Relazione Caso di Studio Ingegneria Della Conoscenza

Movie Recommender System

Sviluppo di un sistema di classificazione e raccomandazione basato su conoscenza

Giorgia Summa, 775197, g.summa4@studenti.uniba.it

Repository GitHub:

https://github.com/giosumma/NETFLIX-ICON.git

AA 2024-2025

Sommario

1. Introduzione

Questo progetto ha come obiettivo lo sviluppo di un sistema intelligente per l'analisi, la classificazione e la raccomandazione di film basato su un dataset di Netflix. In particolare, il sistema è stato progettato per:

- **Previsione della popolarità:** sviluppo di un modello di classificazione in grado di predire se un film sarà "popolare" o meno, basandosi sulle sue caratteristiche
- **Raccomandazione basata su similarità:** implementazione di un sistema per suggerire film simili a un titolo selezionato dall'utente, sfruttando metriche di vicinanza tra i vettori di rappresentazione dei film.
- **Raggruppamento automatico (Clustering):** organizzazione dei film in cluster omogenei attraverso algoritmi di apprendimento non supervisionato, facilitando l'esplorazione e la comprensione della struttura intrinseca del dataset.

Quindi, il sistema integra moduli basati su apprendimento supervisionato, non supervisionato e similarità, costruendo un sistema ibrido per l'analisi e la raccomandazione di film

Elenco Argomenti di Interesse

1. **Rappresentazione della conoscenza**
   La conoscenza sui film è rappresentata attraverso le caratteristiche estratte dal dataset (come generi, valutazioni, anno di uscita) e le regole per definire la variabile target (ad esempio, un film è "di successo" se ha una valutazione IMDb ≥ 7.0 e almeno 10.000 voti). Questa formalizzazione permette al sistema di lavorare in modo strutturato sui dati.
2. **Ragionamento automatico**
   La fase di classificazione e di raccomandazione utilizza algoritmi che automatizzano il processo decisionale: assegnano etichette ai film (successo/non successo) o suggeriscono film simili basandosi sulle caratteristiche calcolate, mostrando un comportamento di ragionamento automatico basato sui dati.
3. **Apprendimento e gestione dell'incertezza**
   L'uso di modelli di machine learning supervisionati come Random Forest,

K-Nearest Neighbors e Logistic Regression e Decision Tree consentono al sistema di imparare dai dati, gestendo l'incertezza presente nei voti e nelle valutazioni per fare predizioni accurate riguardo al successo dei film.

4. **Integrazione di moduli per funzioni diverse**
   Il progetto integra moduli di clustering (metodi non supervisionati) con moduli di classificazione e raccomandazione (supervisionati e basati su similarità), dimostrando una competenza trasversale nell'assemblaggio di sistemi ibridi complessi per l'analisi dei dati.

1.1 Strumenti utilizzati

Per lo sviluppo di questo progetto è stato scelto il linguaggio di programmazione **Python** (www.python.org), per la sua versatilità e ampia diffusione nell'ambito dell'analisi dei dati e del machine learning.
Il codice è stato ospitato su **GitHub**, una piattaforma che facilita la gestione dei progetti.

1.2 Librerie utilizzate

Per l'analisi, la manipolazione dei dati e lo sviluppo degli algoritmi di raccomandazione e classificazione, sono state impiegate le seguenti librerie:

- **Scikit-learn (sklearn)**
  Scikit-learn è una libreria open source di machine learning per Python. Include una vasta gamma di algoritmi per la **classificazione**, la **regressione** e il **clustering**, come Support Vector Machines (SVM), Regressione Logistica, Naive Bayes, K-Means e DBSCAN. È progettata per integrarsi facilmente con le librerie **NumPy** e **SciPy**, rendendola adatta per applicazioni avanzate di apprendimento automatico.
- **NumPy:** Libreria fondamentale per il calcolo scientifico in Python, offre supporto per array multidimensionali e operazioni matematiche ad alte prestazioni.
- **Pandas**: Pandas è una libreria fondamentale per la manipolazione e l'analisi dei dati. Offre strutture dati flessibili come **DataFrame** e **Series**,

ideali per gestire tabelle di dati eterogenei come quelle presenti nel dataset Netflix, facilitando operazioni di filtraggio, raggruppamento, pulizia e trasformazione.

- **Matplotlib**
  Matplotlib è una libreria per la visualizzazione grafica dei dati. Consente di creare **grafici statici, interattivi e personalizzati** direttamente all'interno di applicazioni Python. È ampiamente usata per rappresentare in modo chiaro le distribuzioni dei dati, i risultati degli algoritmi e le analisi esplorative.
- **Termcolor:** Utilizzata per colorare l'output testuale nella console, migliorando l'interfaccia utente.

2. Preprocessing per il clustering

Per preparare i dati al processo di **clustering**, è stata sviluppata una funzione specifica di preprocessing, finalizzata a strutturare il dataset in modo compatibile con algoritmi di apprendimento non supervisionato. Le operazioni principali svolte sono state le seguenti:

- **Gestione dei valori mancanti**:
  Le colonne numeriche imdbAverageRating, imdbNumVotes e releaseYear sono state completate sostituendo i valori mancanti rispettivamente con la **mediana** o lo **zero**, a seconda del contesto. La colonna type è stata riempita con la stringa "Unknown" dove assente.
- **Pulizia e trasformazione dei generi**:
  La colonna genres, che può contenere più generi separati da virgole, è stata trasformata in una lista (genres_list). A partire da questa lista, sono stati estratti tutti i generi distinti presenti nel dataset.
- **One-hot encoding dei generi**:
  Per ogni genere individuato, è stata creata una nuova **colonna binaria** (ad esempio genre_Drama, genre_Comedy, ecc.) che assume valore 1 se il contenuto appartiene a quel genere, 0 altrimenti. Questo ha permesso di rappresentare i generi in forma numerica e utilizzabile dagli algoritmi di clustering.

- **Codifica della colonna type**:
  Il tipo del contenuto (es. film o serie) è stato convertito in una variabile numerica tramite **Ordinal Encoding**, necessaria per l'elaborazione successiva.
- **Scalatura delle variabili numeriche**:
  Tutte le colonne numeriche (eccetto il titolo e le colonne binarie dei generi) sono state standardizzate mediante **StandardScaler**, per evitare che differenze di scala influenzassero il clustering.
- **Unione delle informazioni**:
  La rappresentazione binaria dei generi (genres) è stata unita alle altre variabili (attributes) mediante il campo title. La funzione restituisce un nuovo dataframe movies con le feature pronte per il clustering.

3. Clustering

Il clustering è un insieme di tecniche che permettono di raggruppare oggetti simili tra loro in classi omogenee, chiamate cluster. Un cluster è formato da elementi che si assomigliano molto, mentre tra cluster diversi gli elementi sono più diversi tra loro.
L'input di un algoritmo di clustering è un insieme di elementi (nel mio caso, film), mentre l'output sono diversi gruppi in cui gli elementi vengono divisi in base a quanto sono simili.

Vantaggi del clustering nel mio progetto

1. **Supporta la Raccomandazione:** avendo implementato un sistema di raccomandazione basato sulla similarità, esso è potenziato e migliorato grazie al clustering:

- I cluster permettono di restringere lo spazio di ricerca dei film simili: invece di confrontare il film target con tutti i film del dataset (costoso e meno preciso), confronti solo con quelli appartenenti allo stesso cluster.
- Questo approccio migliora la precisione (perché i film nel cluster sono già simili) e riduce il tempo computazionale.

- Realizzazione di un modello ibrido che combina due tecniche: prima, l'algoritmo K-means viene utilizzato per suddividere il dataset in gruppi omogenei (cluster) di elementi simili; successivamente, all'interno del cluster selezionato, si applica la similarità del coseno per identificare con maggiore precisione i contenuti più affini a quello scelto.

2. **Favorisce l'Esplorazione del Dataset e l'Analisi della Struttura Latente:** dati molti film con caratteristiche diverse (genere, anno, voti IMDb) il clustering aiuta a individuare gruppi latenti di film con caratteristiche comuni, che possono non essere immediatamente evidenti con una semplice analisi esplorativa.

3.1 K-Means

Nel progetto ho utilizzato l'algoritmo K-Means. Questo algoritmo serve a suddividere i film in k gruppi, basandosi sulle loro caratteristiche (come voto, anno, genere, ecc.).
Il funzionamento di K-Means è iterativo:

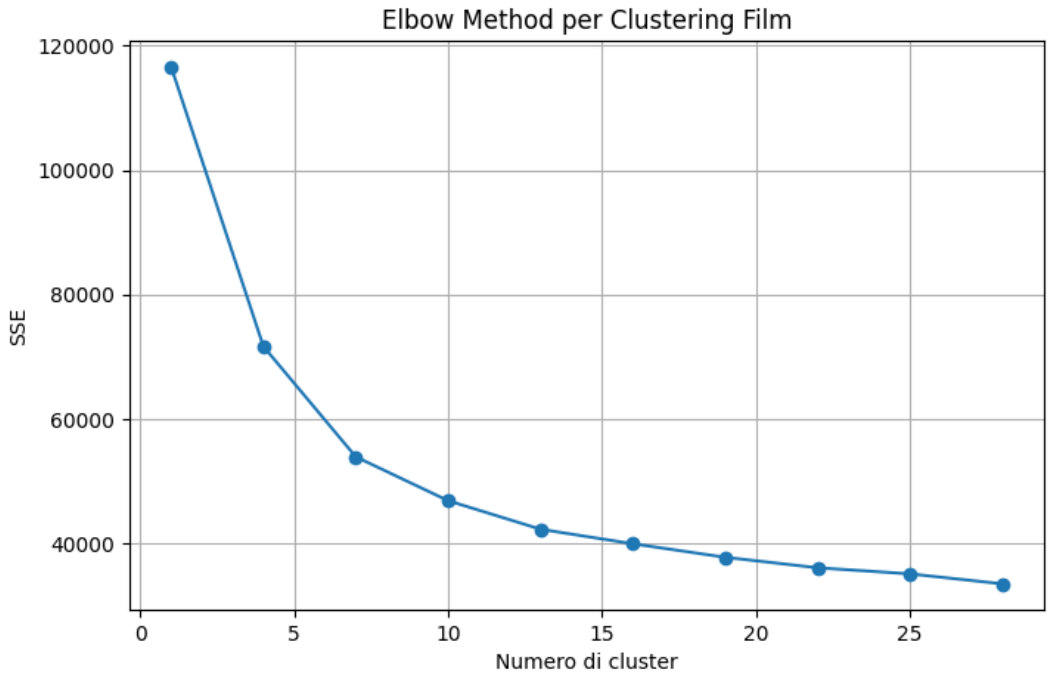
- all'inizio crea k gruppi (cluster) e assegna i film a ciascun gruppo in modo casuale o usando alcune regole;
- calcola il centroide (il punto medio) di ogni gruppo;
- riassegna ogni film al cluster con il centroide più vicino;
- ricalcola i centroidi in base alle nuove assegnazioni;
- ripete questo processo finché i gruppi non cambiano più, cioè quando l'algoritmo converge.

L'algoritmo usa la distanza euclidea per calcolare la vicinanza tra i film e i centroidi dei cluster.

3.2 Elbow Method

Per scegliere il numero ottimale di cluster (il valore di k) ho usato il "Metodo del Gomito" (Elbow Method). Questo metodo è utile perché dà un criterio oggettivo per decidere quanti cluster usare.
Tracciando un grafico con i valori di k sull'asse orizzontale e la somma delle

distanze al quadrato (Sum of Squared Errors, SSE) tra i punti e i loro centroidi sull'asse verticale, si ottiene una curva che tende a decrescere.



Il punto in cui la curva comincia a "piegarsi" (a formare un gomito) indica il numero migliore di cluster perché, dopo quel punto aumentare k porta a un miglioramento minore.
Ho quindi deciso di usare 7 cluster per ottenere una suddivisione più varia e significativa.

L'uso del clustering, combinato con un modello di classificazione, mi ha permesso di migliorare la qualità del sistema di raccomandazione film, ottenendo risultati più accurati e personalizzati.

4. Sistema di Raccomandazione

Dopo aver effettuato il clustering sul dataset dei film, ho sviluppato un sistema di raccomandazione basato sulla similarità tra film, focalizzandomi in particolare sui generi.
Per misurare quanto due film sono simili, ho scelto di usare la **similarità del coseno**, una metrica che calcola il coseno dell'angolo tra due vettori in uno spazio multidimensionale. Più l'angolo è piccolo, maggiore è la similarità tra i film.

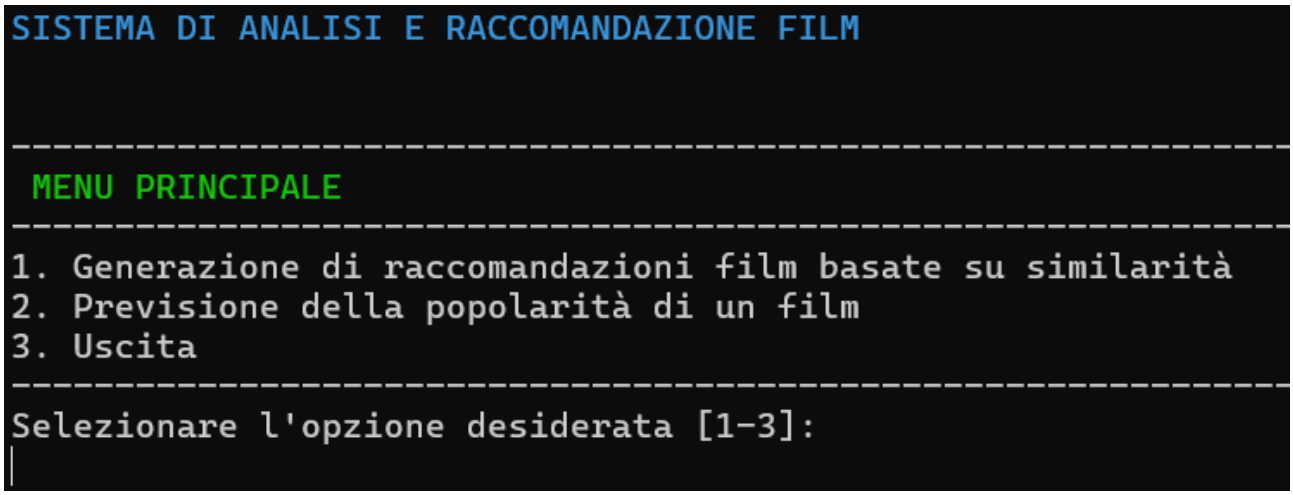
Il sistema di raccomandazione, per suggerire film simili, richiede all'utente di fornire:

- Il titolo del film di riferimento
- Il genere preferito
- Il numero di film simili da consigliare

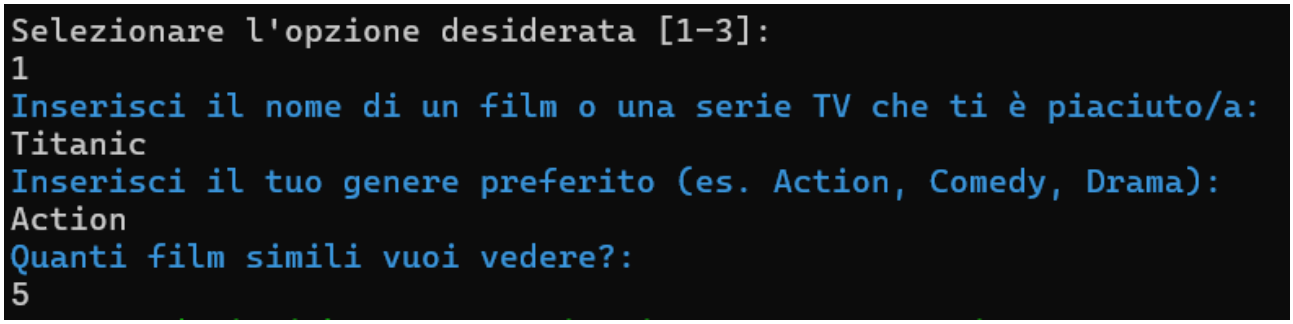
In base a queste informazioni, il sistema cerca i film più simili nel cluster e restituisce una lista personalizzata di suggerimenti.

Principio di Funzionamento nella Raccomandazione:

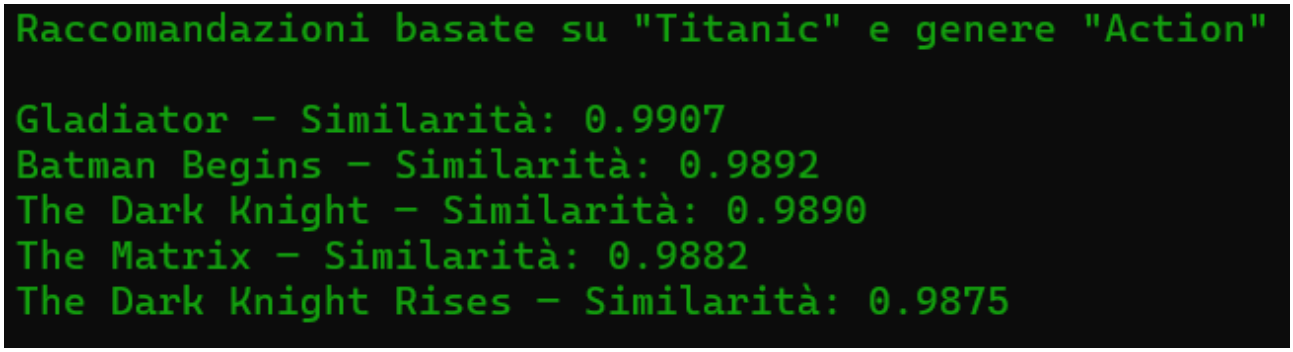
1. Ogni film è rappresentato come un vettore di feature numeriche (es. anno di rilascio, rating, genere, cluster di appartenenza).
2. Data la scelta di un film da parte dell'utente, si calcola la similarità del coseno tra il vettore del film messo in input e i vettori di tutti gli altri film nel dataset.
3. I film vengono ordinati in base al loro punteggio di similarità (dal più alto al più basso).
4. Vengono selezionati i N film con il punteggio di similarità più alto (escludendo il film di input stesso).
5. Un filtro aggiuntivo è applicato per raccomandare film che appartengono a un genere preferito specificato dall'utente.



Visualizzazione del menù principale



Scelta dell'opzione 1: inserito il film "Titanic", genere "Action", e il numero di film simili che vorrei vedere



Stampa delle raccomandazioni basati sia sul film "Titanic", che sul genere "Action"

5. Preprocessing per la classificazione

Per preparare il dataset per la classificazione dei film, ho effettuato alcune operazioni di pulizia e trasformazione:

- Per i valori mancanti in colonne come imdbAverageRating, imdbNumVotes e releaseYear ho usato la mediana o valori di default.
- Per la colonna type e genres ho riempito i valori mancanti con "Unknown".
- Ho creato una nuova colonna chiamata successful che indica se un film è "di successo": un film è considerato di successo se ha una valutazione media IMDb maggiore o uguale a 7.0 **e** almeno 10.000 voti.

6. Classificazione

La classificazione è una tecnica di Machine Learning che permette di assegnare una classe o etichetta a un nuovo elemento basandosi su un modello creato da

dati già noti (training set). Nel progetto, ho voluto classificare i film in base al loro successo, cioè predire se un film sarà/è popolare o meno.

Ho diviso il dataset in due parti:

- **Training set** (80% dei dati) per addestrare il modello
- **Test set** (20% dei dati) per valutare l'accuratezza del modello

La variabile target da predire è la colonna successful.

I modelli di classificazione che ho usato sono:

- **Random Forest Classifier**
- **K-Nearest Neighbors (KNN)**
- **Decision Tree Classifier**
- **Logistic Regression**

Definizione di "Successo"

Come specificato in precedenza, un film è etichettato come "successful" (di successo) se soddisfa due criteri combinati:

- imdbAverageRating (valutazione media IMDb) ≥7.0
- imdbNumVotes (numero di voti IMDb) ≥10000

Questa definizione binaria trasforma il problema di previsione della popolarità in un task di classificazione binaria.

6.1 K-Nearest Neighbors (KNN)

KNN è un algoritmo semplice e efficace. Classifica un nuovo film osservando i *k* film più simili nel dataset (i "vicini") e assegna la classe più comune tra questi vicini. Nel mio caso, la similarità si basa sulle caratteristiche del film (come generi, valutazione, ecc.). Così, KNN aiuta a predire se un film sarà di successo o meno.

6.2 Random Forest

Il Random Forest crea molti alberi decisionali diversi e prende una decisione finale basandosi sulla maggioranza dei loro risultati. Questo metodo riduce problemi come l'overfitting (quando un modello impara troppo dai dati di training e poi non funziona bene su dati nuovi) e migliora la precisione della classificazione.

6.3 Logistic Regression

La regressione logistica è un modello statistico usato per stimare la probabilità che un film sia di successo o meno. Trasforma le caratteristiche del film in una probabilità compresa tra 0 e 1, e quindi decide a quale classe assegnare il film basandosi su questa probabilità.

6.4 Decision Tree

L'albero decisionale è un modello che utilizza una struttura ad albero per fare previsioni. Ogni nodo interno dell'albero rappresenta una decisione basata su una caratteristica del film, e ogni foglia rappresenta la classificazione finale (successo o non successo).

6.5 Accuratezza dei classificatori

Per valutare l'efficacia dei modelli di classificazione, vengono utilizzate due metriche principali:

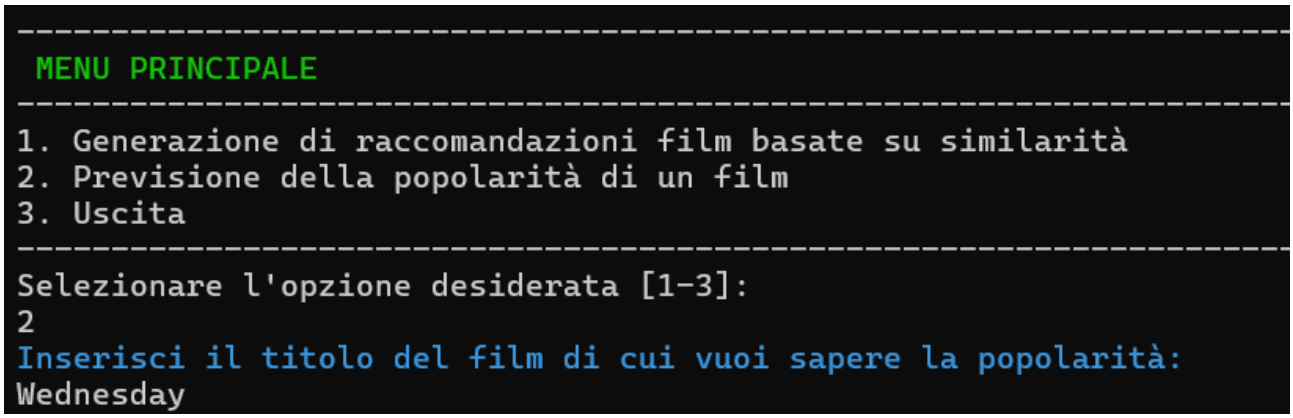
- **Accuratezza (Accuracy):** Rappresenta la proporzione di previsioni corrette (sia veri positivi che veri negativi) sul totale delle osservazioni.

$$\text{Accuracy} = \frac{\text{Numero di previsioni corrette}}{\text{Numero totale di previsioni}}$$

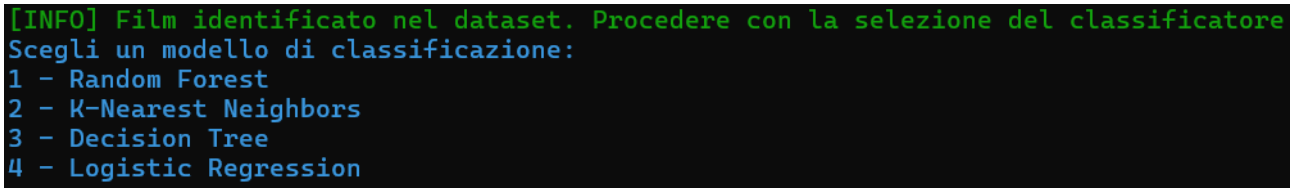
- **Area Sotto la Curva ROC (AUC - Area Under the Receiver Operating Characteristic Curve):** Misura la capacità di un modello di distinguere tra classi positive e negative. Un valore AUC vicino a 1.0 indica un classificatore eccellente, mentre un valore di 0.5 indica un classificatore

che performa come un'assegnazione casuale. È meno sensibile agli sbilanciamenti di classe rispetto all'accuratezza.

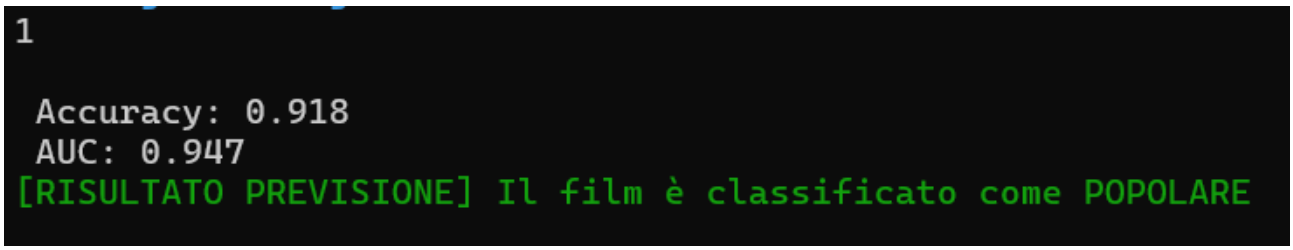
Queste metriche sono calcolate e stampate a console per ciascun modello dopo l'addestramento e la valutazione sul set di validazione.



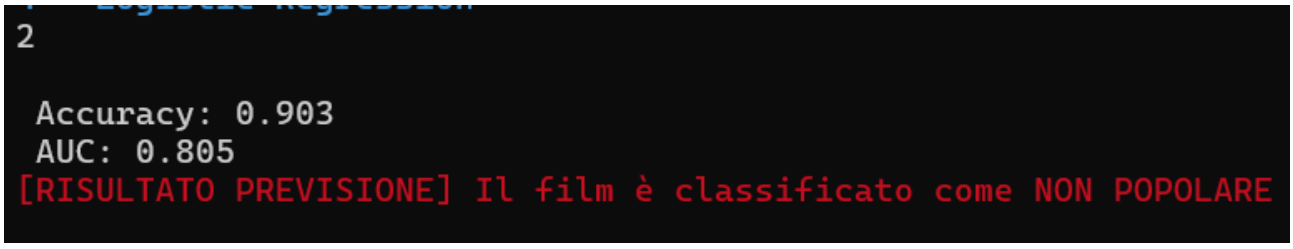
Scelta del titolo di cui voglio sapere la popolarità



Scelta del modello di classificazione



Scelta 1: Random Forest



Scelta 2: K-nearest Neighbors

```
3

 Accuracy: 0.910
 AUC: 0.781
[RISULTATO PREVISIONE] Il film è classificato come POPOLARE
```

Scelta 3: Decision Tree

```
4

 Accuracy: 0.902
 AUC: 0.919
[RISULTATO PREVISIONE] Il film è classificato come NON POPOLARE
```

Scelta 4: Logistic Regression

L'esito di questo confronto mi ha portato a scegliere il Random Forest come miglior classificatore per la predizione della popolarità.

6.6) Guida all'uso

```
SISTEMA DI ANALISI E RACCOMANDAZIONE FILM


------------------------------------------------------------------
 MENU PRINCIPALE
------------------------------------------------------------------
1. Generazione di raccomandazioni film basate su similarità
2. Previsione della popolarità di un film
3. Uscita
------------------------------------------------------------------
Selezionare l'opzione desiderata [1-3]:
2
Inserisci il titolo del film di cui vuoi sapere la popolarità:
Inception
[INFO] Film identificato nel dataset. Procedere con la selezione del classificatore
```

```
Scegli un modello di classificazione:
1 - Random Forest
2 - K-Nearest Neighbors
3 - Decision Tree
4 - Logistic Regression
1

 Accuracy: 0.918
 AUC: 0.946
[RISULTATO PREVISIONE] Il film è classificato come POPOLARE
```

7. Conclusioni

Il sistema sviluppato rappresenta un prototipo funzionale di **Recommender System ibrido** per film, che integra tecniche di **clustering**, **classificazione** e **similarità**. Utilizzando un approccio modulare, il progetto riesce a combinare algoritmi di machine learning supervisionati (come Random Forest, KNN, Logistic Regression e Decision Tree), metodi non supervisionati (come K-Means per il clustering) e metriche di similarità (come la similarità del coseno).

La struttura del codice è organizzata in moduli, rendendolo facilmente **leggibile**, **manutenibile** ed **estensibile**. Ciò apre la possibilità di aggiungere in futuro nuove funzionalità, come:

- Filtri personalizzati per paese, anno...
- Salvataggio o esportazione delle raccomandazioni
- Interfacce grafiche per una migliore interazione con l'utente

Questo progetto fornisce un'ottima base per sviluppare applicazioni reali nel contesto della raccomandazione e predizione della popolarità di contenuti.

8. Riferimenti Bibliografici

[1] Scikit-learn – Machine Learning in Python: https://scikit-learn.org/
[2] Pandas – Data analysis and manipulation tool: https://pandas.pydata.org/
[3] IMDb Datasets – Access to large-scale movie metadata: https://www.imdb.com/interfaces/
[4] Matplotlib – Visualization with Python: https://matplotlib.org/
[5] NumPy – Fundamental package for scientific computing: https://numpy.org/