Gestione e Analisi dei Film

Componenti del gruppo

- de Panizza Aurora [MAT. 797790] – a.depanizza@studenti.uniba.it
- Bruno Nicolò [MAT. 798497] – n.bruno19@studenti.uniba.it
- Cormio Sara [MAT. 810310] – s.cormio2@studenti.uniba.it

Link GitHub: Repository

A.A. 2025-2026

Indice

Capitolo 0 - Introduzione **2**
0.1 Obiettivi del progetto **2**
0.2 Ambiente di sviluppo e linguaggio **2**
0.3 Librerie utilizzate **2**
Capitolo 1 - Acquisizione e preparazione del Dataset **4**
1.1 Acquisizione del dataset grezzo e filtraggio preliminare **4**
1.2 Preprocessing del dataset **5**
Capitolo 2 – Apprendimento non supervisionato **6**
2.1 Scelta dei parametri del clustering **6**
2.2 Analisi e interpretazione dei cluster **7**
Capitolo 3 – Apprendimento Supervisionato **10**
3.1 Struttura del modello di classificazione **10**
3.2 Modelli utilizzati **10**
3.2.1 Decision Tree 11
3.2.2 Random Forest 11
3.2.3 Regressione Logistica 11
3.3 Valutazione dei modelli **12**
3.3.1 Accuracy 12
3.3.2 F1 - score 13
3.3.3 Matrici di confusione 14
Matrice di confusione - Decision Tree 14
Matrice di confusione - Random Forest 15
Matrice di confusione - Regressione Logistica 16
3.3.4 Analisi delle curve di apprendimento 17
3.3.5 Confronto tra modelli 19
Capitolo 4 - Ragionamento probabilistico e Bayesian Network **20**
4.1 Introduzione alle Bayesian Network **20**
4.2 Preparazione dei dati e costruzione della rete **20**
4.3 Inferenza probabilistica e risultati ottenuti **21**
Capitolo 5 – Ragionamento Logico e Prolog **21**
5.1 Integrazione del ragionamento logico nel sistema di conoscenza **23**
5.2 Costruzione della Knowledge Base **23**
5.3 Utilizzo del Ragionamento Logico **24**
5.4 Integrazione dei modelli nel sistema di conoscenza **24**
5.5 Conclusioni e sviluppi futuri **24**
Bibliografia **26**

Capitolo 0 - Introduzione

0.1 Obiettivi del progetto

L'obiettivo del progetto FilmProject è di realizzare un sistema intelligente capace di integrare tecniche di apprendimento automatico e modelli di rappresentazione della conoscenza per analizzare e organizzare un insieme eterogeneo di film. In particolare, il sistema utilizza tecniche di clustering per individuare strutture latenti nei dati, modelli supervisionati per la classificazione di nuovi elementi e meccanismi di ragionamento logico e probabilistico per arricchire e interpretare le informazioni disponibili.

L'intero processo consente di ottenere un sistema scalabile e automatizzabile, in cui l'inserimento di nuovi film avviene tramite la stessa pipeline di preprocessing, apprendimento e ragionamento, senza richiedere una rielaborazione manuale della conoscenza.

Si precisa che il sistema non è finalizzato alla raccomandazione di contenuti, ma all'analisi strutturata e alla classificazione semantica dei film attraverso modelli di apprendimento automatico e meccanismi di ragionamento logico e probabilistico.

0.2 Ambiente di sviluppo e linguaggio

Per lo sviluppo del progetto è stato utilizzato Visual Studio Code data la sua semplicità e per il supporto alle estensioni Python, linguaggio scelto per la sua sintassi chiara e per la presenza di librerie dedicate alla gestione dei dati e al machine learning. FilmProject è stato realizzato interamente su macOS.

0.3 Librerie utilizzate

Le librerie utilizzate nel progetto sono:

- **pandas**: importazione, selezione e pulizia del dataset, manipolazione dei DataFrame.
- **scikit_learn:** per il preprocessing, la normalizzazione delle feature numeriche e l'applicazione dei modelli di clustering e classificazione.
    - sklearn.preprocessing.MinMaxScaler: per la normalizzazione delle feature.
    - sklearn.cluster.KMeans: per il clustering.
    - sklearn.model_selection.train_test_split: per train/test split.
    - sklearn.tree.DecisionTreeClassifier, RandomForestClassifier, LogisticRegression.
    - sklearn.metrics: per accuracy, confusion matrix, classification report.
- **os**: gestione dei percorsi dei file e interazione con il filesystem.
- **matplotlib**: creazione di grafici e visualizzazione dei risultati relativi a clustering e classificazione.

- **seaborn**: libreria complementare a matplotlib, permette di ottenere grafici più leggibili, utili nella fase di analisi esplorativa dei dati.
- **pgmpy**: per la costruzione e l'inferenza di reti bayesiane, utilizzate nella fase di ragionamento probabilistico.

    - pgmpy.models.DiscreteBayesianNetwork: è la classe che consente di definire e costruire una rete bayesiana discreta a partire dalla struttura specificata dagli archi.
    - pgmpy.inference.VariableElimination: è il modulo dedicatoa all'inferenza probabilistica esatta.

- **pyswip**: interfaccia Python–Prolog utilizzata per caricare la knowledge base Prolog ed eseguire query di ragionamento logico a partire dai dati del progetto.

Capitolo 1 - Acquisizione e preparazione del Dataset

1.1 Acquisizione del dataset grezzo e filtraggio preliminare

Il dataset utilizzato nel progetto, movies_data.csv, costituisce la base informativa iniziale del sistema ed è composto da 10.880 film. Esso include informazioni eterogenee quali titolo originale, id del film, identificatore IMDb, pagina ufficiale del film, genere, frase promozionale, case di produzione, popolarità, valutazioni, voto medio, anno di uscita, cast, direttori, budget, botteghino, parole chiave (keywords), durata, una breve descrizione (overview), budget aggiustato con l'inflazione **(**budget_adj) e incasso aggiustato con l'inflazione (revenue_adj).

Tale dataset, reperito online (8), viene sottoposto a un processo di selezione finalizzato alla costruzione di una rappresentazione coerente con gli obiettivi del sistema. Data l'eterogeneità delle informazioni disponibili, una prima fase di filtraggio risulta necessaria per isolare esclusivamente gli attributi utili alle successive fasi di apprendimento e ragionamento (2).

La fase di acquisizione del dataset viene gestita attraverso il file data_creation.py che si occupa di leggere il file e convertirlo in un DataFrame utilizzabile tramite la funzione **load_raw_dataset()**.

Successivamente, la funzione **select_relevant_columns()** esegue un filtraggio preliminare delle colonne, riducendo il dataset alle sole informazioni ritenute necessarie. Il DataFrame viene quindi ridotto alle *colonne rilevanti*, definite all'interno del codice nella variabile *colonneUtili.* Le colonne considerate definitivamente utili ai fini del progetto sono:

- **original_title**: titolo nella lingua originale del film
- **genres**: raccoglie l'elenco dei generi associati al film, indispensabile perché nel preprocessing verrà trasformata in un insieme di variabili binarie che rappresentano la presenza o assenza dei singoli generi.
- **popularity**: esprime la popolarità del film e pertanto è una misura dell'interesse generale del pubblico.
- **vote_average**: rappresenta la valutazione media ottenuta dal film.
- **vote_count**: indica il numero totale di voti ricevuti, fornendo un contesto alla valutazione media; difatti una valutazione alta ottenuta da pochissime persone non ha lo stesso peso di una valutazione simile ottenuta da migliaia di utenti e viceversa in caso di valutazione bassa.
- **runtime**: indica la durata in minuti del film.
- **release_year**: l'anno di rilascio del film.

1.2 Preprocessing del dataset

Il preprocessing ha lo scopo di rendere il dataset coerente e pronto per essere utilizzato dagli algoritmi di machine learning. Questa fase comprende la gestione dei valori mancanti, l'uniformazione dei formati e la trasformazione delle variabili in forme numeriche adeguate (3).

La prima parte di questa fase riguarda la pulizia dei dati, realizzata tramite la funzione **clean_dataset()**, che crea una copia del DataFrame originale, rimuove eventuali duplicati ed elimina le istanze prive di valori nelle colonne fondamentali (popularity, vote_average e vote_count). L'assenza di questi dati comprometterebbe la qualità delle informazioni disponibili per i modelli di apprendimento.

Successivamente viene elaborata la colonna genres**,** originariamente rappresentata come stringa con elementi separati dal carattere "| (ad esempio "Action|Drama"). La funzione **encode_genres()** utilizza **parse_genres()** per trasformare tale stringa in una lista di generi, memorizzata nella nuova colonna lista_generi. A partire da questa rappresentazione, il sistema analizza la frequenza dei generi e seleziona i top-k più comuni tramite un parametro configurabile, che consente di controllare direttamente il numero di variabili binarie generate.

La scelta di limitare la rappresentazione ai *top-k* generi consente di ridurre la dimensionalità dello spazio delle feature ed evitare una rappresentazione eccessivamente sparsa, migliorando la stabilità e l'efficacia dei modelli di clustering e classificazione. Per ciascun genere selezionato viene creata una variabile binaria del tipo *is_<genere>*, che assume valore 1 se il film appartiene al genere corrispondente, 0 altrimenti.

Una volta generate le nuove feature, la funzione **select_feature_columns()** seleziona esclusivamente le colonne numeriche e booleane, eliminando qualsiasi informazione non compatibile con i modelli utilizzati. Le variabili numeriche vengono poi normalizzate tramite MinMaxScaler, che porta tutti i valori nell'intervallo [0, 1]. Prima della normalizzazione, le colonne numeriche vengono convertite in tipo float per garantire la compatibilità con i valori reali prodotti dallo scaling e preservare la coerenza del dataset.

La normalizzazione è un passaggio essenziale per gli algoritmi basati su misure di distanza, come il clustering KMeans, poiché impedisce che variabili con scale diverse influenzino in modo scorretto il modello (2).

Al termine del preprocessing, il dataset risultante è composto esclusivamente da feature numeriche normalizzate e variabili binarie derivate dai generi. Il dataset così ottenuto viene salvato tramite la funzione **save_clean_dataset()** in formato CSV, risultando pronto per le successive fasi di clustering, classificazione e integrazione con i moduli di ragionamento del sistema.

Capitolo 2 – Apprendimento non supervisionato

In questo progetto l'apprendimento non supervisionato viene utilizzato come strumento esplorativo per individuare strutture latenti nel dataset dei film e costruire una categorizzazione emergente basata esclusivamente sulle caratteristiche osservabili. L'obiettivo del clustering non è quello di riprodurre etichette predefinite, ma di estrarre nuove informazioni dai dati, successivamente integrate nelle fasi di apprendimento supervisionato e di ragionamento del sistema (1).

Il clustering consente di assegnare a ciascun film un'identità di gruppo (cluster_id) che sintetizza il profilo complessivo dell'elemento in termini di popolarità, valutazioni, durata e generi. Tale informazione rappresenta una forma di conoscenza derivata automaticamente, che arricchisce la rappresentazione del dataset.

Nell'ambito dell'apprendimento non supervisionato esistono diversi approcci, tra cui il clustering e la riduzione della dimensionalità; nel presente progetto l'attenzione è rivolta esclusivamente al clustering, in quanto funzionale all'individuazione di gruppi omogenei di film e alla successiva integrazione dei risultati nei moduli di classificazione e di ragionamento.

Nel clustering esistono due principali approcci:

- Hard clustering: ogni elemento viene assegnato in modo univoco a un cluster.
- Soft clustering: ogni elemento possiede una distribuzione di probabilità sui cluster.

Il progetto utilizza un approccio di hard clustering, nello specifico il metodo K-Means, che suddivide i dati minimizzando la distanza tra ogni punto e il centroide del cluster corrispondente. (1). Questa scelta è coerente con l'obiettivo di utilizzare l'etichetta di cluster cluster_id come informazione discreta nei moduli successivi di classificazione e di ragionamento del sistema. Per la generazione dei grafici sono state utilizzate le librerie matplotlib e seaborn.

2.1 Scelta dei parametri del clustering

Uno degli aspetti centrali del K-Means consiste nella scelta del numero ottimale di cluster, parametro noto come k, i cui diversi valori possono portare a soluzioni molto differenti, rendendo necessaria una metodologia oggettiva per selezionarlo (4).

Per questo motivo è stato utilizzato il *metodo del gomito* (elbow method), una strategia che valuta l'andamento dell'*inertia* al variare del numero di cluster. L'inertia rappresenta la somma delle distanze quadrate tra ogni punto e il centroide del cluster assegnato, e tende naturalmente a diminuire man mano che aumenta il numero dei cluster (4).

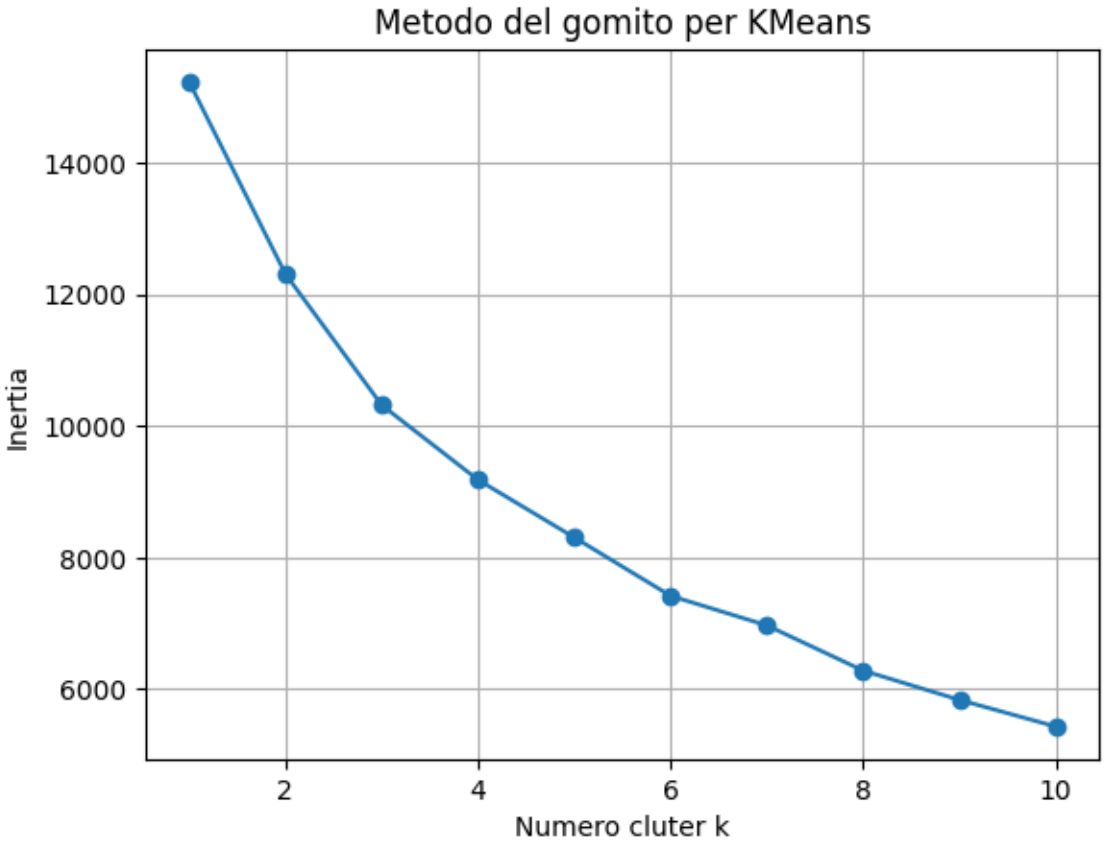
Nel codice questa procedura è implementata in due passaggi principali. La funzione **compute_inertia_for_k()** si occupa di eseguire il KMeans per valori di k compresi tra 1 e 10; per ciascuno di essi viene creato un modello con n_init = 5 e random_state = 42, in modo da ottenere risultati stabili e meno influenzati dall'inizializzazione casuale dei centroidi. Per ogni valore di k, viene calcolata l'inertia e memorizzata in una lista.

I valori ottenuti vengono poi passati alla funzione **plot_elbow_curve()**, che si occupa di visualizzare graficamente l'andamento dell'inertia al variare del numero di cluster. La funzione crea un nuovo

grafico matplotlib, traccia la curva che mette in relazione i valori di k con le rispettive inertia e applica alcuni elementi grafici utili alla lettura del risultato.

Infine, il grafico viene salvato automaticamente nel percorso plots/elbow.png tramite la funzione **plt.savefig()**, che consente di esportarlo direttamente come immagine. Questo permette di avere una rappresentazione visiva chiara della curva del gomito, indispensabile per riconoscere il valore di k in corrispondenza del quale la diminuzione dell'inertia diventa meno pronunciata.

Il grafico mostra come l'inertia decresca rapidamente per valori di k fino a circa 3–4, per poi diminuire più lentamente. Il punto in cui la curva cambia pendenza in modo significativo rappresenta il cosiddetto "gomito" e coincide con il valore ottimale di k. Dall'osservazione del grafico prodotto, il numero di cluster più adeguato risulta essere k = 4, che rappresenta il miglior compromesso tra la complessità del modello e la capacità di descrivere correttamente la struttura interna dei dati. Questo valore viene quindi utilizzato per eseguire il clustering definitivo nella fase successiva.

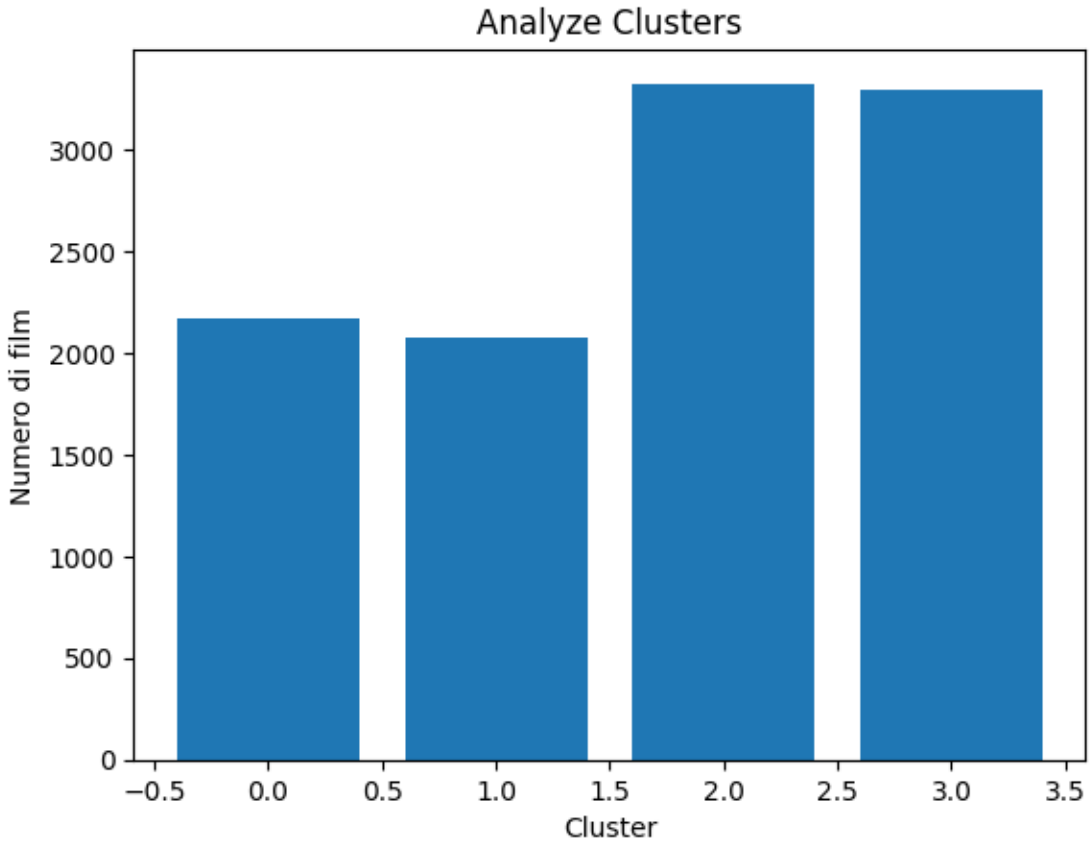


2.2 Analisi e interpretazione dei cluster

Una volta determinato il numero ottimale di cluster, il passo successivo è applicare il KMeans definitivo al dataset preprocessato. Nel codice questa operazione è gestita dalla funzione **run_kmeans()**, che esegue l'algoritmo utilizzando k = 4 e restituisce sia l'assegnazione di ciascun film al proprio cluster sia le coordinate dei centroidi. Le etichette ottenute vengono poi integrate nel dataset tramite la funzione **attach_cluster_labels()**, che aggiunge una nuova colonna denominata cluster_id. Il dataset arricchito viene infine salvato nel file movies_with_clusters.csv, così da essere riutilizzato nelle fasi successive del progetto.

Tale distribuzione non risulta perfettamente bilanciata, un fenomeno comune nei problemi reali di clustering, in cui i gruppi naturali presenti nei dati possono avere dimensioni molto diverse (4).

Per comprendere la distribuzione dei film nei diversi gruppi è stata utilizzata la funzione **analyze_clusters()**, che calcola il numero di elementi presenti in ciascun cluster attraverso **value_counts()** e genera un grafico salvato nel file plots/cluster_distribution.png.



Questo grafico permette di verificare visivamente come i cluster non siano perfettamente bilanciati in quanto alcuni gruppi includono un numero maggiore di film, mentre altri risultano meno popolati. Tale sbilanciamento è un aspetto importante da considerare, poiché può influenzare le prestazioni degli algoritmi di apprendimento supervisionato.

Oltre alla distribuzione numerica, il file cluster_means.csv riporta le medie di tutte le feature per ciascun cluster, permettendo di ottenere una prima interpretazione del contenuto dei gruppi individuati. I cluster non rappresentano categorie definite manualmente, ma emergono automaticamente dalle somiglianze presenti nei dati.

Sulla base delle medie calcolate, è possibile descrivere ciascun cluster nel modo seguente:

Cluster 0 – Film thriller/drammatici con valutazioni nella media.
Questo cluster presenta valori medi di popolarità e numero di voti relativamente bassi ma un'alta presenza del genere thriller (≈ 0.80), insieme a componenti drama e horror. I film appartenenti a questo gruppo risultano spesso più cupi o intensi, con una valutazione media attorno a 0.53 e una popolarità contenuta.

Cluster 1 – Film d'azione e thriller, più popolari e votati.
Il cluster 1 è caratterizzato da una dominanza assoluta del genere action (valore medio = 1.0) e una forte presenza di thriller e adventure. I valori medi di popolarità e voto_count sono più alti rispetto agli altri cluster, indicando film tendenzialmente più commerciali o diffusi, con valutazioni leggermente superiori rispetto al cluster 0.

Cluster 2 – Film comici e leggeri con buon gradimento del pubblico.
Nel cluster 2 prevale nettamente il genere comedy (≈ 0.67), mentre gli altri generi sono meno rappresentati. Questo gruppo mostra valori bassi di popolarità e numero di voti, ma un voto medio relativamente alto (≈ 0.59), suggerendo si tratti di film meno "mainstream" ma fortemente apprezzati da chi li guarda.

Cluster 3 – Film drammatici con le valutazioni più elevate.
Il cluster 3 mostra una forte predominanza del genere drama (valore medio = 1.0), con una componente crime significativa. I film di questo cluster presentano in media il voto più alto tra tutti i gruppi (≈ 0.62) e una durata leggermente superiore. Si tratta tipicamente di film più seri, narrativi e spesso più apprezzati dal pubblico.

L'etichetta di cluster cluster_id ottenuta in questa fase verrà utilizzata come variabile target nella fase di apprendimento supervisionato, consentendo di valutare la capacità dei modelli di classificare correttamente nuovi film sulla base delle feature disponibili.

Capitolo 3 – Apprendimento Supervisionato

3.1 Struttura del modello di classificazione

L'apprendimento supervisionato rappresenta la fase del progetto in cui l'obiettivo è costruire un modello in grado di prevedere automaticamente il cluster di appartenenza di un film a partire dalle sue caratteristiche. In questo contesto, ogni esempio è descritto da un vettore di feature e associato a un'etichetta nota: nel nostro caso il cluster di appartenenza, ottenuto tramite la fase di clustering non supervisionato. Questo schema rientra nella formulazione classica dei problemi di classificazione in apprendimento supervisionato (5).

A differenza dell'apprendimento non supervisionato, in cui l'algoritmo analizza dati privi di etichette, qui il modello dispone di una variabile target nota, ossia la colonna cluster_id. Il compito del classificatore diventa quindi apprendere la relazione tra le feature (popolarità, voto medio, numero di voti, durata e generi codificati) e il cluster assegnato al film, in modo da poter generalizzare questa relazione su nuovi esempi mai visti prima (2).

Il dataset utilizzato in questa fase è il file movies_with_clusters.csv, generato al termine del clustering. Tramite la funzione **split_features_target()** il dataset viene suddiviso nelle variabili di input X, contenenti tutte le colonne numeriche e binarie relative alle caratteristiche dei film, e nella variabile di output Y, corrispondente all'etichetta di cluster.

Per valutare correttamente il comportamento dei modelli anche su dati mai visti prima, il dataset viene suddiviso in due parti tramite la funzione **train_test_split_ds()**:

- training set (80% dei dati), utilizzato per addestrare i modelli.
- test set (20% dei dati), utilizzato esclusivamente per la valutazione finale.

La suddivisione viene effettuata in modo casuale ma ripetibile mediante il parametro *random_state=42*, garantendo coerenza nei risultati tra esecuzioni diverse.

Questa struttura consente pertanto ai modelli supervisionati di apprendere correttamente i pattern presenti nei dati, riducendo il rischio di adattarsi eccessivamente agli esempi utilizzati per l'addestramento.

3.2 Modelli utilizzati

In questa fase del progetto sono stati adottati tre diversi modelli di classificazione, scelti per la loro semplicità, interpretabilità e capacità di gestire dati eterogenei come quelli presenti nel dataset dei film da noi preso in considerazione.

Tutti i modelli sono stati implementati tramite la libreria *scikit-learn*, che fornisce strumenti affidabili per l'apprendimento supervisionato. I tre classificatori scelti — Decision Tree, Random Forest e Regressione Logistica — permettono inoltre di confrontare approcci diversi, rispettivamente basati su strutture gerarchiche, metodi ensemble e modelli lineari.

Ogni modello è stato addestrato utilizzando lo stesso training set ottenuto dalla suddivisione del dataset, così da garantire un confronto equo tra le loro prestazioni.

3.2.1 Decision Tree

Il Decision Tree è un classificatore che organizza il processo decisionale in una struttura ad albero. Ogni nodo interno rappresenta una condizione su una feature del dataset, mentre le foglie corrispondono alle classi finali, in questo caso i cluster individuati nella fase non supervisionata. L'albero viene costruito in modo ricorsivo, selezionando a ogni passo la suddivisione che rende i gruppi risultanti il più omogenei possibile rispetto all'etichetta di classe, secondo criteri come l'impurity o l'information gain (2).

Nel codice, il modello è implementato tramite **DecisionTreeClassifier()** con random_state = 42, parametro utilizzato per ottenere risultati ripetibili. La funzione addestra il modello sui dati di training tramite il metodo **fit()** e successivamente genera le predizioni sul test set usando **predict()**. In uscita restituisce sia il modello addestrato sia il vettore delle predizioni, che verranno poi utilizzati per la valutazione delle prestazioni.

3.2.2 Random Forest

La Random Forest è un modello ensemble basato sull'aggregazione di numerosi alberi decisionali addestrati su sottoinsiemi casuali dei dati e/o delle feature. Ogni albero produce una propria predizione e il risultato finale viene ottenuto aggregando le decisioni dei singoli modelli, tipicamente tramite voto di maggioranza. Questo approccio consente di ridurre la varianza e il rischio di overfitting tipico del singolo albero, migliorando la stabilità e la robustezza del classificatore (5).

Nel progetto la Random Forest è implementata tramite **RandomForestClassifier()** con random_state = 42, così da garantire stabilità dei risultati. Senza modificare altri iperparametri, il modello sfrutta la configurazione predefinita della libreria, che si dimostra già efficace per dataset di questo tipo.

3.2.3 Regressione Logistica

La Regressione Logistica appartiene alla famiglia dei modelli lineari ed è utilizzata per problemi di classificazione, anche nel caso multiclasse. L'idea di base è associare a ogni classe una funzione lineare delle feature e utilizzare una funzione di attivazione (nel caso multiclasse, la softmax) per trasformare tali valori in probabilità di appartenenza a ciascuna classe. La classe predetta è quella con probabilità massima (2).

Nel codice il modello è implementato tramite LogisticRegression() con random_state = 42 e un numero massimo di iterazioni esteso (max_iter = 5000) per assicurare la convergenza dell'algoritmo. Il modello viene valutato su più esecuzioni indipendenti del processo di training e test, al fine di analizzarne la stabilità delle prestazioni.

Essendo un modello lineare, la regressione logistica può incontrare difficoltà quando i dati non sono linearmente separabili, come spesso accade nei cluster ottenuti tramite KMeans.

3.3 Valutazione dei modelli

La valutazione dei modelli di classificazione è stata condotta utilizzando un approccio basato su train/test split, in cui il dataset è stato suddiviso in un training set (80%) e un test set (20%). L'obiettivo è misurare la capacità dei modelli di generalizzare correttamente su dati mai visti, assegnando il cluster corretto a ciascun film.

Per evitare valutazioni dipendenti da una singola esecuzione e garantire risultati più affidabili, l'intero processo di addestramento e valutazione è stato ripetuto più volte, mantenendo fissa la strategia di suddivisione dei dati. I risultati finali sono quindi espressi in termini di media e deviazione standard delle metriche di valutazione.

Sono state utilizzate le seguenti metriche:

- **Accuracy**
- **F1-score macro**
- **F1-score weighted**
- **Matrici di confusione**

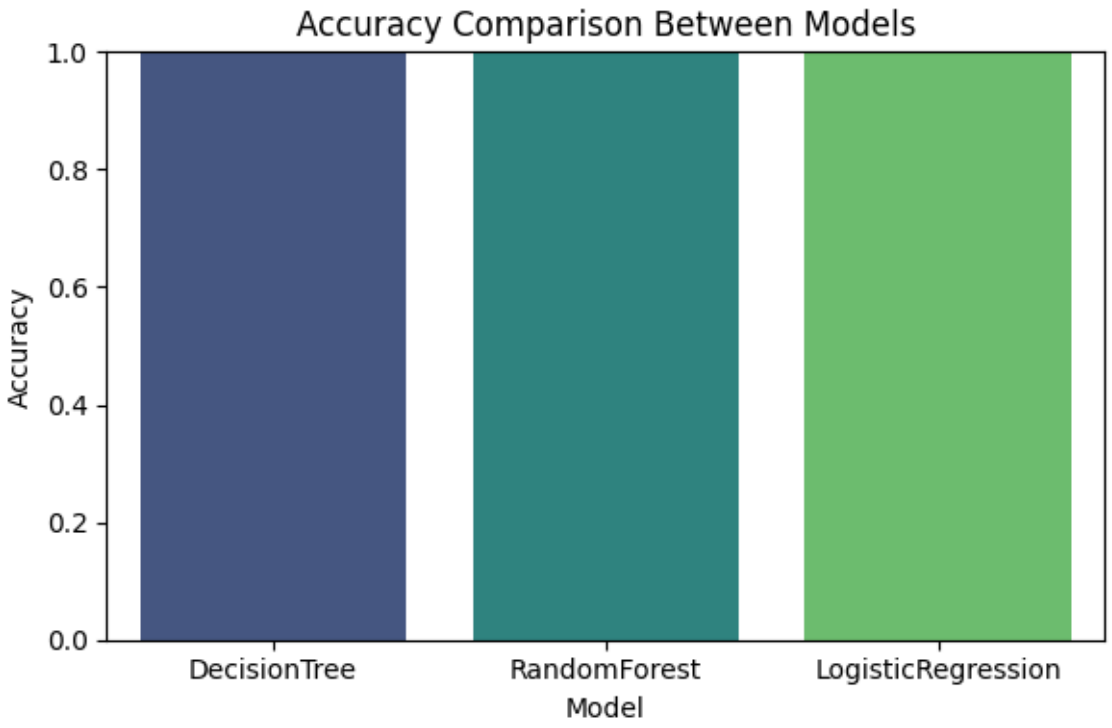
I valori medi e le relative deviazioni standard delle metriche sono stati calcolati automaticamente dal sistema e salvati nel file results/supervised/summary_mean_std.csv, permettendo un confronto quantitativo e stabile tra i diversi modelli supervisionati.

3.3.1 Accuracy

L'accuracy rappresenta la proporzione di predizioni corrette effettuate dal modello rispetto al totale degli esempi presenti nel test set. Si tratta di una metrica semplice ma efficace, particolarmente utile quando le classi non sono fortemente sbilanciate, come nel caso dei cluster generati nella fase di apprendimento non supervisionato.

Nel progetto, il valore di accuracy è stato calcolato per ciascuno dei tre modelli supervisionati **—** Decision Tree**,** Random Forest e Regressione Logistica — tramite la funzione **evaluate_model()**, che restituisce anche la matrice di confusione e il report di classificazione. Per facilitare il confronto tra i modelli, i valori ottenuti vengono salvati all'interno della cartella results/supervised/ e rappresentati graficamente tramite un barplot.

Il grafico riportato di seguito mostra la comparazione delle accuracy dei tre modelli:



Dal grafico emerge che i tre modelli raggiungono valori di accuracy molto simili, prossimi all'unità. In particolare, il Decision Tree ottiene la migliore performance media, seguito dalla Random Forest, mentre la Regressione Logistica presenta valori leggermente inferiori. Le differenze risultano tuttavia contenute, come evidenziato dalle basse deviazioni standard, indicando una sostanziale stabilità delle prestazioni.

3.3.2 F1 - score

Oltre all'accuracy, la valutazione delle prestazioni dei modelli è stata estesa al calcolo dell'F1-score, una metrica che combina precision e recall in un'unica misura ed è particolarmente indicata nei contesti di classificazione multiclasse.

In particolare, sono state considerate due varianti della metrica:

- **F1 macro**, che calcola la media dell'F1-score sulle singole classi assegnando lo stesso peso a ciascun cluster;
- **F1 weighted**, che pesa il contributo di ciascuna classe in base alla numerosità del cluster di appartenenza.

L'utilizzo congiunto di queste due misure consente di valutare il comportamento dei modelli sia in termini di equità tra le classi, sia tenendo conto della distribuzione reale dei cluster nel dataset.

I risultati ottenuti mostrano valori di F1-score molto elevati per tutti i modelli considerati, in linea con le accuracy osservate. In particolare, il Decision Tree ottiene i valori medi di F1-score più elevati, seguito dalla Random Forest, mentre la Regressione Logistica presenta prestazioni leggermente inferiori. Anche in questo caso, le differenze risultano contenute e accompagnate da deviazioni standard ridotte, indicando una buona stabilità dei modelli.

Come anticipato in precedenza, i valori riportati nella tabella sottostante sono stati calcolati automaticamente dal sistema e salvati nel file summary_mean_std.csv, generato al termine di più esecuzioni indipendenti del processo di training e valutazione. Per ciascun modello vengono quindi riportate accuracy media, F1-score macro medio e F1-score weighted medio, insieme alle rispettive deviazioni standard, al fine di valutare non solo le prestazioni medie ma anche la stabilità dei modelli.

model	accuracy_mean	accuracy_std	f1_macro_mean	f1_macro_std	f1_weighted_mean	f1_weighted_std
DecisionTree	0.9996088357109987	0.0005013450673062969	0.9995814942550059	0.0005309414801590653	0.9996087148030103	0.0005016131260630857
LogisticRegression	0.9983893235158767	0.00088959564377202	0.9982729249941465	0.0009789835468426065	0.9983892642247341	0.0008897407259000795
RandomForest	0.9992406810860561	0.0007797598161600688	0.9991874159096534	0.0008526465398165277	0.9992405859175555	0.0007797692140230984

3.3.3 Matrici di confusione

Le matrici di confusione riportate di seguito fanno riferimento a una singola esecuzione rappresentativa e vengono utilizzate esclusivamente a fini interpretativi. La valutazione quantitativa delle prestazioni dei modelli è invece basata sulle metriche aggregate (media e deviazione standard) calcolate su più run indipendenti, come discusso nella sezione precedente.

Per comprendere nel dettaglio le prestazioni dei modelli, oltre all'accuracy è fondamentale analizzare le matrici di confusione, che mostrano come le predizioni si distribuiscono tra le varie classi. Questo permette di individuare quali cluster vengono riconosciuti con maggiore precisione, quali risultano più difficili da distinguere e se esistono pattern di errore sistematici (2).

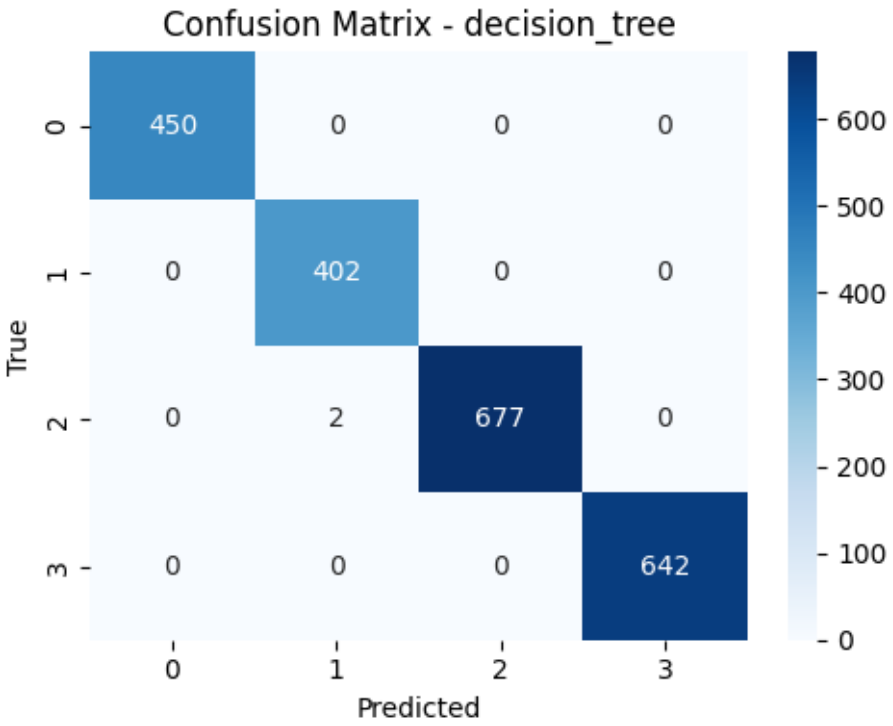
Nel contesto di questo progetto, le classi corrispondono ai cluster generati con K-Means, quindi valutare la matrice di confusione consente di verificare quanto efficacemente i modelli riescano a recuperare la struttura individuata dal clustering.

Matrice di confusione - Decision Tree

La matrice di confusione del Decision Tree mostra prestazioni complessivamente buone, con la maggior parte delle predizioni correttamente posizionate lungo la diagonale principale.

Le osservazioni più rilevanti sono:

- Il cluster 0 viene classificato correttamente in tutti i 450 casi.
- Il cluster 1 è anch'esso riconosciuto con elevata precisione (402 istanze corrette su 402).
- Il cluster 2 presenta un piccolo errore: 2 film appartenenti a questa classe vengono scambiati con il cluster 1.
- Il cluster 3 viene classificato senza errori (642 istanze corrette).



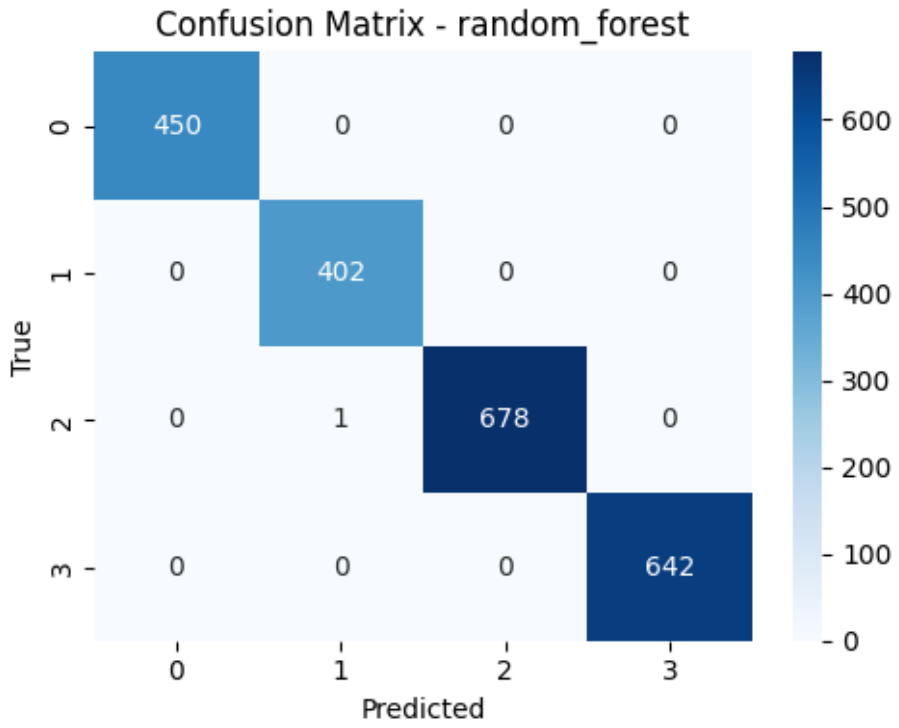
Nel complesso, la matrice di confusione evidenzia prestazioni molto elevate del modello. L'unica criticità riguarda il cluster 2, che risulta leggermente più difficile da distinguere dal cluster 1, come mostrato dalle poche misclassificazioni osservate. Questo comportamento è coerente con il funzionamento degli alberi decisionali, che basano le proprie decisioni su regole semplici e soglie nette sulle feature, risultando meno efficaci quando le classi presentano caratteristiche parzialmente sovrapposte.

Matrice di confusione - Random Forest

La matrice di confusione della Random Forest evidenzia prestazioni molto elevate, con la quasi totalità delle predizioni correttamente collocate lungo la diagonale principale.

Le osservazioni più significative sono:

- Il cluster 0 viene classificato correttamente in tutti i 450 casi.
- Il cluster 1 è riconosciuto quasi perfettamente, con 402 istanze corrette su 403.
- Il cluster 2 è anch'esso identificato con grande precisione (678 predizioni corrette su 679).
- Il cluster 3 viene riconosciuto senza alcun errore (642 istanze corrette).



La Random Forest mostra una capacità di classificazione molto elevata e un numero estremamente ridotto di errori. Rispetto al singolo albero decisionale, il modello beneficia della combinazione delle predizioni di più alberi, che consente di ridurre la sensibilità a suddivisioni locali meno efficaci e di ottenere un comportamento più stabile.

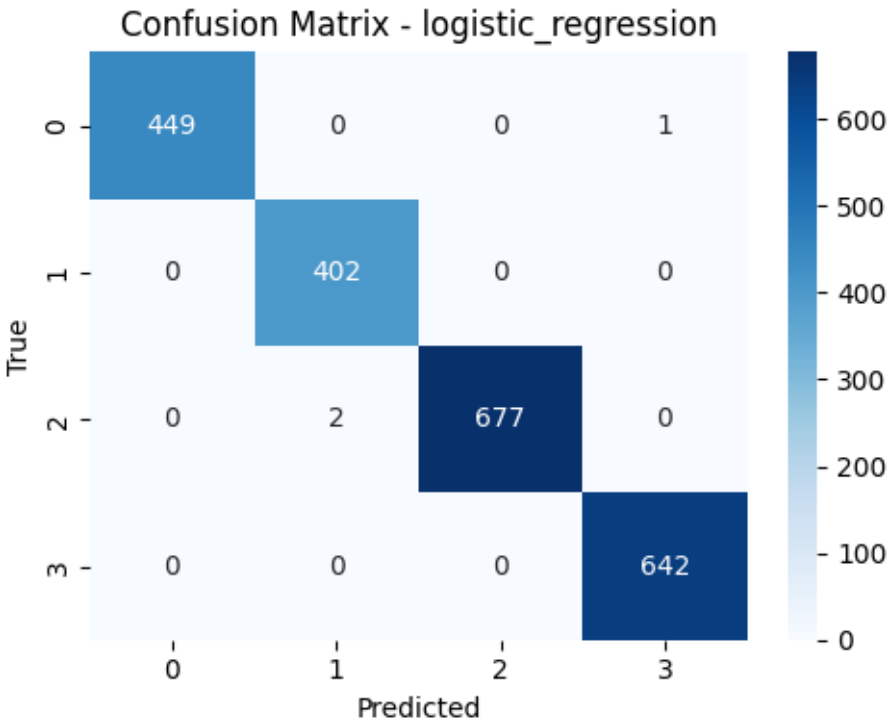
Pur non risultando il modello con le metriche medie più elevate in questo specifico esperimento, la Random Forest conferma un'elevata robustezza e una buona capacità di separazione tra i cluster, con un'unica misclassificazione osservata tra i cluster 1 e 2 come per il Decision Tree.

Matrice di confusione - Regressione Logistica

La matrice di confusione della Regressione Logistica mostra prestazioni buone anche se leggermente inferiori rispetto ai modelli basati su alberi.

I risultati principali sono:

- Il cluster 0 viene classificato quasi completamente in modo corretto, con un unico errore verso il cluster 3.
- Il cluster 1 è riconosciuto perfettamente, con 402 predizioni corrette.
- Il cluster 2 presenta due errori: 2 film vengono scambiati con il cluster 1.
- Il cluster 3 viene classificato senza errori (642 istanze corrette)



Pur mantenendo una buona accuratezza generale, la Regressione Logistica mostra difficoltà in alcuni casi specifici, in particolare nella distinzione tra il cluster 1 e il cluster 2. Questo accade perché il modello utilizza confini decisionali lineari, che non sempre riescono a separare correttamente classi che presentano strutture più complesse o parzialmente sovrapposte.

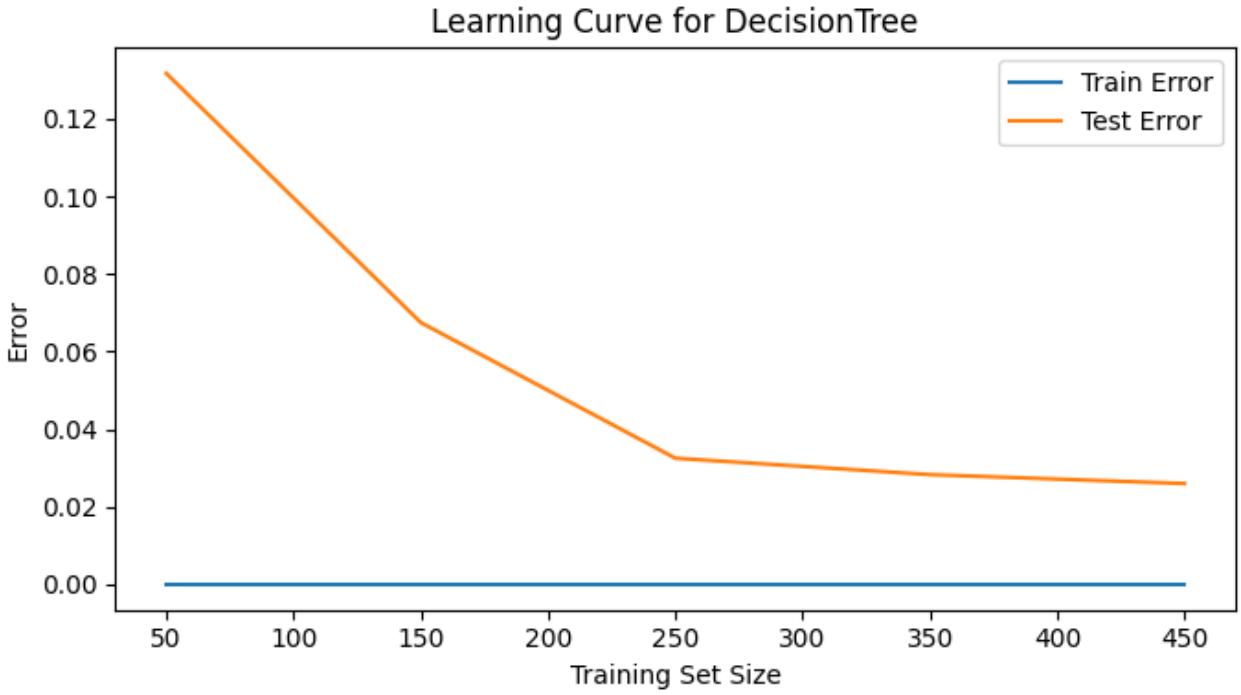
Gli errori osservati riflettono proprio questo limite: in aree del dataset dove i cluster hanno pattern simili, il modello fatica a identificare una separazione netta. Nonostante ciò, la regressione logistica offre un comportamento stabile e prevedibile, con un numero limitato di misclassificazioni.

3.3.4 Analisi delle curve di apprendimento

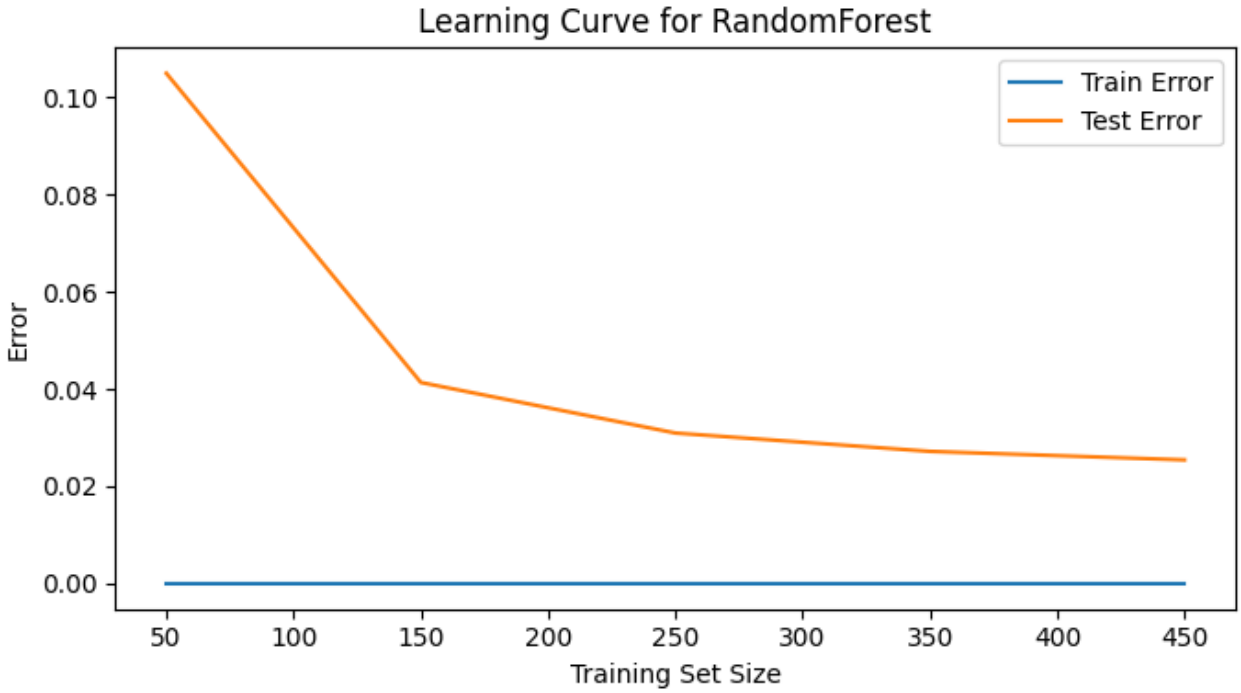
Per analizzare il comportamento dei modelli al variare della quantità di dati disponibili, sono state generate le curve di apprendimento (learning curves) per ciascun classificatore. Le curve riportano l'andamento dell'errore di training e dell'errore di validazione al crescere della dimensione del training set, consentendo di valutare la capacità di generalizzazione dei modelli e l'eventuale presenza di overfitting.

Il Decision Tree mostra un errore di training prossimo allo zero per tutte le dimensioni del dataset, accompagnato da un errore di test inizialmente elevato che decresce con l'aumentare dei dati. Questo

comportamento evidenzia una tendenza al sovra-adattamento, tipica dei modelli ad alta varianza.

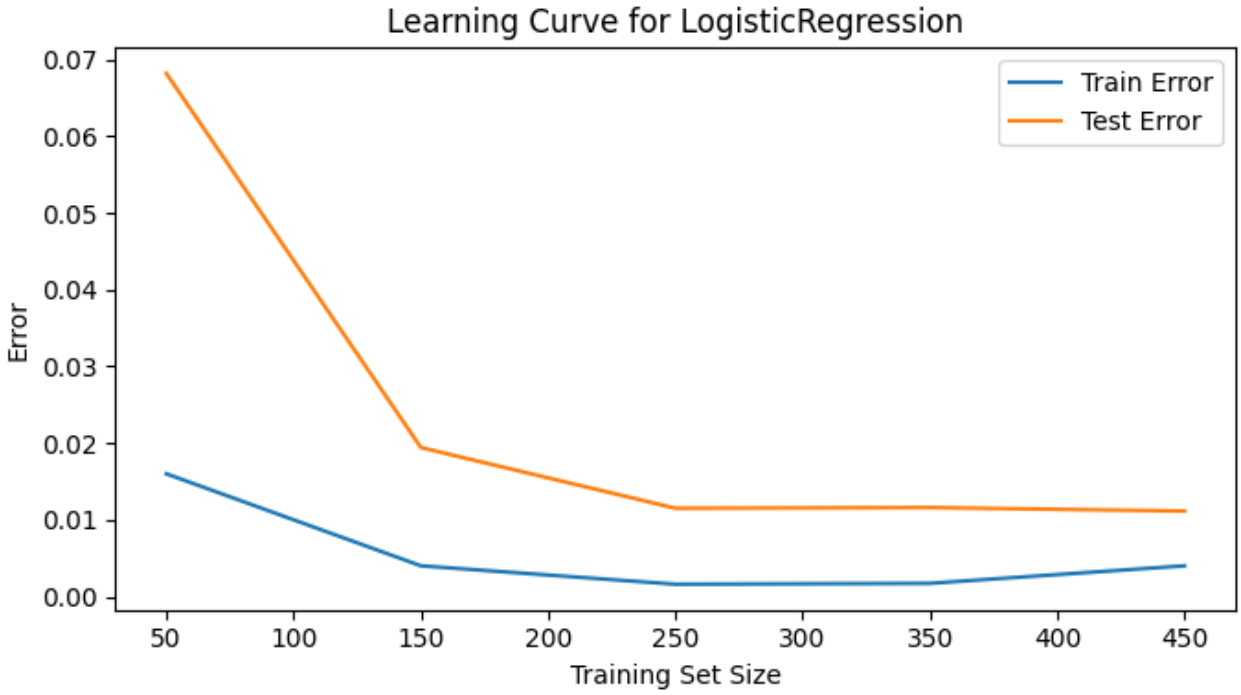


La Random Forest presenta un andamento più stabile, con un errore di test sistematicamente inferiore rispetto al Decision Tree. L'aggregazione di più alberi consente infatti di ridurre la varianza e migliorare la capacità di generalizzazione.



La Regressione Logistica mostra curve di training e test più ravvicinate, indicando un modello più regolarizzato e meno soggetto a overfitting. Tuttavia, l'errore residuo più elevato riflette la limitata capacità del modello di rappresentare confini decisionali complessi, coerentemente con la sua

natura lineare.



Nel complesso, le curve di apprendimento confermano quanto osservato tramite le metriche quantitative e le matrici di confusione, rafforzando la scelta della Random Forest come modello più efficace per questo dataset.

3.3.5 Confronto tra modelli

Dall'analisi complessiva delle metriche e delle matrici di confusione emerge che tutti i modelli supervisionati ottengono prestazioni molto elevate, risultando in grado di apprendere efficacemente le etichette derivate dal clustering.

Il Decision Tree fornisce le migliori prestazioni medie, pur mostrando una lieve sensibilità nelle regioni di confine tra cluster con caratteristiche sovrapposte. La Random Forest presenta un comportamento più stabile e robusto, con un numero minimo di misclassificazioni, mentre la Regressione Logistica risulta meno adatta alla struttura non lineare dei cluster, pur mantenendo risultati complessivamente soddisfacenti.

Capitolo 4 - Ragionamento probabilistico e Bayesian Network

4.1 Introduzione alle Bayesian Network

Le Bayesian Network sono modelli probabilistici grafici che rappresentano variabili casuali e le loro dipendenze tramite un grafo diretto aciclico (DAG). Ogni nodo corrisponde a una variabile, mentre gli archi codificano relazioni di dipendenza che consentono di sfruttare esplicitamente le indipendenze condizionali presenti nel dominio (6).

Nel contesto di questo progetto, le Bayesian Network sono utilizzate come strumento di ragionamento probabilistico per integrare le categorie discrete ottenute dal clustering con una gestione esplicita dell'incertezza. L'inferenza sulle variabili di interesse viene effettuata tramite l'algoritmo di Variable Elimination, un metodo di inferenza esatta che sfrutta la struttura del grafo per calcolare distribuzioni posteriori a partire da evidenze parziali (4).

4.2 Preparazione dei dati e costruzione della rete

Poiché la maggior parte degli algoritmi di apprendimento per Bayesian Network richiede variabili discrete, le feature numeriche del dataset sono state discretizzate tramite la funzione **discretize_features()**. In particolare:

- *popularity* e *vote_average* → {low, medium, high}
- *runtime* → {short, medium, long}

Questa trasformazione riduce la complessità del modello rendendo più stabile l'apprendimento delle tabelle di probabilità condizionate (4).

La struttura del modello viene definita esplicitamente tramite la funzione **define_structure(),** che stabilisce i seguenti archi:

- cluster_id → popularity_disc
- cluster_id → voteAvg_disc
- cluster_id → runtime_disc

Questa scelta riflette un modello generativo semplice, in cui il cluster individuato tramite K-Means viene interpretato come variabile discreta derivata automaticamente dalla fase di clustering, utilizzata come variabile generativa nel modello probabilistico.

Si sottolinea che tale interpretazione ha valore puramente modellistico: il cluster_id non rappresenta una causa reale, ma una variabile latente derivata automaticamente dai dati, utilizzata come nodo generativo per rendere esplicite e interpretabili le dipendenze probabilistiche tra le feature.

Questa impostazione consente di interpretare le probabilità apprese come una descrizione compatta e probabilistica delle proprietà tipiche di ciascun cluster, rendendo il modello facilmente interpretabile e coerente con le categorie individuate nella fase di clustering.

Dal punto di vista implementativo, **define_structure()** restituisce una lista di coppie ordinate (genitore, figlio) che specificano quali CPD devono essere apprese e l'ordine di propagazione delle dipendenze durante l'inferenza.

Dopo aver definito la struttura, la rete viene costruita con DiscreteBayesianNetwork(edges) e addestrata tramite il metodo **fit()**, che apprende automaticamente le CPD dai dati discretizzati. Poiché la struttura è già nota, il processo si riduce al solo parameter learning, rendendo il modello semplice e computazionalmente efficiente.

A differenza di approcci che tentano di apprendere la struttura della rete a partire da variabili continue, il progetto adotta una discretizzazione preventiva delle feature e una struttura fissata a priori. Questa scelta evita i problemi tipici delle reti bayesiane con variabili continue, come l'esplosione delle CPD e l'impossibilità di gestire valori mai osservati durante l'addestramento.

4.3 Inferenza probabilistica e risultati ottenuti

Una volta appresa la rete, è possibile stimare la probabilità di una variabile dato un insieme di evidenze. Per farlo il progetto utilizza il metodo di inferenza Variable Elimination, disponibile nella libreria pgmpy.

La query considerata è:

$$P(\text{cluster\_id} \mid \text{popularity\_disc} = low, \text{runtime\_disc} = short)$$

che permette di determinare quale cluster risulti più probabile per un film poco popolare e di durata breve. La query considerata simula un caso in cui il sistema dispone di evidenze incomplete e permette di analizzare il comportamento della rete bayesiana nel determinare il cluster più probabile sulla base di informazioni parziali.

L'output fornito dal modello è una distribuzione di probabilità sui quattro cluster. Questa informazione è utile sia per assegnare film a categorie non viste durante l'addestramento, sia per interpretare il ruolo delle variabili e verificare la coerenza tra clustering preliminare e struttura probabilistica appresa.

Un vantaggio rilevante delle Bayesian Network è la capacità di produrre stime anche quando alcune informazioni sono mancanti: l'incertezza viene propagata nella rete e la predizione finale rimane coerente con la conoscenza codificata (6). Il risultato dell'inferenza viene infine salvato nel file:

results/bayes/inference_result.txt

consentendo di documentare e confrontare in modo chiaro diversi esperimenti.

A differenza dei modelli supervisionati presentati nel Capitolo 3, la rete bayesiana non viene utilizzata come classificatore alternativo né valutata tramite metriche di accuratezza. Il suo ruolo all'interno del sistema è quello di fornire un livello di ragionamento probabilistico interpretabile, capace di gestire esplicitamente l'incertezza e di supportare l'analisi delle relazioni tra cluster e feature in presenza di evidenze parziali.

Il modulo di ragionamento probabilistico completa il sistema integrando i risultati del clustering e dell'apprendimento supervisionato, prepararando le informazioni per la successiva fase di rappresentazione simbolica e di ragionamento logico basata su knowledge base.

Capitolo 5 – Ragionamento Logico e Prolog

5.1 Integrazione del ragionamento logico nel sistema di conoscenza

Nel progetto il ragionamento logico viene utilizzato come supporto alle fasi di apprendimento automatico e di ragionamento probabilistico, con l'obiettivo di rendere i risultati ottenuti più facilmente interrogabili e interpretabili. I modelli di machine learning e la Bayesian Network lavorano infatti su rappresentazioni numeriche e producono output sotto forma di etichette o probabilità, che risultano poco adatte a un'interazione diretta con l'utente o a interrogazioni basate su concetti del dominio.

L'uso di Prolog consente di colmare questo divario introducendo una rappresentazione simbolica della conoscenza, in cui i film sono identificati esplicitamente tramite il loro titolo e associati ai cluster appresi nelle fasi precedenti. In questo modo è possibile formulare query dichiarative che facciano riferimento a entità comprensibili, come singoli film o gruppi di film appartenenti alla stessa categoria.

Il ragionamento logico non opera quindi sui dati grezzi, ma su informazioni già elaborate dai modelli di apprendimento. I cluster individuati automaticamente vengono trattati come conoscenza consolidata e resi accessibili tramite una Knowledge Base, separando chiaramente la fase di apprendimento dalla fase di utilizzo della conoscenza. (7)

In questo contesto Prolog non viene impiegato come semplice archivio di dati, ma come strumento per interrogare e combinare in modo simbolico i risultati prodotti dai modelli statistici, completando il sistema con un livello di ragionamento esplicito e interpretabile.

5.2 Costruzione della Knowledge Base

La Knowledge Base utilizzata per il ragionamento logico viene costruita a partire dal file movies_with_clusters_prolog.csv, che contiene, oltre alle feature numeriche e all'attributo cluster_id, anche il titolo originale del film (original_title). Questo dataset rappresenta il collegamento tra le fasi di apprendimento automatico e il livello di ragionamento simbolico, poiché associa a ciascun film una categoria appresa automaticamente e un identificatore testuale utilizzabile nelle query logiche.

Dal punto di vista implementativo, il file CSV viene letto dal modulo dedicato al ragionamento logico e utilizzato per popolare la Knowledge Base con informazioni sui film e sulla loro appartenenza ai cluster. Ogni film viene quindi rappresentato come un'entità del dominio, a cui è associato il cluster individuato nella fase di clustering non supervisionato.

La scelta di mantenere una rappresentazione semplice della Knowledge Base è intenzionale. L'obiettivo non è costruire un'ontologia complessa, ma rendere disponibili in forma simbolica i risultati del processo di apprendimento, così da poterli interrogare in modo dichiarativo. I cluster, pur essendo ottenuti tramite un algoritmo statistico, vengono trattati come categorie discrete e stabili, risultando adatti a essere utilizzati nel ragionamento logico.

La semplicità della Knowledge Base non implica un uso di Prolog come database relazionale: le regole definite introducono un livello di astrazione che consente di interrogare e combinare i risultati dell'apprendimento automatico in modo dichiarativo, separando i concetti del dominio dalla loro rappresentazione numerica.

L’inclusione del titolo del film svolge un ruolo centrale: consente di riferirsi direttamente alle istanze del dominio in modo chiaro e semanticamente significativo. Senza questa informazione, l’interazione con il sistema sarebbe limitata a identificatori numerici, riducendo il valore aggiunto del ragionamento logico rispetto alle altre componenti del progetto.

Nel complesso, la Knowledge Base funge da livello di collegamento tra i modelli statistici e l’interrogazione simbolica, permettendo di accedere e interpretare la conoscenza appresa dal sistema in modo più diretto e comprensibile.

5.3 Utilizzo del Ragionamento Logico

Il ragionamento logico non introduce nuove categorie né modifica i risultati prodotti dai modelli di apprendimento automatico, ma consente di utilizzare tali risultati in modo dichiarativo. In particolare, le regole definite permettono di collegare i titoli dei film alle categorie apprese, rendendo possibile un accesso simbolico alla conoscenza senza dover interagire con le rappresentazioni numeriche.

A differenza delle fasi precedenti, in cui il focus è sull’apprendimento e sulla predizione, il ragionamento logico è orientato all’esplorazione e all’interrogazione della conoscenza. Le query consentono quindi di verificare l’appartenenza di specifici film ai cluster individuati e di analizzare il contenuto delle categorie apprese in termini di istanze concrete del dominio.

In questo modo, Prolog viene utilizzato come strumento di supporto all’interpretazione dei risultati del sistema, fornendo una vista simbolica e spiegabile delle categorie ottenute tramite i modelli statistici.

Pur non modificando le categorie apprese, il ragionamento logico consente di combinare e filtrare le informazioni disponibili secondo criteri simbolici, rendendo esplicite relazioni e proprietà che non emergono direttamente dalle rappresentazioni numeriche.

5.4 Integrazione dei modelli nel sistema di conoscenza

Il sistema sviluppato integra modelli di apprendimento automatico, ragionamento probabilistico e ragionamento logico all’interno di una pipeline coerente. Il clustering non supervisionato individua categorie latenti di film, che vengono successivamente utilizzate sia come etichette per l’apprendimento supervisionato sia come variabile centrale nella rete bayesiana.

I risultati ottenuti vengono infine resi accessibili a livello simbolico tramite Prolog, permettendo di interrogare le categorie apprese facendo riferimento a entità concrete del dominio.

Questa integrazione consente di sfruttare in modo complementare approcci statistici e simbolici, separando chiaramente la fase di apprendimento dai dati dalla fase di utilizzo e interpretazione della conoscenza.

5.5 Conclusioni e sviluppi futuri

In questo progetto è stato realizzato un sistema che integra tecniche di apprendimento automatico, ragionamento probabilistico e ragionamento logico per l’analisi di un dataset di film. Il clustering non

supervisionato ha permesso di individuare gruppi di film con caratteristiche simili senza l'uso di etichette predefinite, fornendo una prima organizzazione dei dati guidata esclusivamente dalle feature disponibili.

I cluster ottenuti sono stati successivamente utilizzati come etichette per l'apprendimento supervisionato, consentendo di valutare quanto tali categorie fossero riproducibili da modelli di classificazione differenti. I risultati mostrano che modelli più complessi, come la Random Forest, riescono a rappresentare meglio la struttura dei dati rispetto a modelli più semplici o lineari.

L'introduzione della rete bayesiana ha permesso di affrontare il problema da un punto di vista probabilistico, modellando in modo esplicito l'incertezza e consentendo inferenze anche in presenza di informazioni parziali. Infine, il ragionamento logico ha reso i risultati più facilmente interrogabili, separando la fase di apprendimento dalla fase di utilizzo della conoscenza attraverso una rappresentazione simbolica.

Nel complesso, il progetto evidenzia come l'integrazione di approcci statistici e simbolici consenta una comprensione più flessibile e strutturata del dominio rispetto all'utilizzo di una singola tecnica.

Il lavoro svolto può essere ulteriormente esteso sotto diversi aspetti. Una prima possibile evoluzione riguarda l'introduzione di tecniche di validazione incrociata, che potrebbero affiancare l'attuale valutazione basata su più esecuzioni indipendenti, rendendo l'analisi delle prestazioni ancora più robusta.

Dal punto di vista del ragionamento probabilistico, la rete bayesiana potrebbe essere arricchita includendo nuove variabili o sperimentando l'apprendimento automatico della struttura, valutandone l'impatto sulla complessità del modello e sulla qualità dell'inferenza. Un'altra estensione possibile consiste in una maggiore integrazione tra rete bayesiana e ragionamento logico, utilizzando i risultati dell'inferenza probabilistica come supporto alle interrogazioni simboliche.

Infine, il sistema potrebbe essere esteso verso un utilizzo più applicativo, ad esempio permettendo all'utente di esplorare i cluster, confrontare film simili o ottenere suggerimenti basati sulle categorie apprese. Questi sviluppi rappresentano una naturale prosecuzione del lavoro svolto e potrebbero costituire la base per approfondimenti futuri.

Bibliografia

(1) Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

(2) James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R*(2nd ed.). Springer.

(3) McKinney, W. (2017). *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython* (2nd ed.). O'Reilly Media.

(4) Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT Press.

(5) Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

(6) Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

(7) D. Poole, A. Mackworth (2023) *Artificial Intelligence: Foundations of Computational Agents.* Cambridge University Press

(8) Dataset iniziale https://www.kaggle.com/datasets/juzershakir/tmdb-movies-dataset

Ulteriori fonte usate per la realizzazione del codice:

(9) Decision Tree Python https://www.youtube.com/watch?v=PHxYNGo8NcI

(10) Random Forest Python https://www.youtube.com/watch?v=ok2s1vV9XW0

(11) Logistic Regression Python https://www.youtube.com/watch?v=J5bXOOmkopc

(12) Training and Testing Data https://www.youtube.com/watch?v=fwY9Qv96DJY

(13) K Means Clustering Algorithm https://www.youtube.com/watch?v=EItlUEPCIzM