Analisi e Modellazione Predittiva dei Voti IMDb con Machine Learning

Gruppo di lavoro:

Porrelli Sebastiano, 760461, s.porrelli@studenti.uniba.it

Renna Francesco, 761666, f.renna19@studenti.uniba.it

Link Progetto

AA 2025-2026

Sommario

1.Introduzione

Questo caso di studio mira ad analizzare i dati relativi a film e serie TV presenti su Netflix, unendo informazioni provenienti da due dataset differenti per costruire un modello di regressione in grado di stimare il punteggio IMDb di un contenuto.
L'obiettivo principale è sfruttare un insieme di **feature ibride** (cast, registi, generi, statistiche di collaborazioni, durata, anno) combinate con tecniche di **Machine Learning (Stacked Ensemble)** per ottenere previsioni accurate del punteggio. In caso di mancanza di dati utili per la predizione, verrà visualizzato un messaggio di errore informativo.
Il flusso del progetto include:

1. Acquisizione e integrazione dei dataset, con tecniche di normalizzazione e fuzzy matching per garantire una fusione accurata dei dati.

2. Feature engineering avanzato, con creazione di variabili derivate da performance storiche di attori e registi, collaborazioni e popolarità dei generi.

3. Costruzione di una Knowledge Base (OWL) per supportare ragionamenti semantici e potenziale estensione verso sistemi intelligenti.

4. Costruzione e valutazione di un modello di regressione basato su Stacked Ensemble, validato tramite metriche quantitative (MAE, RMSE) e visualizzazioni (scatter plot, distribuzione errori).

5. Predizione su nuovi contenuti, con gestione di casi incompleti o con informazioni limitate.

1.1 Requisiti funzionali:

Il caso di studio è stato realizzato in Python in quanto è un linguaggio che ci mette a disposizione molte librerie che permettono di trattare i dati in modo facile e intuitivo.

IDE utilizzato: Kaggle.

1.2 Librerie utilizzate:

Gestione dei Dati e Utilità

- **pandas**: Una delle librerie più importanti per l'analisi dei dati in Python. Permette di lavorare con i **DataFrame**, che sono tabelle strutturate, e di eseguire operazioni come caricare dati da file CSV (pd.read_csv), unirli (pd.merge), e manipolarli in modo efficiente.
- **numpy**: La libreria di base per i calcoli numerici e scientifici in Python. È ottimizzata per lavorare con array e matrici e viene spesso usata insieme a Pandas.
- **os**: Fornisce un modo per interagire con il sistema operativo.

- **re**: Permette di usare le **espressioni regolari** per trovare e manipolare modelli di testo nelle stringhe.
- **warnings**: Utile per controllare e ignorare specifici messaggi di avviso che a volte appaiono durante l'esecuzione del codice.
- **collections**: Contiene strutture dati aggiuntive come defaultdict, che può semplificare la scrittura di codice per l'aggregazione di dati.
- **itertools**: Contiene funzioni per creare iteratori efficienti per cicli di dati complessi, come product e combinations.

Machine Learning (Scikit-learn)

Questa è la libreria principale per il machine learning e include molti moduli diversi.

- **sklearn.model_selection**: Contiene strumenti per la suddivisione e la validazione dei dati, come train_test_split per dividere i dati in set di training e test, e RepeatedKFold per la validazione incrociata.
- **sklearn.ensemble**: Moduli per i modelli di machine learning che combinano più stimatori. RandomForestRegressor e StackingRegressor sono due esempi di questi modelli.
- **sklearn.metrics**: Usato per valutare la performance del modello con metriche come mean_squared_error (Errore Quadratico Medio).
- **sklearn.preprocessing**: Contiene funzioni per la pre-elaborazione dei dati, come OneHotEncoder per convertire variabili categoriche in un formato numerico che il modello può capire.
- **sklearn.impute**: Contiene SimpleImputer, che è lo strumento che usiamo per riempire i valori mancanti con un valore (come la mediana) del dataset.
- **sklearn.linear_model**: Include modelli di regressione lineare. Ridge e LinearRegression sono usati come "modelli base" all'interno del StackingRegressor.

Modellazione Avanzata e Statistica

- **rapidfuzz**: Una libreria per l'analisi e il confronto di stringhe molto veloce. È usata per trovare somiglianze tra testi, utile per confrontare i titoli o nomi di attori quando potrebbero esserci piccole differenze.
- **xgboost**: Una libreria estremamente popolare e potente per il **gradient boosting**, un tipo di algoritmo di machine learning che costruisce una serie di alberi decisionali per fare previsioni. XGBRegressor è la sua versione per la regressione.

- **pgmpy**: Questa libreria è specializzata nella creazione e gestione di **modelli grafici probabilistici**, come le reti bayesiane. È uno strumento avanzato per la modellazione statistica e il ragionamento causale.

- **owlready2**: Questa libreria serve a lavorare con le **ontologie**, che sono rappresentazioni formali e strutturate della conoscenza. Viene usato per trasformare i dati tabulari in un grafo di conoscenza, dove i film, gli attori, i registi sono nodi che si collegano tra di loro.
- **networkx**: È una libreria per lo studio e la manipolazione di **grafi e reti**. Viene utilizzata per visualizzare o analizzare il grafo di conoscenza.

---

2. Utilizzo dei dataset

L'integrazione dei dataset è un passaggio fondamentale per poter unire informazioni provenienti da fonti eterogenee. Nel nostro caso, abbiamo due dataset distinti:

- Netflix TV Shows and Movies.csv: contiene informazioni relative ai contenuti (titolo, punteggio IMDb, numero di voti, durata).
- netflix_titles.csv: include metadati aggiuntivi (cast, registi, genere, anno di rilascio, descrizione).

L'obiettivo di questa fase è stato unire coerentemente le informazioni relative a uno stesso contenuto, garantendo che ogni record finale abbia sia i dati statistici (voti, punteggi) sia le informazioni descrittive (cast, registi, genere).

2.1 Normalizzazione dei dati

Prima del merge, è stata applicata una normalizzazione dei titoli, necessaria per ridurre discrepanze dovute a:

- Differenze di maiuscole/minuscole (Inception vs inception).
- Presenza di caratteri speciali (Spider-Man vs Spiderman).
- Spaziature multiple (Game of Thrones vs Game of Thrones).

È stata implementata una funzione di normalizzazione:

```
def normalize_title(t):
    if pd.isna(t):
        return ""
    s = str(t).strip().lower()
    s = re.sub(r'[^0-9a-z\s]', '', s)
    s = re.sub(r'\s+', ' ', s)
    return s
```

Questa funzione viene applicata a entrambi i dataset per ottenere una colonna title_norm pulita, utilizzabile come chiave di unione.

2.2 Merge Esatto

Il primo tentativo di integrazione è stato eseguito tramite merge esatto sulle chiavi:

- title_norm (titolo normalizzato)
- year (anno di rilascio)

```
merged = pd.merge(df_info, df_ratings,
                  left_on=['title_norm','year'],
                  right_on=['title_norm','year'],
                  how='inner',
                  suffixes=('_info','_ratings'))
```

Questo approccio garantisce la massima precisione, ma rischia di perdere contenuti per cui i titoli nei due dataset differiscono leggermente.

2.3 Fuzzy Matching

Per ridurre la perdita di informazioni, è stato introdotto un fuzzy matching per i titoli che non hanno trovato corrispondenza nell'unione esatta.
Il fuzzy matching consente di:

- Identificare corrispondenze anche in presenza di differenze minori (errori di battitura, abbreviazioni).
- Assegnare un punteggio di similarità tra 0 e 100.

Implementazione:

```
match, score, match_idx = process.extractOne(t, choices, scorer=fuzz.QRatio)
if score >= 90:
    matched_rows = df_ratings[(df_ratings['title_norm'] == match) & (df_ratings['year'] == y)]
```

Flusso operativo:

1. Per ogni contenuto nel dataset netflix_titles.csv, si considerano solo i titoli del dataset Netflix TV Shows and Movies.csv con lo stesso anno.
2. Si calcola la similarità con RapidFuzz.
3. Se la similarità è ≥90, i record vengono uniti.

Esempio:

- title_info = "The Matrix"
- title_ratings = "Matrix"
- Similarità calcolata = 92 → Match valido.

2.4 Risultato dell'Integrazione

L'output di questa fase è un dataset unico e arricchito, che per ogni film/serie include:

- Dati anagrafici e descrittivi (titolo, cast, registi, generi, anno).
- Dati quantitativi (punteggio IMDb, numero di voti, durata).

Questo dataset costituisce la base per le successive fasi di feature engineering e modellazione predittiva.

---

3. Feature Engineering

Il processo di feature engineering ha permesso di trasformare i dati grezzi in un insieme di variabili significative per la predizione del punteggio IMDb. Le feature sono state progettate per catturare informazioni sia quantitative (valori numerici derivanti dai dataset) sia qualitative (relazioni semantiche tra entità come attori, registi e generi).

3.1 Tipologie di Feature

Le feature create si suddividono in diverse categorie:

a) Feature legate al team di produzione

- director_mean: media dei punteggi IMDb ottenuti dal regista nei film presenti nel dataset.
- actor_mean_avg: media dei punteggi IMDb dei membri principali del cast.
- team_mean: media dei punteggi dei film in cui regista e attori hanno collaborato in passato.
- num_frequent_actors: numero di attori del cast che hanno una presenza significativa nel dataset (≥3 apparizioni).

b) Feature legate ai generi

- primary_genre: genere principale del film.
- primary_genre_pop: popolarità del genere principale, calcolata come media dei punteggi dei film di quel genere.

- director_combined_mean_avg: punteggio medio del regista pesato in base al genere del film (unisce performance generali e specifiche di genere).
- actor_combined_mean_avg: punteggio medio degli attori pesato in base al genere del film.
- genre_X: variabili binarie (one-hot encoding) che indicano quale genere è presente nel film che si sta analizzando.

c) Feature temporali e descrittive

- year: anno di rilascio del contenuto.
- duration: durata del film convertita in minuti.
- imdb_votes: numero di voti ricevuti dal contenuto.

d) Target

- imdb_score: punteggio IMDb del contenuto, utilizzato come variabile target per la regressione.

3.2 Pulizia e Imputazione dei Dati

Alcune variabili presentavano valori mancanti (es. durata, numero di voti, medie registi/attori assenti).
Per gestirli:

- È stato utilizzato un SimpleImputer con strategia “median” per le feature numeriche. Viene utilizzato nei casi in cui ci sia un valore mancante (NaN) in alcune colonne del dataset, per evitare che i modelli di Machine Learning vengano penalizzati da questi vuoti. Nel nostro caso viene utilizzato per imbd_votes(numero di voti) che nei nuovi film è una variabile vuota. Ad essa viene dato un peso minore nella predizione grazie all’inserimento di un’altra variabile (imdb_votes_missing) che viene messa ad 1 se il valore in precedenza era Nan e 0 altrimenti.
- Le feature categoriali sui generi sono state codificate tramite One-Hot Encoding. È una tecnica per trasformare variabili categoriche (testuali o discrete) in variabili numeriche in modo che possano essere utilizzate da algoritmi di Machine Learning. Nel nostro caso viene utilizzato perché ogni film ha diversi generi. Ogni film ha 1 nei generi che lo caratterizzano, 0 negli altri, in questo modo, i modelli di ML possono interpretare la presenza/assenza di ciascun genere senza introdurre ordini fittizi.
- Le durate sono state convertite in minuti eliminando formati testuali ("1h 30min" → 90).

3.3 Output Finale

Il risultato di questa fase è un dataset numerico pronto per il Machine Learning, con:

- Feature derivate da dati storici (cast/registi)
- Feature basate sul contesto (genere, anno, popolarità)
- Target continuo (punteggio IMDb)

---

4. Creazione e Utilizzo della Knowledge Base (KB)

4.1 Introduzione alla Knowledge Base

Per migliorare la capacità predittiva del modello di regressione, il progetto sfrutta i principi dell'Ingegneria della Conoscenza per costruire e interrogare una Knowledge Base (KB). Questa KB, implementata come un'ontologia, modella le entità principali del nostro dominio (film, attori, registi, generi) e le relazioni che intercorrono tra di esse.

4.2 Definizione dell'Ontologia

L'ontologia, definita nel file movielab.owl, è il cuore della nostra KB. Si basa su una struttura gerarchica di classi e sulla definizione delle relazioni che le legano.

Classi (Tipi di entità):

- Film: Rappresenta una singola opera cinematografica o televisiva.
- Person: Una classe generale che funge da genitore per le entità umane.
    - Actor (Sotto-classe di Person): Rappresenta un attore.
    - Director (Sotto-classe di Person): Rappresenta un regista.
- Genre: Rappresenta un genere cinematografico.

Proprietà (Relazioni e attributi):

Le proprietà definiscono i collegamenti tra le classi o gli attributi delle stesse.

- hasActor (ObjectProperty): Collega un individuo di tipo Film a uno o più individui di tipo Actor.
- hasDirector (ObjectProperty): Collega un individuo di tipo Film a uno o più individui di tipo Director.
- hasGenre (ObjectProperty): Collega un individuo di tipo Film a uno o più individui di tipo Genre.
- hasRating (DataProperty): Collega un individuo di tipo Film a un valore numerico (float) che rappresenta il suo voto IMDb.

- releaseYear (DataProperty): Collega un individuo di tipo Film a un valore numerico (int) che rappresenta l'anno di uscita.

La definizione di queste proprietà garantisce la correttezza strutturale della KB. Ad esempio, la proprietà hasRating è definita come functional, il che significa che ogni Film può avere un solo voto IMDb.

4.3 Materializzazione dei Fatti (Popolamento della KB)

Una volta definita la struttura dell'ontologia, il passo successivo consiste nel popolarla con i dati reali del nostro dataset. Questo processo, noto come "materializzazione dei fatti", trasforma le righe del DataFrame in **individui** (istanze) all'interno della KB.

L'implementazione segue questi passaggi:

1. Vengono identificate le entità frequenti (attori e registi con almeno 3 occorrenze) e per ognuno di essi viene creato un individuo all'interno della rispettiva classe (Actor o Director).
2. Per ogni film nel dataset, viene creato un nuovo individuo della classe Film.
3. Le proprietà vengono popolate in base ai dati del film:
    - Il voto IMDb viene assegnato alla proprietà hasRating.
    - L'anno di uscita viene assegnato alla proprietà releaseYear.
    - Gli individui Actor e Director vengono collegati all'individuo Film tramite le rispettive proprietà hasActor e hasDirector.

4.4 Utilizzo della KB per il Feature Engineering

La KB non viene utilizzata direttamente per la predizione, ma funge da base per derivare feature più sofisticate. Le "feature ibride", come director_combined_mean_avg e actor_combined_mean_avg, sono un esempio diretto di come la conoscenza strutturata della KB venga sfruttata.

Per calcolare director_combined_mean_avg, il codice non si limita a usare la media generale di un regista (calcolata su tutti i suoi film), ma la combina con la sua media specifica per un dato genere. La logica applicata è:

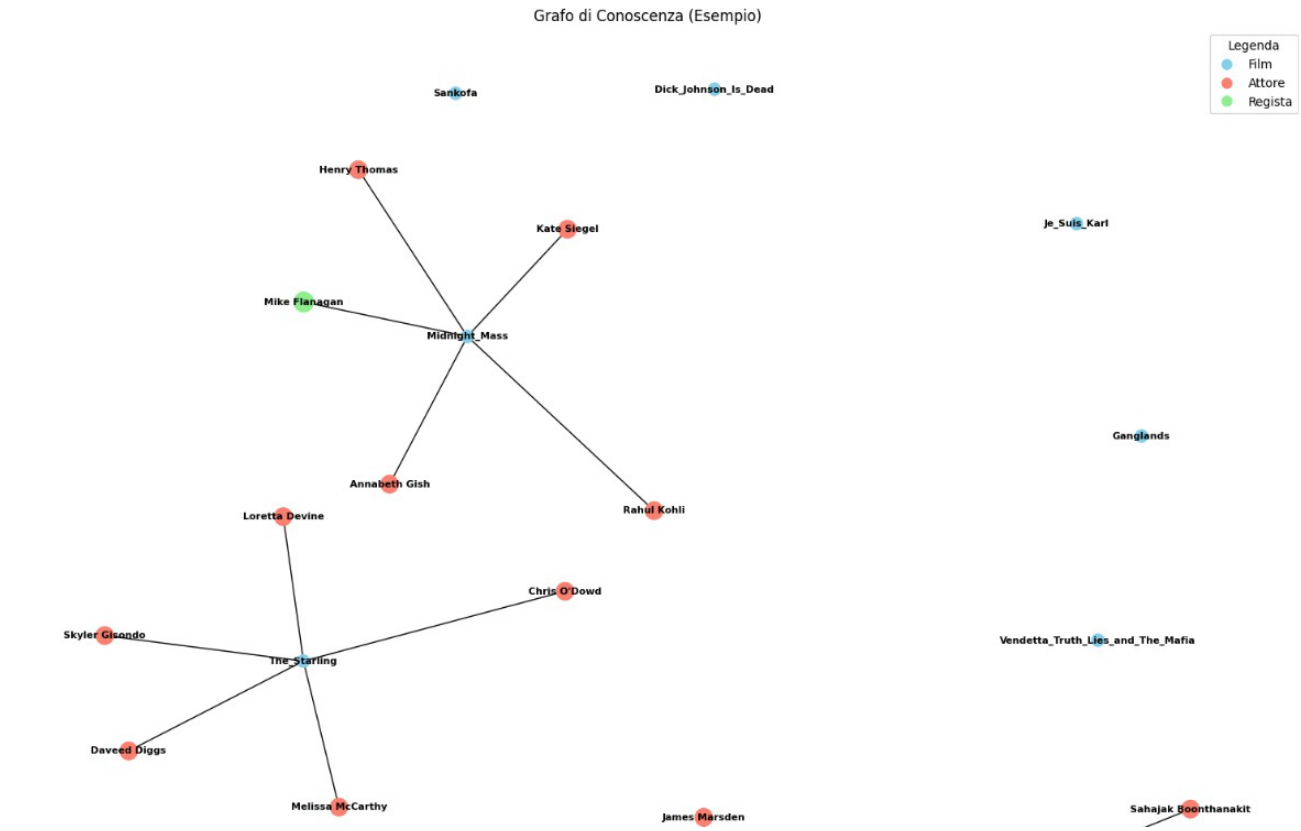
- Se un regista ha una media di voto significativa in un genere specifico, quel valore ha un peso maggiore.
- Se tale dato non è disponibile o non è significativo, si usa unicamente la media generale del regista.

Questo approccio consente al modello di riconoscere la specializzazione di un professionista in un

determinato genere. Il processo è replicato anche per gli attori.

4.5 Visualizzazione della KB

Un modo efficace per comprendere la natura interconnessa della Knowledge Base è visualizzarla come un grafo. La libreria **NetworkX** permette di rappresentare le entità come **nodi** e le relazioni come **archi**. La figura di seguito mostra un esempio di come un film e le sue entità correlate sono modellati all'interno della KB.



Come si può osservare, il grafo fornisce una rappresentazione chiara e diretta della struttura della KB:

- **Nodi**: Ogni cerchio colorato nel grafo rappresenta un individuo della nostra ontologia. I nodi sono etichettati in modo chiaro (film: Midnight_Mass, regista: Mike Flanagan, actor: Henry Thomas) indicando sia la classe (film, person, actor) sia il nome dell'individuo.
- **Archi**: Collegano i nodi che rappresentano le proprietà o relazioni definite.

Questa struttura interconnessa viene sfruttata per estrarre le feature ibride utilizzate nel modello di machine learning, collegando la performance di un film alle collaborazioni specifiche tra i suoi professionisti.

---

5. Il Modello di Regressione: Stacked Ensemble

Per la predizione del voto IMDb è stato adottato un modello di regressione di tipo **Stacked Ensemble**. Si configura come un meta-algoritmo che combina in modo ottimale le predizioni di più modelli di base.

Questo approccio ibrido riduce il rischio di overfitting, migliora la generalizzazione e sfrutta i punti di forza individuali di ciascun algoritmo componente.

5.1 Gli Algoritmi Componenti del Modello

Il nostro Stacked Ensemble è stato progettato utilizzando una combinazione di algoritmi noti per la loro efficacia in

La metodologia si articola in due livelli ben definiti:

1. **Livello 0 (Base Learners)**: Un insieme di modelli diversi, scelti per le loro differenti capacità e modalità di apprendimento, viene addestrato in modo indipendente sul set di dati di training. Ogni modello impara a sua volta, elaborando le feature di input e producendo un proprio set di predizioni. I 3 algoritmi da noi scelti sono:

    - **XGBoost (XGBRegressor)**: Questo è un algoritmo di *Gradient Boosting* altamente performante. Costruisce una serie di alberi decisionali in modo sequenziale, dove ogni nuovo albero viene addestrato per correggere gli errori residui delle predizioni fatte dagli alberi precedenti.

    - **Random Forest (RandomForestRegressor)**: Si tratta di un algoritmo di *bagging* che crea un "bosco" di alberi decisionali. Ogni albero viene addestrato su un sottoinsieme casuale dei dati e delle feature. Le previsioni finali sono il risultato della media dei voti predetti da tutti gli alberi.

    - **Ridge Regression (Ridge)**: A differenza dei due precedenti, questo è un modello di regressione lineare. La sua peculiarità risiede nell'applicazione della **regolarizzazione L2**, che aggiunge una penalità alla funzione di costo proporzionale al quadrato dell'ampiezza dei coefficienti. Questa tecnica impedisce che i coefficienti assumano valori eccessivamente elevati, stabilizzando il modello e migliorando la sua capacità di generalizzazione.

2. **Livello 1 (Meta-Modello)**: Le predizioni generate dai modelli di Livello 0 non sono il risultato finale, ma diventano le nuove feature di input per un secondo e ultimo modello, il "meta-modello". Quest'ultimo viene addestrato per apprendere la combinazione ottimale (la "ponderazione") delle predizioni dei base learners, al fine di produrre la stima definitiva del voto. L’algoritmo scelto da noi è:

- **Regressione Lineare (LinearRegression)**: Abbiamo scelto un modello di regressione lineare come meta-modello. Il suo ruolo è quello di apprendere i pesi ottimali da assegnare a ciascuna delle predizioni dei tre base learners. Dato che le predizioni dei modelli di Livello 0 sono già feature di alta qualità e rappresentano una sintesi complessa dei dati originali, un modello lineare è solitamente sufficiente per combinarle in modo efficace e trasparente.

5.2 Implementazione del Modello

La costruzione del modello di Stacked Ensemble è stata realizzata utilizzando le classi di scikit-learn, seguendo una sintassi chiara e modulare.

1. **Definizione dei Base Learners**: I tre algoritmi di regressione di Livello 0 vengono definiti in una lista di tuple che definisce esplicitamente quali "stimatori" (i modelli di base) devono far parte del primo livello del nostro stack. Grazie al random_state, inoltre, ogni volta che il codice verrà rieseguito con gli stessi dati avremo gli stessi risultati (riproducibilità dell'esperimento).

```
# Definisci i modelli di base (Level 0)
estimators = [
    ('xgb', XGBRegressor(objective='reg:squarederror', random_state=42, n_estimators=300)),
    ('rf', RandomForestRegressor(n_estimators=100, random_state=42)),
    ('ridge', Ridge(random_state=42))
]
```

2. **Definizione del Meta-Modello e del Modello di Stacking**: Viene specificato il modello finale e viene istanziato lo StackingRegressor. L'argomento cv=5 indica che le predizioni dei base learners, che fungeranno da input per il meta-modello, saranno generate tramite una **cross-validation a 5 fold,** quindi 4 di apprendimento e 1 di testing. Questo passaggio cruciale previene il *data leakage*, garantendo che il meta-modello venga addestrato solo su predizioni prodotte su dati che i base learners non hanno "visto" durante il loro addestramento.

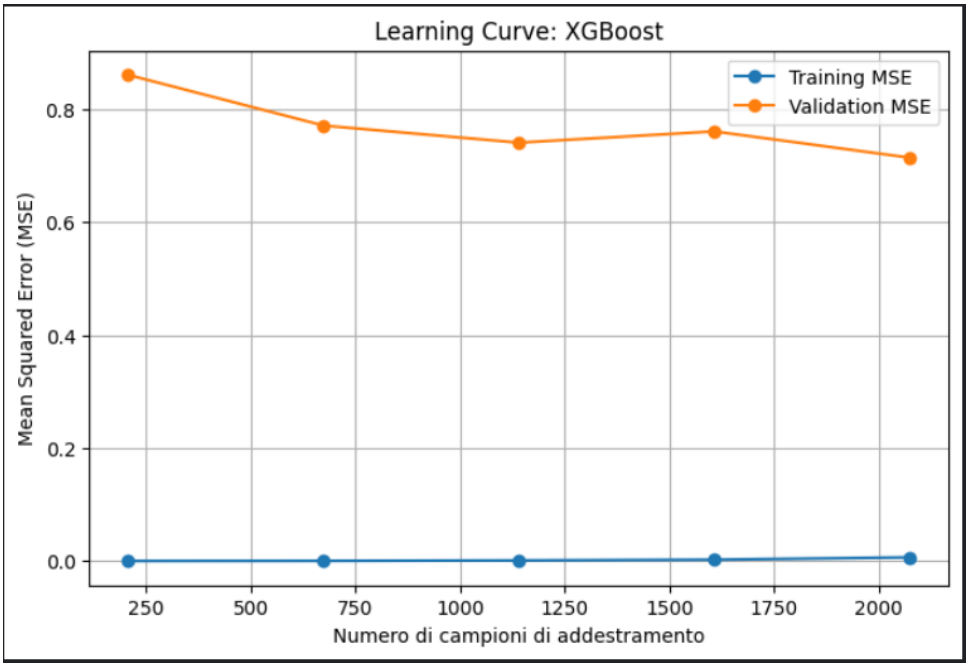
```
# Definisci il meta-modello (Level 1), che combina le predizioni dei modelli di base
final_estimator = LinearRegression()

# Crea il modello di stacking
stacked_regressor = StackingRegressor(
    estimators=estimators,
    final_estimator=final_estimator,
    cv=5,
    n_jobs=-1
)
```

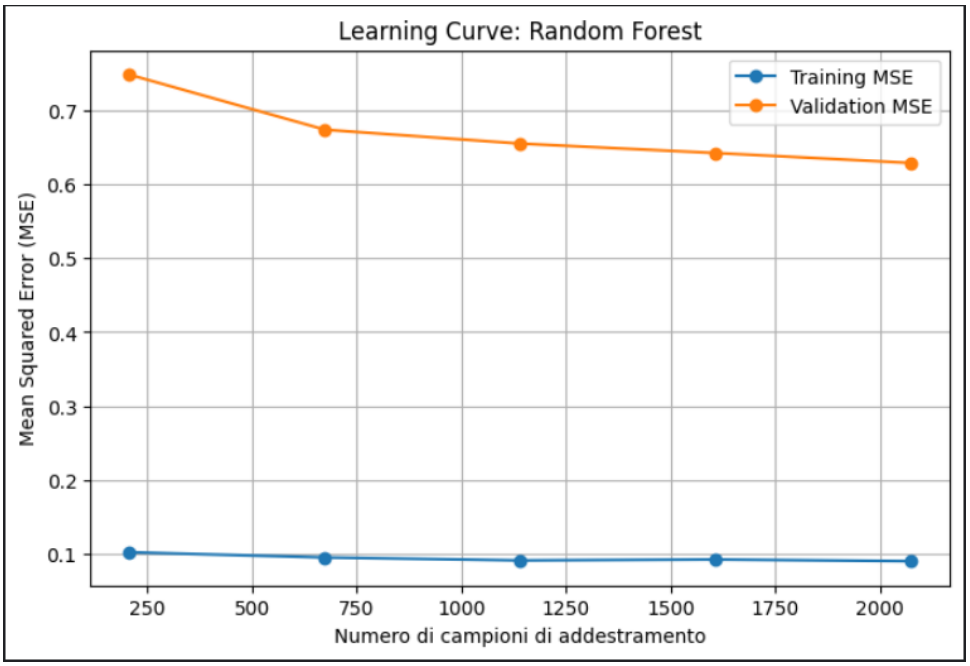
3. **Addestramento e Predizione**: Il modello completo viene addestrato sul set di dati di training e, una volta completato, viene utilizzato per generare le predizioni sul set di test.

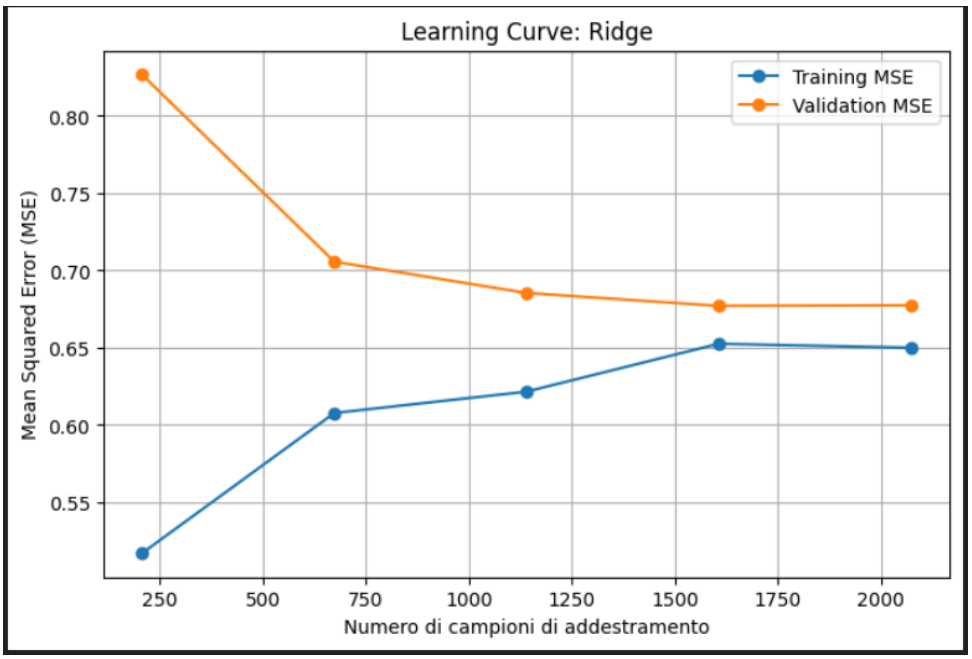
```
# Adatta il modello di stacking ai dati di addestramento
stacked_regressor.fit(X_train, y_train)

# Predizioni sul set di test
y_pred = stacked_regressor.predict(X_test)
```

5.3 Analisi delle Curve di Apprendimento e dei grafici

Per diagnosticare il comportamento del modello durante l'addestramento e identificare potenziali problemi di overfitting o underfitting, sono state analizzate le curve di apprendimento dei 3 algoritmi.

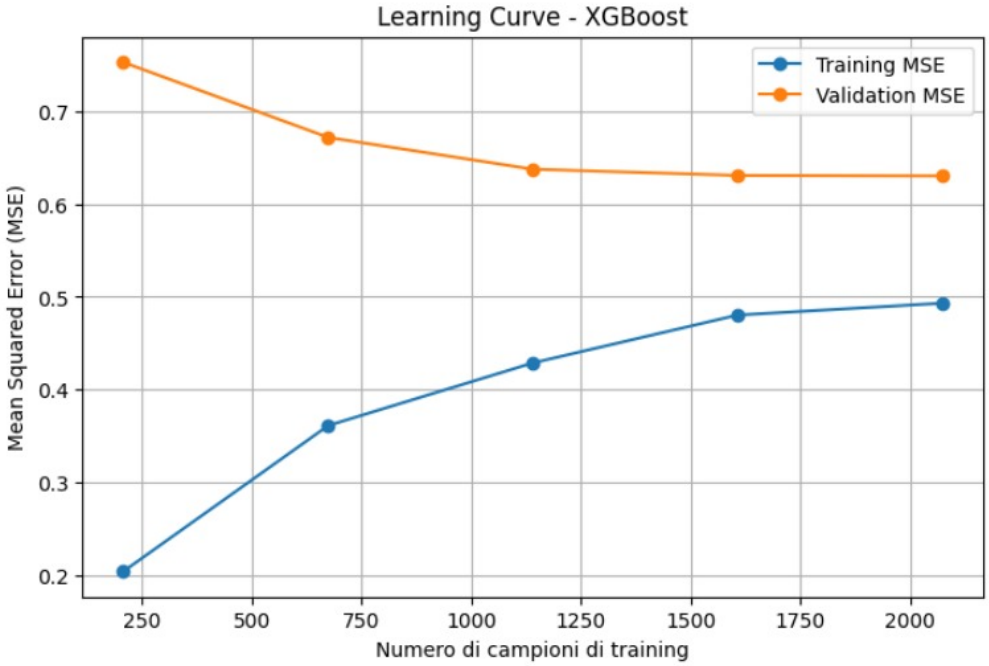


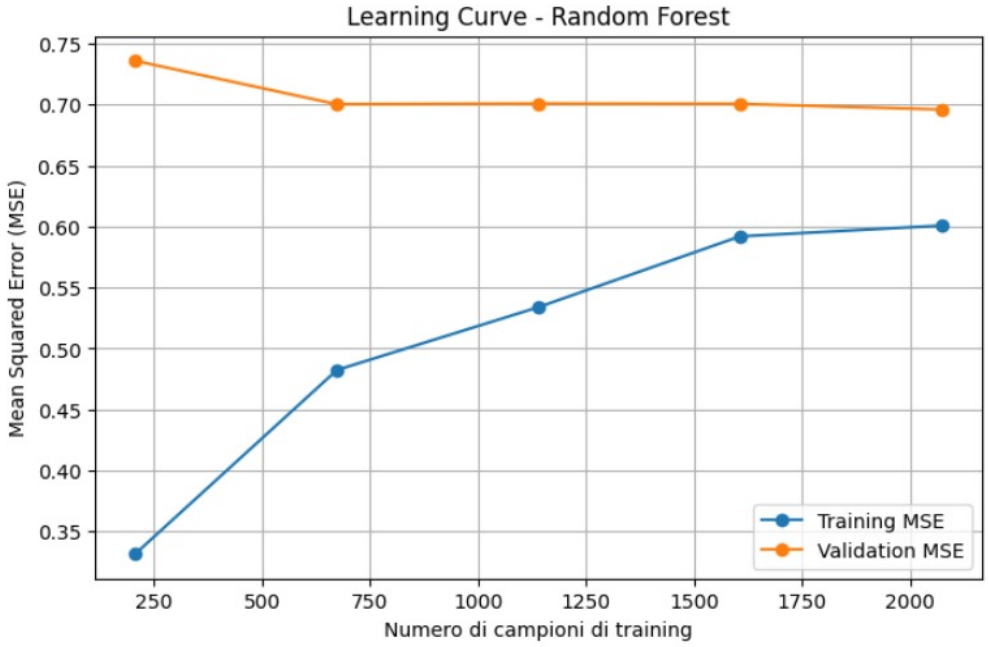


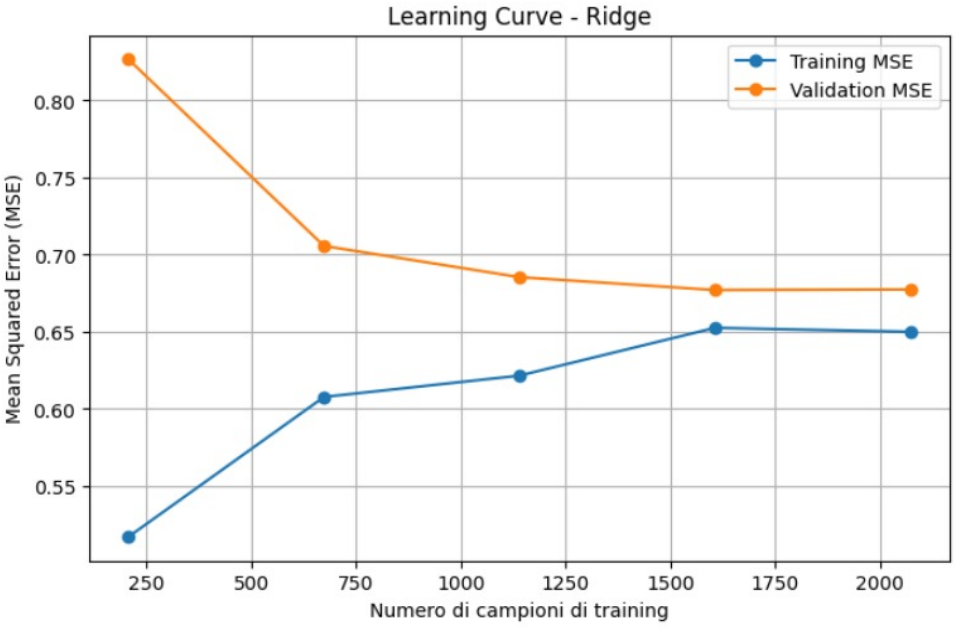


i modelli **XGBoost** e **Random Forest** mostrano un forte **overfitting**, mentre il modello **Ridge** sembra essere il più **adatto a generalizzare.**

Ottimizzando parametri come estimatori, profondità e tasso di apprendimento, abbiamo ottenuto un modello meno complesso e meno propenso all'overfitting

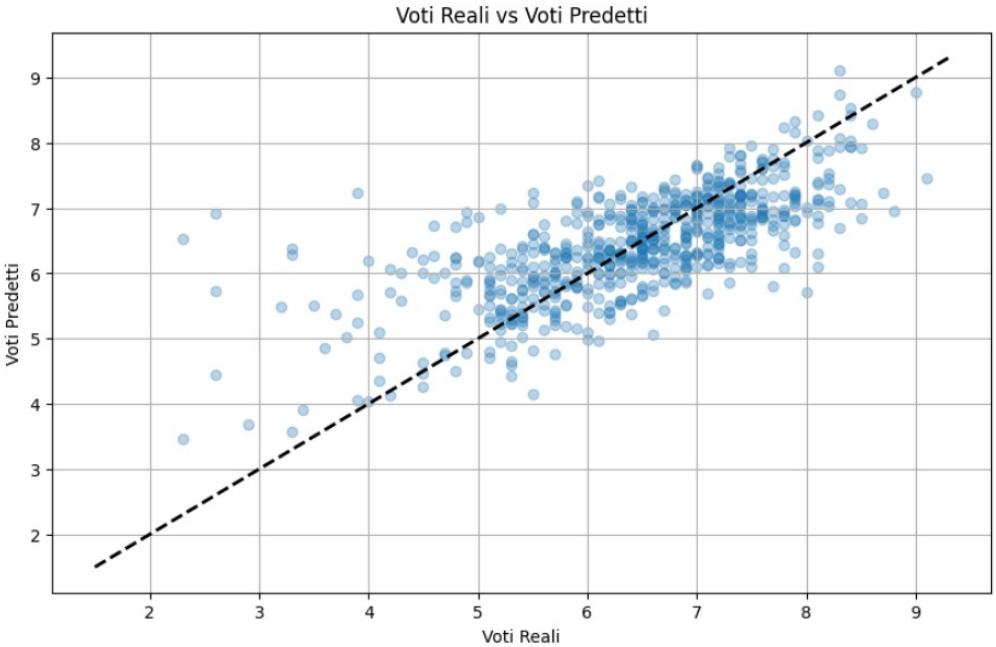

Learning Curve - XGBoost
Training MSE
Validation MSE
Mean Squared Error (MSE)
0.7
0.6
0.5
0.4
0.3
0.2
250
500
750
1000
1250
1500
1750
2000
Numero di campioni di training


Learning Curve - Random Forest
Mean Squared Error (MSE)
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
Training MSE
Validation MSE
250
500
750
1000
1250
1500
1750
2000
Numero di campioni di training

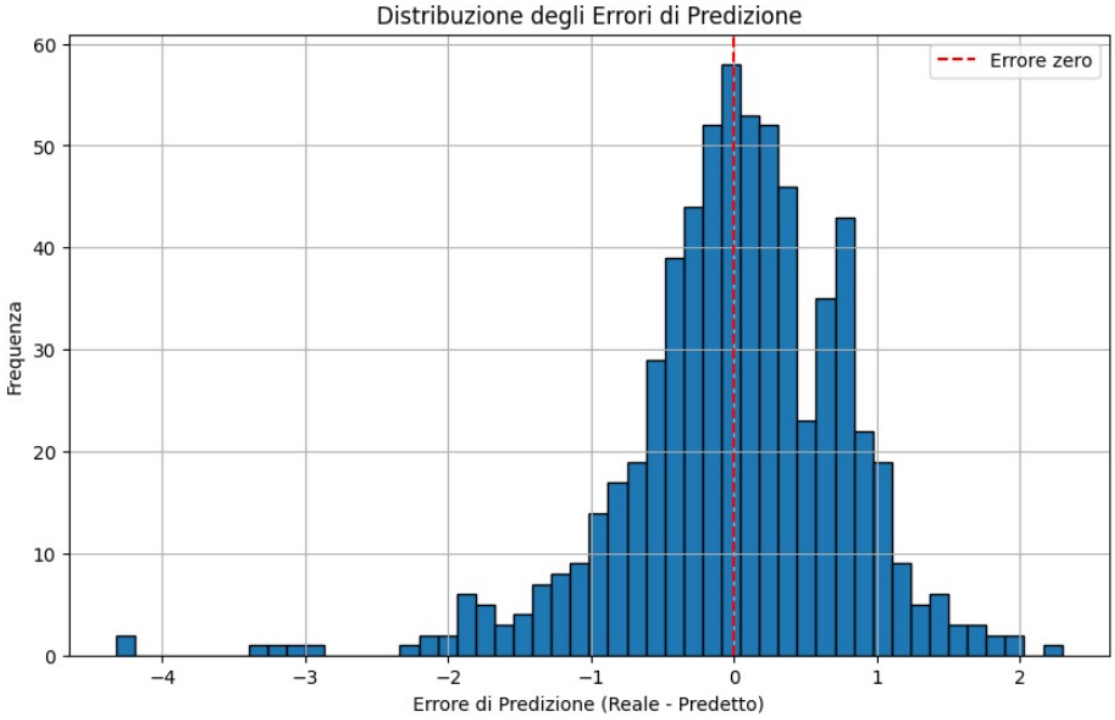


Per una comprensione immediata e intuitiva delle prestazioni del modello, sono stati generati due grafici che visualizzano i risultati della predizione sul set di test.

- Scatter Plot delle Predizioni (Voti Reali vs. Voti Predetti): Questo grafico visualizza ogni punto dato del set di test, posizionando i voti reali sull'asse X e le predizioni del modello sull'asse Y. La presenza di una linea diagonale y = x funge da benchmark per un modello perfetto. Nel nostro grafico i punti si raggruppano in modo denso e lineare attorno alla linea ideale. La dispersione è relativamente contenuta. Questo significa che il modello sta facendo previsioni molto accurate



- Istogramma degli Errori di Predizione: Questo istogramma mostra la distribuzione degli errori del modello (calcolati come voto_reale - voto_predetto). Idealmente, la distribuzione degli errori dovrebbe essere concentrata attorno a zero, proprio come rappresenta il nostro grafico. La maggior parte degli errori si concentrano in un intervallo molto ristretto attorno a -0.5 e 0.5. Ciò indica che la maggior parte delle previsioni del modello sono molto vicine ai valori reali.



5.4 Valutazione Finale del Modello

Per una valutazione esaustiva del modello, sono state calcolate diverse metriche di performance sul set di test, le cui definizioni sono fondamentali per comprendere l'efficacia del modello.

- Mean Absolute Error (MAE): Misura l'errore medio assoluto tra i valori predetti e i valori reali.
- Mean Squared Error (MSE): Misura la media degli errori al quadrato. Penalizza in modo significativo gli errori più grandi, rendendo il modello sensibile agli outlier.
- Root Mean Squared Error (RMSE): La radice quadrata del MSE. Riporta la metrica all'unità di misura originale, rendendola più interpretabile del MSE.

```
--- Valutazione del Modello di Regressione (Stacked Ensemble) ---
Mean Absolute Error (MAE): 0.5743
Mean Squared Error (MSE): 0.6195
Root Mean Squared Error (RMSE): 0.7871
Percentuale predizioni differenza <= 0.5: 54.85%
Percentuale predizioni differenza <= 0.7: 68.41%
```

I valori ottenuti mostrano un **Mean Absolute Error** (MAE) di **0.5743** e il **Mean Squared Error** (MSE) di **0.6195.** Non sono troppo elevati, specialmente considerando il range dei punteggi. Il **Root Mean Squared Error** (RMSE) di **0.7871** conferma che gli errori di predizione non sono eccessivamente grandi in media. La percentuale di predizioni con una differenza dal valore reale inferiore o uguale a 0.5 è del 54.85%. Questo significa che per più della metà del dataset, il modello è molto preciso. La percentuale di predizioni con un errore inferiore o uguale a 0.7 è del 68.41%. Questo valore, sebbene non perfetto, è molto buono e indica che il modello funziona in modo affidabile sulla maggior parte dei dati. Il modello si comporta in maniera discreta, soprattutto se si tiene conto del fatto che il dataset di origine è limitato a film e serie TV di Netflix.

---

6. Predizione di un Nuovo Film

6.1 Introduzione alla Funzione di Predizione

Dopo aver addestrato e validato il modello, l'obiettivo finale del sistema è la sua applicazione pratica: predire il voto di un film completamente nuovo che non è presente nel dataset di addestramento. A tal fine, è stata implementata una funzione dedicata, predict_new_movie, che garantisce che le nuove istanze siano trattate in modo identico a quelle utilizzate per addestrare il modello.

La funzione non si limita a passare i dati grezzi al modello, ma applica in modo intelligente la logica di *feature engineering*, incluso l'utilizzo delle statistiche aggregate derivate dalla Knowledge Base.

6.2 Funzionamento della Pipeline di Predizione

La funzione predict_new_movie prende in input un dizionario che rappresenta le caratteristiche di un nuovo film. Il suo funzionamento è una riproduzione in miniatura della pipeline di addestramento, garantendo coerenza e accuratezza:

1. **Conversione e Pulizia**: Il dizionario di input viene convertito in un DataFrame a riga singola. Le liste di attori, registi e generi vengono estratte e pulite, esattamente come fatto nel processo di pre-elaborazione del dataset completo.
2. **Calcolo delle Feature Derivate**: Per ogni attore e regista menzionato nel dizionario di input, la funzione interroga le mappe di statistiche create durante la fase di preparazione dei dati Riutilizza la "conoscenza" appresa dal dataset per calcolare le feature ibride e le medie generali. Se un attore o regista non è presente nelle mappe (ossia, non è stato trovato nel dataset di addestramento), la sua media viene gestita come valore mancante.
3. **Gestione dei Valori Mancanti (Imputazione)**: Una volta create le feature, la funzione utilizza l'imputatore (SimpleImputer) addestrato in precedenza per riempire eventuali valori mancanti (come le medie per i professionisti sconosciuti o i voti IMDb non disponibili).

4. **Predizione del Voto**: Il DataFrame a riga singola, con le feature pre-elaborate, viene passato al modello stacked_regressor. La funzione restituisce il voto predetto dal meta-modello.

5. **Controllo di Robustezza**: Per evitare predizioni non significative, la funzione include un controllo per verificare se sono disponibili informazioni sufficienti (almeno un attore o un regista noto nel dataset di addestramento). Se non ci sono dati a sufficienza per produrre una previsione affidabile, la funzione ritorna un messaggio di errore informativo, preservando l'integrità del sistema.

6.3 Esempio di Utilizzo

Possiamo vedere in questo esempio che se un film ha un cast e un regista noto, predice un risultato numerico come ci aspettiamo, nel caso in cui il cast e il regista non siano noti non abbiamo dati con cui possiamo portare a termine la predizione, quindi semplicemente facciamo ritornare un messaggio di errore informativo che dice che non ci sono abbastanza informazioni sui membri del cast o del team per effettuare la predizione.

```
--- Esempio di predizione di un nuovo film ---
Predizione per 'The Next Big Thing': 7.41
Predizione per 'The Unknown One': Non ci sono abbastanza informazioni sui membri del cast o del team per effettuare una predizione.
```

7.Conclusioni

Il presente lavoro ha dimostrato con successo come l'integrazione di tecniche di Ingegneria della Conoscenza con un'architettura di Machine Learning possa portare a un sistema predittivo robusto ed efficace. L'obiettivo di predire il voto IMDb dei film di Netflix è stato raggiunto attraverso una pipeline metodologica completa e ben strutturata. Per ottenere risultati migliori sarebbe stato necessario utilizzare un dataset più corposo (più film e più serie TV).

Sviluppi futuri:

- Lo sviluppo di un’interfaccia grafica permetterebbe agli utenti di interfacciarsi direttamente con il modello, si potrebbe legare alla sezione “Novità del momento->Prossimamente” di Netflix, dove per ogni nuova uscita di nuova serie TV o film dà il voto predetto.

- Si potrebbe affrontare il problema di dare un messaggio di errore quando il cast è nuovo, inserendo delle nuove feature, legate alla descrizione (trama), quindi predicendo il voto se la trama è ben articolata, oppure se quel tipo di trama piace di solito al pubblico. Utilizzeremmo delle tecniche di Natural Language Processing (LNP). Aggiungendo queste tecniche non aggiungeremmo solo la predizione del voto per film con cast non noto, ma soprattutto miglioreremmo la predizione del voto per film con cast noto.

8. Riferimenti Bibliografici

Searching for Solutions: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.3]

Supervised Machine Learning: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.7]

Neural Networks and Deep Learning: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.8]

Reasoning with Uncertainty: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.9]

Reinforcement Learning: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.13]

Individuals and Relations: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.15]

Knowledge Graphs and Ontologies: D. Poole, A. Mackworth: Artificial Intelligence: Foundations of Computational Agents. 3/e. Cambridge University Press [Ch.16]

Link dataset:
https://www.kaggle.com/datasets/thedevastator/netflix-imdb-scores?resource=download

https://www.kaggle.com/datasets/shivamb/netflix-shows