Gestione di un dataset di Film

Componenti del gruppo

- Giuseppe Caggiano [MAT. 735400], g.caggiano8@studenti.uniba.it

Link progetto GitHub:

Progetto_ICon

Anno Accademico 2023-2024

INDICE

Capitolo 0: Introduzione

L'obiettivo di questo progetto è analizzare, classificare e organizzare i film contenuti nel dataset **IMDB-Movie-Data.csv**. Utilizzando tecniche di apprendimento non supervisionato (clustering) e di apprendimento supervisionato (classificatore) , i film vengono suddivisi in gruppi omogenei in base alle loro caratteristiche. Inoltre, il progetto utilizza **Prolog** per eseguire interrogazioni logiche sui dati dei film e sui loro cluster.

Requisiti funzionali

Il progetto è stato realizzato con Python versione 3.12 per la grande vastità di librerie che aiutano nell'implementazione. L'IDE che ho usato per la stesura del codice è stato PyCharm, semplice e funzionale. Le librerie che ho utilizzato sono le seguenti:

- **pandas**, **numpy**: per la manipolazione e analisi dei dati.

- **scikit-learn**: per il preprocessing, clustering e apprendimento supervisionato.

- **matplotlib**, **seaborn**: per la visualizzazione dei dati attraverso i grafici

- **pyswip**: per l'integrazione con Prolog.

- **imblearn (SMOTE)**: per la gestione delle classi sbilanciate.

- **joblib:** per salvare e caricare modelli di machine learning addestrati in formato binario

Configurazioni iniziali

Installare SWI-Prolog dal sito ufficiale https://www.swi-prolog.org/download/stable, fondamentale per l'utilizzo della libreria pyswip. Successivamente, dopo aver aperto il progetto con l'IDE preferito, avviare lo script install.py per scaricare tutte le librerie importanti per il corretto funzionamento del progetto. Successivamente, è possibile mandare in run il progetto vero e proprio tramite lo script main.py

Capitolo 1 : Dataset e Features

Il dataset è stato richiesto direttamente dal sito imdb.com, un database online che contiene informazioni sui vari film, come attori, valutazioni, anno... . Il dataset è salvato come **IMDB-Movie-Data.csv**. Le features usate per ogni singolo film sono:

- Title: titolo del film

- Genre : genere del film

- Directors :regista del film

- Actors: vari attori recitanti nel film

- Year : anno di pubblicazione del film

- Runtime (minutes) : durata del film in minuti

- Rating : valutazione data dagli utenti

- Metascore : punteggio all'interno del database IMDb

I dati salvati nel file **IMDB-Movie-Data.csv** sono poi preprocessati per renderli pronti all'analisi. I dati preprocessati sono salvati in un altro file, chiamato **processed_data.csv** .

Successivamente, poiché i film possono appartenere a più generi, è stato necessario binarizzare questa colonna per renderla utilizzabile dagli algoritmi di machine learning.

Le caratteristiche numeriche come **Runtime**, **Rating**, e **Metascore** sono state standardizzate per evitare che le diverse scale influiscano negativamente sui modelli. Vengono infine eliminati i dati incompleti per evitare problemi ed errori.Dopo queste fasi viene creato un nuovo file chiamato data_standardized.csv

Capitolo 2 : Clustering (Apprendimento non supervisionato)

L'apprendimento non supervisionato è una branca dell'apprendimento automatico in cui l'agente viene addestrato su un insieme di dati senza etichette. Esistono due principali tipi di apprendimento non supervisionato:

- Clustering : tecnica di machine learning che suddivide i dati in gruppi omogenei chiamati **cluster**. L'obiettivo di questa fase è raggruppare i film in base alle loro caratteristiche (come durata, rating e metascore) per identificare pattern nascosti nei dati, senza che vi sia un'etichetta predefinita per i cluster.

- Riduzione della dimensionalità :per ridurre il numero di feature utilizzate mantenendo però le informazioni più significative. Un esempio è la PCA (Principal Component Analysis)

Personalmente ho applicato due algoritmi di clustering e riduzione della dimensionalità tramite PCA.

Il clustering non supervisionato è stato applicato per raggruppare i film in base alle loro caratteristiche, permettendo di identificare gruppi omogenei di film. Gli algoritmi usati sono:

- KMeans: Algoritmo che divide i film in 5 cluster (scelto empiricamente, ma potrebbe essere ottimizzato utilizzando la tecnica del "gomito). E' stato usato inoltre un parametro random_state con valore 42 per garantire che i risultati del clustering siano riporducibili) Utilizza la distanza euclidea per identificare i punti dati più vicini tra loro.

- **DBSCAN**: Algoritmo di clustering basato sulla densità, in grado di identificare cluster di forma arbitraria e rilevare outlier (ovvero valori/punti anomali che si discostano dagli altri del dataset)

I parametri utilizzati sono:

- eps=0.5: La distanza massima tra due punti affinché siano considerati parte dello stesso cluster.

- min_samples=5: Il numero minimo di punti necessari per formare un cluster.

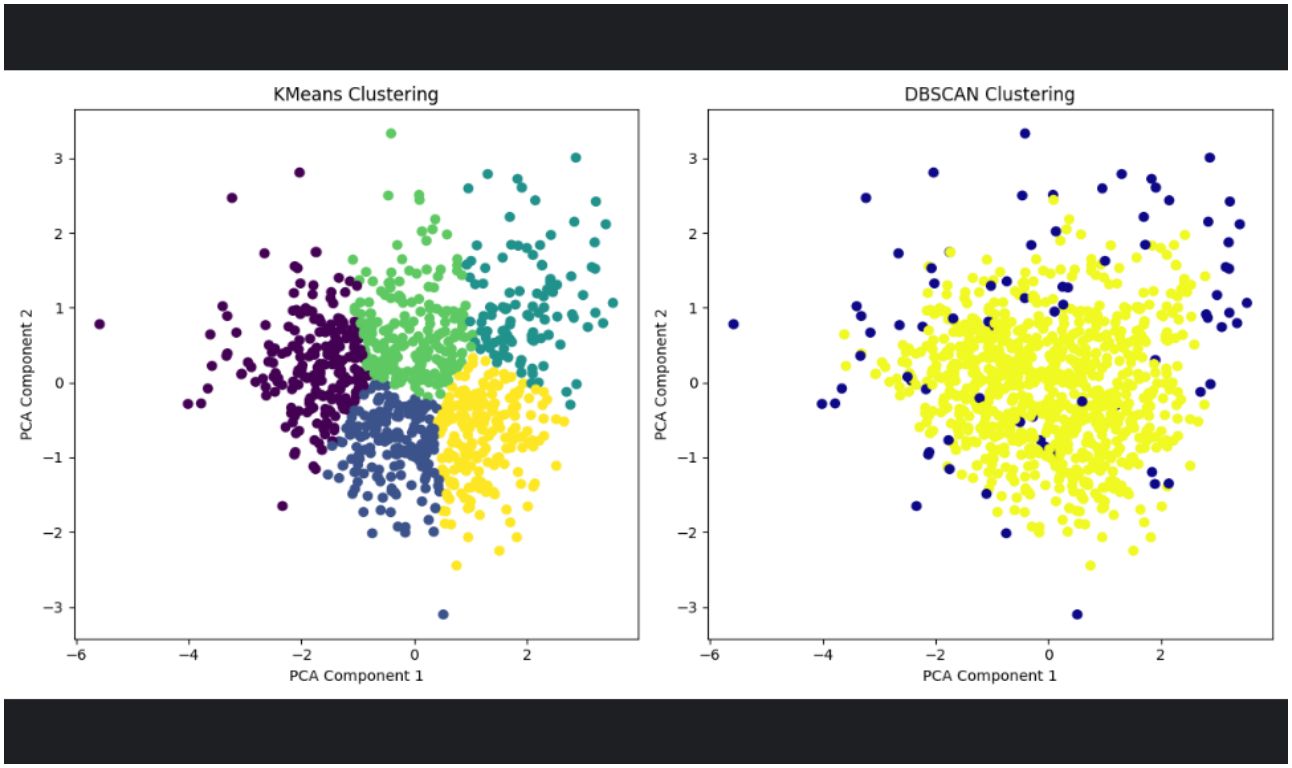
L'algoritmo individua i punti "core" (con almeno min_samples vicini entro una distanza di eps) e li espande finché non si esauriscono. I punti che non appartengono a nessun cluster vengono classificati come **outlier**.

Prima del clustering, i dati sono stati preprocessati tramite :

- Binarizzazione dei generi: I generi dei film sono rappresentati da liste di valori (ad esempio, un film può appartenere ai generi "Azione", "Avventura"). Per utilizzare questi dati nel clustering, i generi sono stati trasformati in variabili binarie (una colonna per ogni genere, con valori 0 o 1 a indicare se il film appartiene o meno a quel genere).

- Standardizzazione delle feature numeriche: È stata applicata una standardizzazione alle feature numeriche (come la durata del film, il rating e il metascore) utilizzando StandardScaler. Questa operazione è importante perché algoritmi come **KMeans** si basano sulla distanza tra punti, e avere caratteristiche con scale molto diverse (ad esempio, minuti vs rating) potrebbe distorcere i risultati.

Ho generato due grafici che mostrano i risultati del clustering per **KMeans** e **DBSCAN**, evidenziando le somiglianze tra i film in uno spazio bidimensionale. Eccoli:



Per visualizzare i risultati del clustering, è stato applicato l'algoritmo di riduzione dimensionale **PCA**, che riduce i dati da uno spazio multidimensionale (caratteristiche originali) a uno spazio bidimensionale. Questo rende possibile visualizzare i cluster in un piano bidimensionale , **PCA** conserva il maggior numero possibile di informazioni, riducendo le dimensioni a due componenti principali.

Infine vengono salvati i dati con i cluster all'interno del file **clustered_data.csv**

Capitolo 3 : Apprendimento Supervisionato

L'apprendimento supervisionato ha lo scopo di addestrare modelli di machine learning per classificare i film nei cluster generati durante la fase di clustering. In questa fase, utilizziamo i dati etichettati dai cluster e applichiamo vari modelli di classificazione per prevedere a quale cluster un film appartiene in base alle sue caratteristiche.

Gli algoritmi di classificazione utilizzati sono :

- **RANDOM FOREST**

Modello di apprendimento supervisionato in cui ogni albero viene addestrato su un sottoinsieme del dataset, e la previsione finale è data dalla media o dalla maggioranza delle predizioni di tutti gli alberi.

I parametri principali utilizzati sono:

n_estimators: Numero di alberi nella foresta.

max_depth: Profondità massima degli alberi, controlla la complessità del modello.

min_samples_split: Numero minimo di campioni richiesti per dividere un nodo interno.

Il Random Forest viene utilizzato per classificare i film in base ai cluster identificati nella fase di clustering.

- SUPPORT VECTOR CLASSIFIER (SVC)

L'SVC cerca di trovare l'iperpiano che massimizza il margine tra le diverse classi nel dataset. E' un modello che separa le classi massimizzando la distanza tra i punti dati più vicini delle classi differenti.

I principali parametri utilizzati sono:

c : Il parametro di regolarizzazione, che controlla il compromesso tra un margine più ampio e la corretta classificazione dei campioni di addestramento.

kernel : La funzione kernel che definisce il tipo di trasformazione applicata ai dati.

L'SVC è stato utilizzato per classificare i film nei rispettivi cluster e include la probabilità di previsione per ciascuna classe.

- **K-NEAREST NEIGHBORS (KNN)**

Il KNN è un classificatore **basato sulla vicinanza**. Assegna una classe a un nuovo punto in base alle classi dei **k-nearest neighbors** (i k punti più vicini) nel set di addestramento. I parametri usati sono:

n_neighbors: Il numero di vicini da considerare per la classificazione.

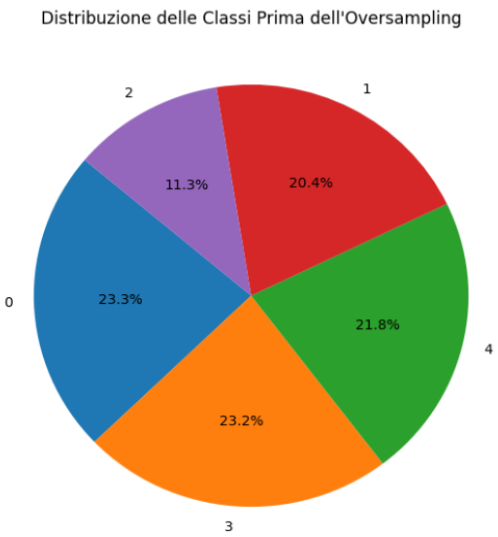
weights: Definisce se i vicini devono essere considerati tutti uguali (uniform) o pesati in base alla distanza (distance)

Partendo col processo di apprendimento supervisionato viene utilizzato come dataset clustered_data.csv , già preprocessato e arricchito con le informazioni sui cluster. Questo viene caricato per addestrare i modelii e le colonne non rilevanti per la classificazione (come **Title**, **Director**, **Year**) vengono eliminate dal dataset.

Uno dei problemi comuni nei dataset di classificazione è lo sbilanciamento tra le classi. In questo progetto, alcune classi (cluster) contengono molti più film rispetto ad altre. Per bilanciare il dataset, è stata utilizzata la tecnica **SMOTE (Synthetic Minority Over-sampling Technique)**,che genera nuovi campioni sintetici per le classi meno rappresentate, assicurando che ciascun cluster abbia un numero equilibrato di dati. Questo migliora la capacità del modello di classificare correttamente anche i film appartenenti a classi più piccole.

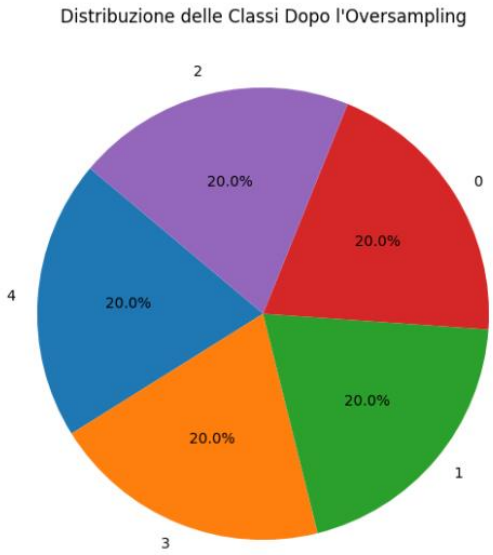
Possiamo notarlo attraverso questi dati ottenuti prima dell'oversampling :

```
Distribuzione delle classi prima dell'oversampling:
clusterIndex
0    218
3    217
4    204
1    191
2    106
```



E dopo...

```
Distribuzione delle classi dopo l'oversampling:
clusterIndex
4    218
3    218
1    218
0    218
2    218
```



Successivamente il dataset viene suddiviso in **training set** (80%) e **test set** (20%) utilizzando la **stratificazione**. La stratificazione garantisce che le proporzioni delle classi siano preservate in entrambi i set, migliorando la rappresentatività del dataset durante l'addestramento e la valutazione.

Le feature numeriche (come **Runtime**, **Rating**, **Metascore**) vengono standardizzate utilizzando **StandardScaler**. Questo garantisce che tutte le feature abbiano una media pari a 0 e una deviazione standard pari a 1, evitando che feature con scale molto diverse influiscano eccessivamente sui modelli.

Per quanto riguarda la ricerca degli **iperparametri** , per ogni modello, viene eseguita una **GridSearchCV** per determinare i migliori iperparametri utilizzando una validazione incrociata a 3 **fold**. La validazione incrociata riduce il rischio di overfitting e assicura che i modelli funzionino bene su dati non visti.

IPERPARAMETRI UTILIZZATI:

```
param_grid = {
    'RandomForest': {
        'n_estimators': [50, 100],  # Ridotto numero di estimatori
        'max_depth': [5, 10],  # Limitata la profondità
        'min_samples_split': [2, 5]
    },
    'SVC': {
        'C': [0.1, 1, 10],
        'kernel': ['linear', 'rbf']
    },
    'KNeighbors': {
        'n_neighbors': [3, 5],
        'weights': ['uniform']
    }
```

I Parametri che sono stati restituiti sono:

```
Addestramento del modello: RandomForest
Migliori parametri per RandomForest: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 50}
Accuracy migliore su CV per RandomForest: 0.5435873128727733

Addestramento del modello: SVC
Migliori parametri per SVC: {'C': 10, 'kernel': 'linear'}
Accuracy migliore su CV per SVC: 0.49772089900067157

Addestramento del modello: KNeighbors
Migliori parametri per KNeighbors: {'n_neighbors': 5, 'weights': 'uniform'}
Accuracy migliore su CV per KNeighbors: 0.46106568708772766
```

Per la fase di addestramento e di test i modelli sono stati addestrati utilizzando una KFold Cross Validation con K = 3

```
grid_search = GridSearchCV(
    models[model_name],
    param_grid[model_name],
    cv=KFold(n_splits=3, shuffle=True, random_state=42),
    scoring='accuracy',
    n_jobs=-1
)
grid_search.fit(X_train, y_train)
best_models[model_name] = grid_search.best_estimator_
```

I modelli addestrati vengono valutati sui dati di test utilizzando diverse metriche:

- **Accuracy**: Percentuale di film classificati correttamente.

```
Accuracy su training per RandomForest: 0.7694954128440367
Accuracy su test per RandomForest: 0.5321100917431193
```

```
Accuracy su training per SVC: 0.7580275229357798
Accuracy su test per SVC: 0.5229357798165137
```

```
Accuracy su training per KNeighbors: 0.6284403669724771
Accuracy su test per KNeighbors: 0.47706422018348627
```

- **Precision, Recall, F1-Score**: Metriche che misurano la capacità del modello di classificare correttamente i film e bilanciare precisione e recall.

```
Metriche dei modelli:
          Model  Accuracy  Precision    Recall  F1 Score
0  RandomForest  0.532110   0.503089  0.532110  0.491273
1           SVC  0.522936   0.505522  0.522936  0.505723
2    KNeighbors  0.477064   0.452221  0.477064  0.454956
```

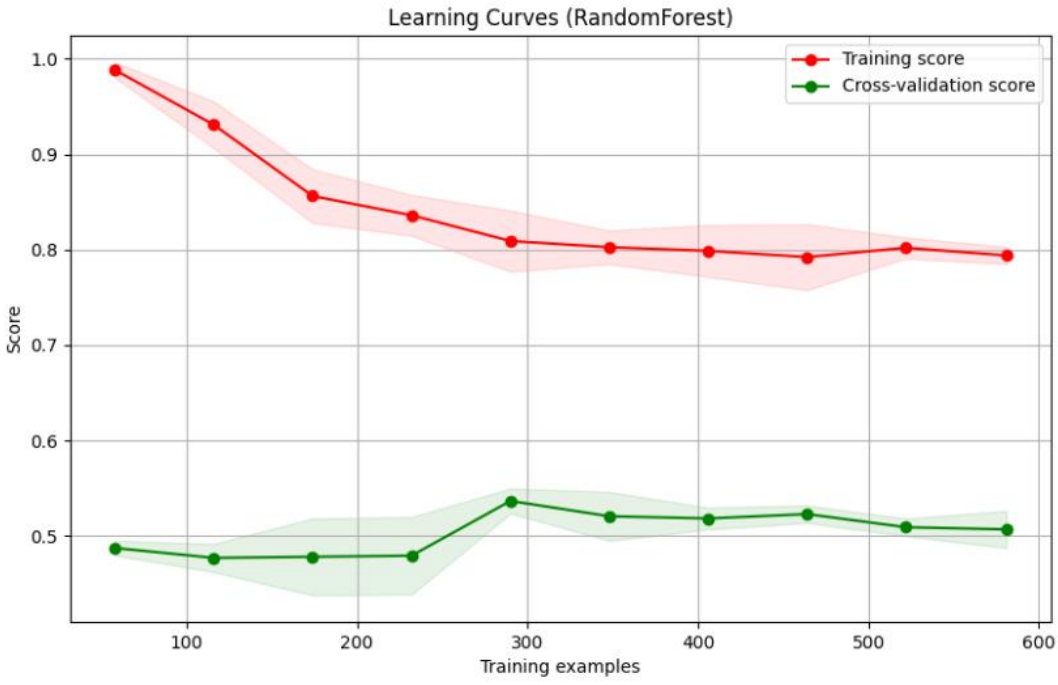
- **Confusion Matrix**: Matrice che mostra come i film di ciascun cluster sono stati classificati.

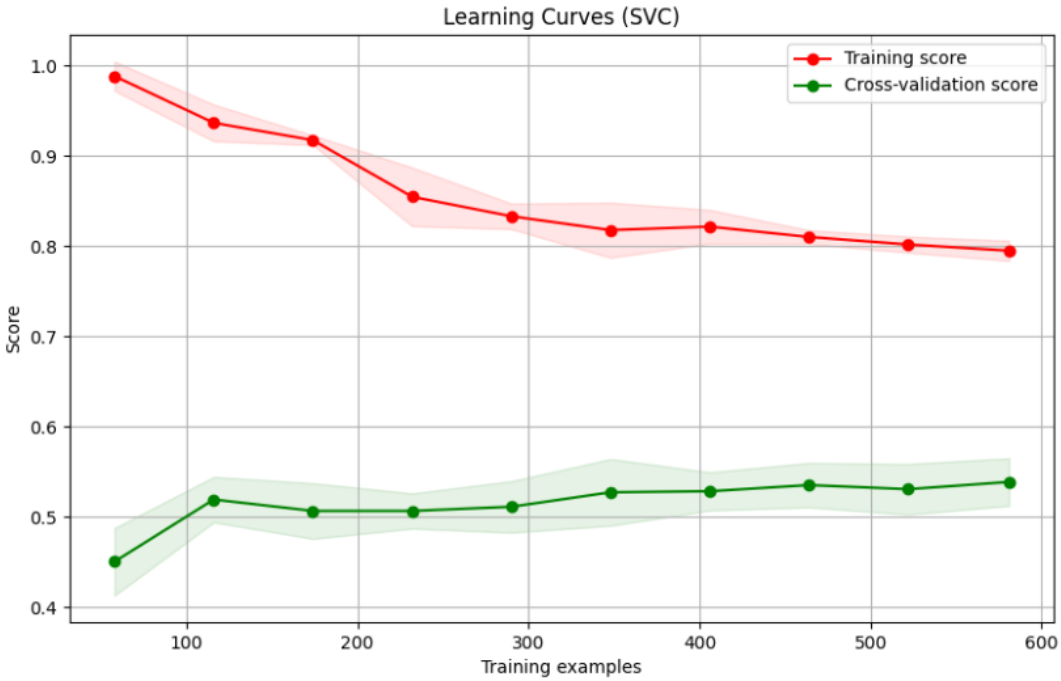
```
Confusion Matrix per RandomForest:
 [[12 22  0  3  7]
 [ 7 30  0  0  6]
 [ 0  0 44  0  0]
 [ 3  2 15  6 18]
 [ 3  9  0  7 24]]
```

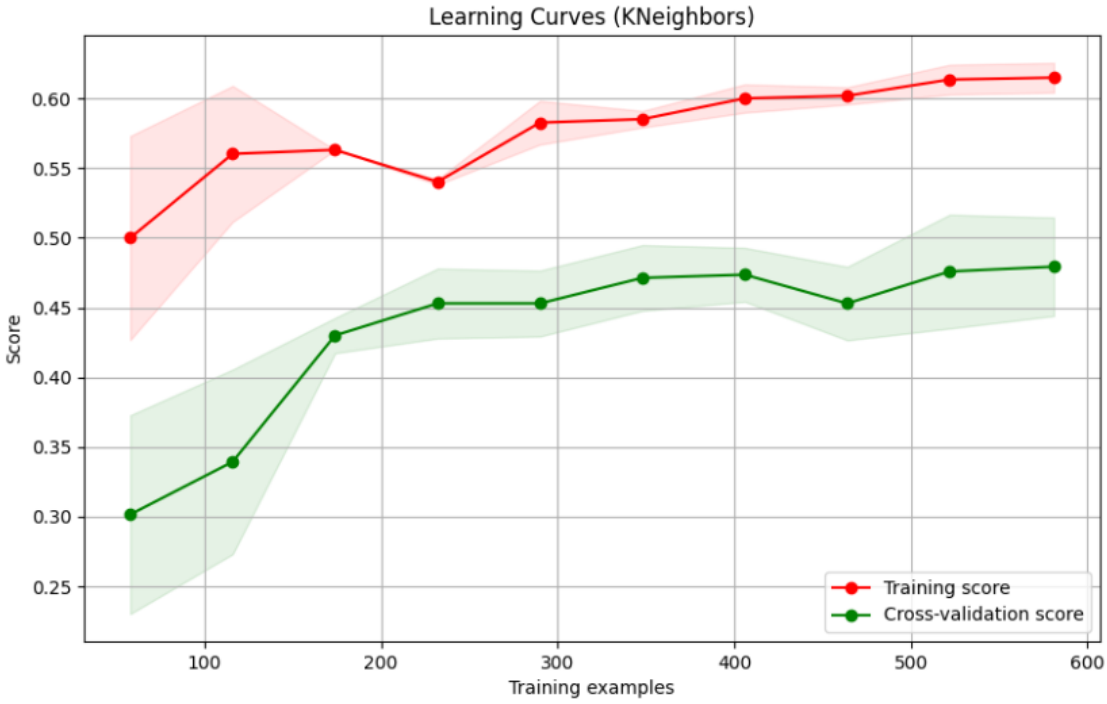
```
Confusion Matrix per SVC:
 [[18 17  0  4  5]
 [11 28  0  1  3]
 [ 0  0 40  4  0]
 [ 4  3 14 13 10]
 [ 7 11  1  9 15]]
```

```
Confusion Matrix per KNeighbors:
 [[20 17  0  4  3]
 [10 28  0  0  5]
 [ 2  1 36  4  1]
 [ 9  3 14  7 11]
 [ 7 13  1  9 13]]
```

- **Curva di apprendimento**: Un grafico che mostra come la performance del modello cambia con l'aumento della quantità di dati di addestramento.







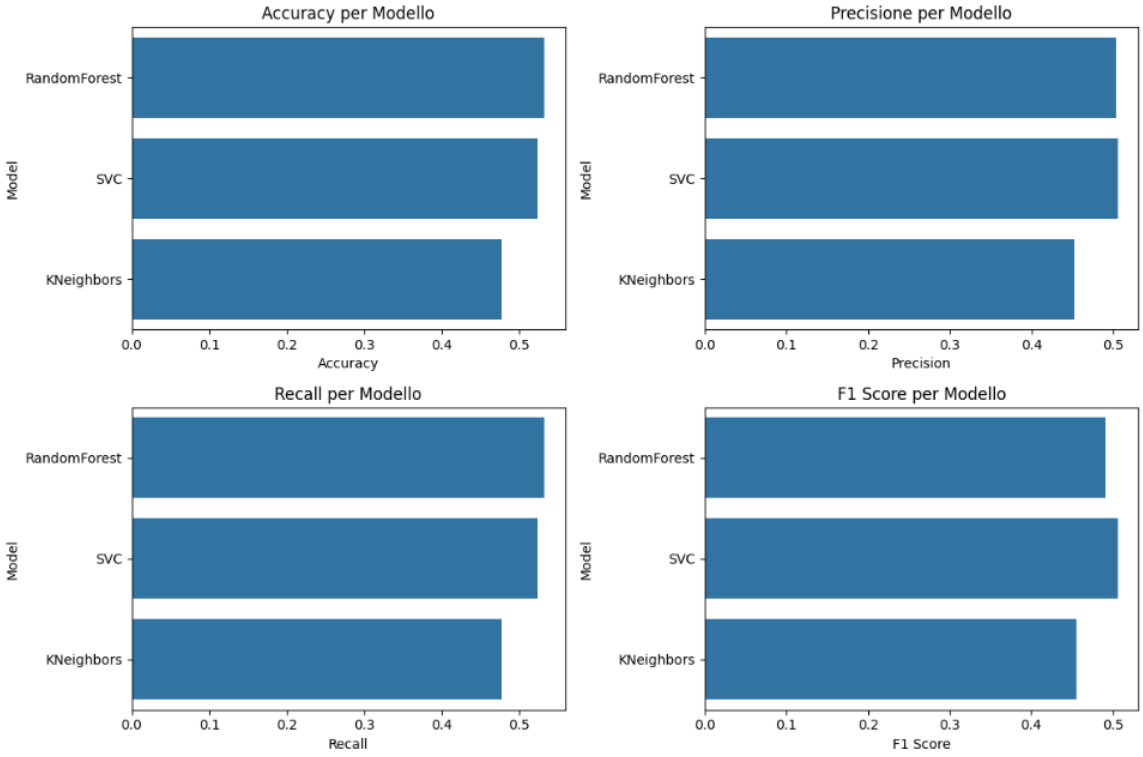
-VARIANZA E DEVIAZIONE STANDARD

Per ogni modello, vengono anche calcolati la **varianza** e la **deviazione standard** per i dati di training e test, al fine di verificare la stabilità del modello. Per esempio quelle del Random Forest:

```
Training Data Variance per RandomForest: 2.00
Training Data Standard Deviation per RandomForest: 1.41
Test Data Variance per RandomForest: 2.00
Test Data Standard Deviation per RandomForest: 1.41
```

- Valutazione delle performance

I risultati vengono visualizzati utilizzando grafici a barre che mostrano le performance relative di **Accuracy**, **Precision**, **Recall** e **F1-Score** per ciascun modello. Questo consente di confrontare facilmente quale modello performa meglio su ciascuna metrica.



Il miglior modello, scelto in base all'accuratezza sui dati di test (in questo caso RandomForest) viene salvato in un file **best_classification_model.pkl** utilizzando **joblib**. Questo permette di riutilizzare il modello in futuro senza doverlo riaddestrare.

Capitolo 4: Integrazione con Prolog

L'integrazione con Prolog ha l'obiettivo di utilizzare un motore logico per eseguire interrogazioni sui film e sui cluster in cui sono stati classificati. Prolog è un linguaggio di programmazione logico per gestire dati dichiarativi e per eseguire inferenze basate su regole e fatti predefiniti. E' necessario l'uso della libreria pyswip.

I dati dei film vengono trasformati in **fatti logici** e vengono definite **regole logiche** che collegano i film ai loro cluster. L'utente può inserire il titolo di un film e ottenere informazioni come l'anno di uscita e il cluster di appartenenza.

Il dataset preprocessato, che include i film e i loro cluster, viene caricato dal file **clustered_data.csv**. Ogni film viene rappresentato come un **fatto logico** in Prolog all'interno di un file temporaneo chiamato **kb_temp.pl**. I fatti descrivono le caratteristiche del film e il cluster di appartenenza.

I fatti sono generati dalla funzione **write_temp_facts**, che trasforma ogni film del dataset in un fatto logico.

Viene definita anche una **regola logica** che permette di collegare le informazioni sui film ai loro cluster. La regola stabilisce una relazione tra i film e i cluster in cui sono stati classificati, permettendo a Prolog di rispondere a query come "In quale cluster si trova il film X?".

L'utente può interrogare la base di conoscenza inserendo il titolo di un film. La query Prolog viene eseguita per cercare l'anno di uscita e il cluster a cui appartiene il film. Se il film è presente nella base di conoscenza, viene restituito il risultato. Di seguito lascio un esempio di interrogazione:

```
Inizio della creazione e interrogazione della base di conoscenza Prolog...
Inserisci il titolo del film da cercare: Deadpool
:- encoding(utf8).
movie('Deadpool', 'Tim Miller', 2016, 108, 8.0, 65.0).
clustered_movie(2016, 108, 8.0, 65.0, 4).
movie_info(Title, Year, Cluster) :- movie(Title, _, Year, _, _, _), clustered_movie(Year, _, _, _, Cluster).

Title: Deadpool, Year: 2016, Cluster: 4
Creazione e interrogazione della base di conoscenza Prolog completate.
```

Capitolo 5 : Conclusioni e Sviluppi Futuri

Questo progetto di ingegneria della conoscenza ha dimostrato l'efficacia dell'integrazione di tecniche di machine learning, clustering, apprendimento supervisionato e ragionamento logico tramite Prolog nell'analisi e organizzazione di un dataset complesso come quello dei film.

Ci sono molte aree che potrebbero essere esplorate e migliorate nel progetto per renderlo più completo ,come ad esempio l'**ottimizzazione del numero di cluster in KMeans** (usare tecniche come la **Curva del Gomito** per determinare automaticamente il numero ottimale di cluster invece di scegliere un valore arbitrario), **la ricerca su più iperparametri** (tramite tecniche come Random Search o Bayesian Optimization per migliorare ulteriormente le performance dei modelli) o integrare il progetto con fonti esterne di dati (API) per arricchire o aggiornare il dataset. Inoltre potrebbe essere più intuitivo creare un'interfaccia grafica per rendere il sistema più accessibile e facile da usare.