],
    'max_depth': [5, 10, 15],
    'n_estimators': [50,100,150],
    'min_samples_split': [10, 20, 50],
    'min_samples_leaf': [5, 10, 20, 50],
}
```

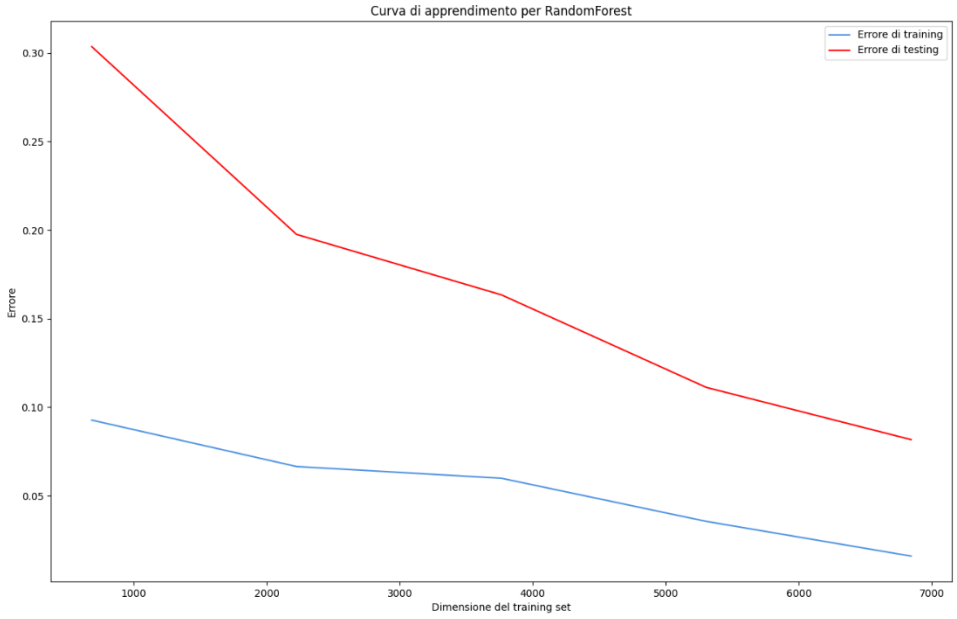
Questi parametri condividono gli stessi valori di max_depth, min_samples_split e min_samples_leaf utilizzati nel Decision Tree. Tuttavia, l'aggiunta del parametro n_estimators viene incrementato per ottenere una maggiore stabilità e robustezza.

Iper-parametri restituiti

	Modello	Parametro	Valore
0	Decision Tree	criterion	entropy
1	Decision Tree	max_depth	15
2	Decision Tree	min_samples_split	10
3	Decision Tree	min_samples_leaf	5
4	Random Forest	criterion	entropy
5	Random Forest	max_depth	15
6	Random Forest	n_estimators	150
7	Random Forest	min_samples_split	10
8	Random Forest	min_samples_leaf	5

Valutazione delle prestazioni





Analisi dei risultati

- **Decision Tree**:
  - *Errore di Training e Testing*:
    La curva di apprendimento mostra una chiara riduzione del divario tra errore di training e di testing rispetto ai risultati precedenti. Questo suggerisce che i parametri ottimizzati, come l'aumento di min_samples_split e min_samples_leaf, insieme alla riduzione di max_depth, hanno mitigato in parte l'overfitting. L'errore di training è aumentato leggermente.

- **Random Forest**:

  - *Errore di Training e Testing*:
    La curva di apprendimento del Random Forest evidenzia una riduzione significativa sia dell'errore di training che di testing. La differenza tra le due curve è ora molto più contenuta rispetto alle analisi precedenti. Questo è dovuto alla scelta di parametri più restrittivi come min_samples_split e min_samples_leaf più elevati, che hanno migliorato la generalizzazione del modello.

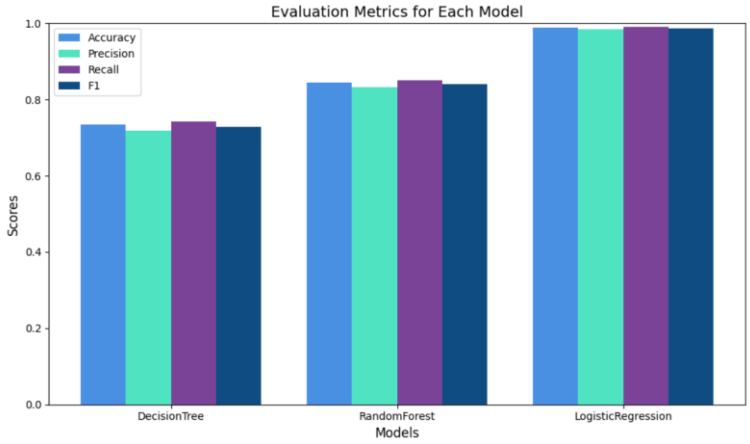
    Il Random Forest ha significativamente ridotto l'overfitting rispetto alle versioni precedenti.

Eliminazione dell'Overfitting

- Decision Tree:
  L'overfitting è stato ridotto.
  La differenza tra errore di training e testing è ancora presente, anche se più contenuta rispetto ai risultati precedenti.
  La riduzione dell'errore di testing e la lieve crescita dell'errore di training indicano una maggiore generalizzazione del modello.

- Random Forest:
  L'overfitting è stato eliminato. La differenza tra errore di training e testing è minima.

Valutazione delle prestazioni su test set



Classification Report:

	precision	recall	f1-score	support
0.0	0.66	0.76	0.71	266
1.0	0.78	0.72	0.75	643
2.0	0.71	0.76	0.73	259
accuracy			0.73	1168
macro avg	0.72	0.74	0.73	1168
weighted avg	0.74	0.73	0.73	1168

Decision Tree:
Nel caso del Decision Tree, le performance sono risultate peggiorate. Questo comportamento è dovuto a una caratteristica intrinseca di questo tipo di modello: i singoli alberi di decisione tendono a non generalizzare bene, essendo molto sensibili al rumore e alle

variazioni nel dataset. Nonostante gli sforzi per migliorare la generalizzazione tramite il bilanciamento dei dati e la modifica dei parametri, il Decision Tree ha continuato a mostrare una tendenza al overfitting, evidenziando così la sua minore robustezza rispetto agli altri modelli.

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.80      0.87      0.83       266
         1.0       0.88      0.84      0.86       643
         2.0       0.82      0.85      0.83       259

    accuracy                           0.85      1168
   macro avg       0.83      0.85      0.84      1168
weighted avg       0.85      0.85      0.85      1168
```

Random Forest:
Il modello di Random Forest ha beneficiato del bilanciamento delle classi ottenuto tramite oversampling. La natura ensemble di Random Forest ha contribuito a ridurre il rischio di overfitting e a migliorare la generalizzazione, portando a prestazioni leggermente migliori.

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.97      1.00      0.99       266
         1.0       0.99      0.98      0.99       643
         2.0       0.99      0.99      0.99       259

    accuracy                           0.99      1168
   macro avg       0.99      0.99      0.99      1168
weighted avg       0.99      0.99      0.99      1168
```

Logistic Regression:
Anche la Logistic Regression ha mostrato un miglioramento. Il bilanciamento del dataset ha reso l'addestramento del modello più affidabile, consentendo una migliore stima dei parametri e, di conseguenza, un incremento della capacità predittiva

4. Reti Bayesiane e Apprendimento Non Supervisionato

Le reti bayesiane sono modelli probabilistici che rappresentano le dipendenze tra variabili attraverso grafi aciclici diretti (DAG). Ogni nodo della rete rappresenta una variabile casuale discreta ,mentre gli archi definiscono relazioni di dipendenza probabilistica tra variabili. L'obiettivo principale è modellare le relazioni causali e utilizzare tali relazioni per ragionamenti probabilistici, come predizioni e inferenza.
Le reti bayesiane sono tradizionalmente progettate per lavorare con variabili discrete, cioè variabili che assumono un numero finito di stati o categorie. Con variabili continue esistono alcune difficoltà pratiche che rendono questa operazione non semplice o inefficiente.

Per costruire le Distribuzioni di Probabilità Condizionata , è necessario che la rete osservi tutti i valori possibili delle variabili. Se ciò non accade, la rete può generare errori durante il processo di inferenza. Il motivo principale di questa limitazione risiede nella natura delle CPD, che rappresentano le probabilità condizionate tra variabili e richiedono la presenza esplicita di ciascun valore nel training set. In assenza di tali osservazioni, la rete non è in grado di calcolare le probabilità per valori mai visti, portando a una mancata previsione.

Questo comportamento evidenzia l'importanza della discretizzazione dei valori continui prima dell'allenamento, per garantire che ogni intervallo rappresentato abbia una corrispondenza esplicita nelle CPD.

Nel nostro progetto, la discretizzazione è stata scelta per semplificare il processo e garantire una gestione efficiente delle variabili continue, preservando la capacità della rete bayesiana di modellare le relazioni tra le variabili.

Prima di costruire la rete, abbiamo trasformato le variabili continue in intervalli discreti utilizzando il metodo KBinsDiscretizer con strategia uniform. La scelta della strategia uniforme permette di suddividere i valori continui in intervalli di uguale ampiezza.
Senza la gestione degli outlier, i valori estremi nelle variabili continue possono distorcere il processo di discretizzazione, portando a bin che non rappresentano accuratamente la distribuzione sottostante, di conseguenza li abbiamo gestiti come per l'apprendimento supervisionato.

Per individuare la struttura ottimale della rete bayesiana, abbiamo utilizzato l'algoritmo Hill Climb Search. Questo algoritmo è un metodo iterativo di ottimizzazione che parte da una struttura iniziale della rete e procede iterativamente migliorandola. La "bontà" della struttura viene valutata tramite una funzione di scoring, in questo caso il K2 score.

Il K2 score misura la probabilità della struttura di una rete bayesiana, data un insieme di dati osservati. Si basa sulla seguente formula probabilistica:

$$P(Struttura \mid Dati) \propto P(Struttura) \cdot P(Dati \mid Struttura)$$

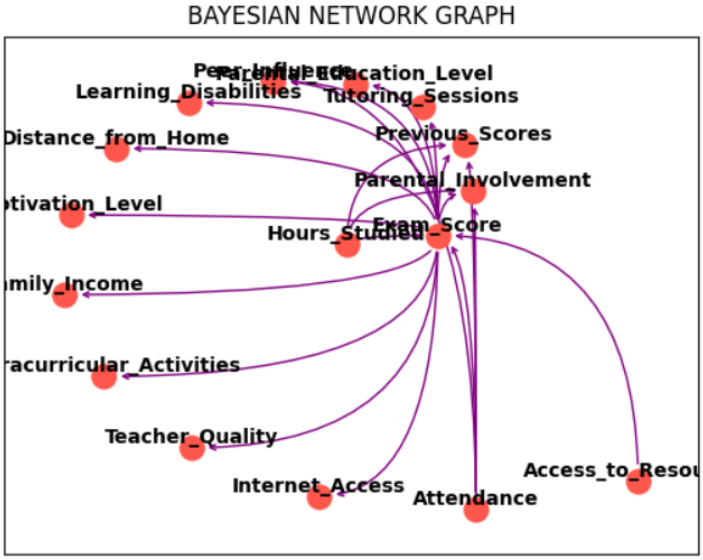
Poiché nella pratica si assume che tutte le strutture abbiano uguale probabilità a priori, la formula viene semplificata a:

$$P(Dati \mid Struttura)$$

In parole semplici, il K2 score valuta quanto è probabile che i dati osservati siano stati generati dalla struttura corrente della rete. Maggiore è il valore del punteggio, migliore sarà la struttura proposta.
Il K2 score richiede variabili discrete per funzionare correttamente.

```
hc=HillClimbSearch(dataSet)
model=hc.estimate(scoring_method='k2score')

model = BayesianNetwork(model.edges())
model.fit(dataSet,estimator=MaximumLikelihoodEstimator,n_jobs=-1)
```



Questi sono esempi di probabilità che la rete è riuscita ad apprendere:

```
CPD per la variabile 'Exam_Score':
+----------------------+-----+-------------------------+
| Access_to_Resources  | ... | Access_to_Resources(2)  |
+----------------------+-----+-------------------------+
| Attendance           | ... | Attendance(2.0)         |
+----------------------+-----+-------------------------+
| Hours_Studied        | ... | Hours_Studied(2.0)      |
+----------------------+-----+-------------------------+
| Exam_Score(0.0)      | ... | 0.0                     |
+----------------------+-----+-------------------------+
| Exam_Score(1.0)      | ... | 0.030927835051546393    |
+----------------------+-----+-------------------------+
| Exam_Score(2.0)      | ... | 0.9690721649484536      |
+----------------------+-----+-------------------------+
=======================================

CPD per la variabile 'Parental_Involvement':
+--------------------------+-----+----------------------+
| Attendance               | ... | Attendance(2.0)      |
+--------------------------+-----+----------------------+
| Exam_Score               | ... | Exam_Score(2.0)      |
+--------------------------+-----+----------------------+
| Hours_Studied            | ... | Hours_Studied(2.0)   |
+--------------------------+-----+----------------------+
| Parental_Involvement(0)  | ... | 0.17704918032786884  |
+--------------------------+-----+----------------------+
| Parental_Involvement(1)  | ... | 0.5311475409836065   |
+--------------------------+-----+----------------------+
| Parental_Involvement(2)  | ... | 0.29180327868852457  |
```

La variabile Exam_Score dipende da altre variabili come:

- Access_to_Resources
- Attendance
- Hours_Studied

Un esempio: P(Exam_Score=2 | Access_to_Resources=2, Attendance=2, Hours_Studied=2) = 0.9690721649484536

Questo significa che, dai dati osservati, se uno studente ha pieno accesso alle risorse (Access_to_Resources=2), un'ottima frequenza (Attendance=2) e ha studiato molto

(Hours_Studied=2), allora c'è circa il 96.9% di probabilità che ottenga il punteggio massimo nell'esame (Exam_Score=2).

La variabile Parental_Involvement dipende invece da:
- Attendance
- Exam_Score
- Hours_Studied

Un esempio: P(Parental_Involvement=1 | Attendance=2, Exam_Score=0, Hours_Studied=0) = 0.5311475409836065

Questo indica che, quando la frequenza è alta (Attendance=2), ma lo studente ha ottenuto un punteggio basso nell'esame (Exam_Score=0) e non ha studiato (Hours_Studied=0), c'è circa il 53.1% di probabilità che il coinvolgimento dei genitori sia medio (Parental_Involvement=1).

Proviamo a predire l'exam score per un esempio randomico. L'esempio randomico viene generato già discretizzato.

```
ESEMPIO RANDOMICO GENERATO --->      Learning_Disabilities  ...  Motivation_Level
0                       0  ...                  1

[1 rows x 15 columns]
PREDIZIONE DEL SAMPLE RANDOM
+-----------------+-------------------+
| Exam_Score      |   phi(Exam_Score) |
+=================+===================+
| Exam_Score(0.0) |            0.6782 |
+-----------------+-------------------+
| Exam_Score(1.0) |            0.3218 |
+-----------------+-------------------+
| Exam_Score(2.0) |            0.0000 |
+-----------------+-------------------+
```

Analisi delle probabilità

- Exam_Score = 0:
  Probabilità: 67.82%
  Questo indica che, date le variabili osservate, è più probabile che lo studente ottenga un punteggio basso nell'esame.
- Exam_Score = 1:
  Probabilità: 32.18%
  C'è una possibilità meno marcata, ma comunque non trascurabile, che lo studente ottenga un punteggio medio.

- Exam_Score = 2:
  Probabilità: 0.00%
  Non ci sono evidenze, dai dati osservati, che suggeriscano che lo studente possa raggiungere il punteggio massimo nell'esame.

5. Sviluppi futuri

Il progetto lascia spazio a diversi miglioramenti e direzioni di sviluppo futuro. Di seguito sono elencate alcune proposte.

Si potrebbero applicare tecniche di feature selection per provare a ridurre la dimensionalità del problema.

Inoltre, si potrebbe integrare il modello in sistemi di e-learning per fornire raccomandazioni personalizzate agli studenti, come piani di studio ottimizzati o suggerimenti per migliorare la motivazione.

Condurre un'analisi approfondita dell'impatto del modello: quanto le predizioni migliorano le decisioni degli educatori o l'efficacia degli interventi mirati sugli studenti.

6. Riferimenti Bibliografici

- Apprendimento supervisionato: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press. 3rd Ed. [Ch.7]
- Apprendimento non supervisionato: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press. 3rd Ed. [Ch.10]
- Ragionamento probabilistico e reti bayesiane: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press. 3/e. [Ch.9]