Documentazione Caso di Studio
Ingegneria della Conoscenza

A.A 2023/2024

Movies Categorization

Gruppo di lavoro

- Giovanni Pio Amato,738348, g.amato17@studenti.uniba.it
- Fabio Povia,746456, f.povia3@studenti.uniba.it

https://github.com/gpamaa/Moviescategorization

Sommario

Introduzione

Il caso di studio riguarda i film internazionali usciti al cinema dal 1980 al 2020. È stato utilizzato un dataset sui film, presente su kaggle, e le informazioni dei film provengono da vari siti incluso IMDb. Poiché il dataset conteneva delle anomalie di tipo rappresentativo, è stato necessario processare il dataset affinché quest'ultimo fosse più congruo ai nostri obiettivi:

- Realizzazione della knowledge base, al fine di ricavare nuova conoscenza utile per gli step successivi
- Previsione del guadagno dei film, in funzione delle feature, stabilendo quelle che influiscono maggiormente (apprendimento supervisionato)
- Suddivisione del dataset in funzione di caratteristiche comuni (apprendimento non supervisionato)
- Creazione di una rete bayesiana, in grado di calcolare la probabilità che un film appartenga a una delle seguenti classi: big flop, flop, success,masterpiece.

Elenco argomenti di interesse

● Rappresentazione e ragionamento relazionale: vengono inferite nuove informazioni, partendo dai dati contenuti all'interno dei dataset, mediante l'utilizzo di una base di conoscenza in Prolog.

● Apprendimento supervisionato: tramite modelli di regressione quali: modello di regressione

lineare, Random Forest e gradient boosting e successiva classificazione con random forest, decision tree e ada boost

- Apprendimento non supervisionato: utilizzo del k-means per il clustering.
- Modelli di conoscenza incerta: creazione di una Belief Network per il ragionamento probabilistico

Preprocessing e descrizione del dataset

Il dataset originario presentava diversi campi quali:
name: nome del film
rating: sito da cui sono stati presi i voti e lo score
genre: il genere del film
Year: l'anno in cui il film è stato rilasciato
released: la data in cui il film è stato rilasciato
score: media dei voti rilasciati dagli utenti(Dominio 0-10)
votes: numero di voti degli utenti
director: nome del regista
writer: nome dello scrittore
star: attore protagonista
country: nazione dove il film è stato rilasciato
budget: soldi spesi per la realizzazione del film(in dollari)
gross: ricavo del film al botteghino(in dollari)
Company: casa produttrice
runtime: durata del film in minuti

Abbiamo eliminato il campo rating perché lo abbiamo ritenuto inutile ai fini del raggiungimento dei nostri obiettivi. Abbiamo, invece, ritenuto opportuno mantenere solo il mese del campo released che conteneva per lo più informazioni ridondanti come l'anno in cui il film è stato rilasciato e la nazione. Inoltre abbiamo eliminato le righe che contenevano campi con valori nulli al fine di avere un dataset senza tuple anomale. Per questioni di usabilità abbiamo rimosso li spazi, i caratteri speciali e i punti all'interno dei campi.
Abbiamo aggiunto per una migliore accessibilità anche un campo id con lo scopo di fornire per ogni tupla una chiave primaria più semplice per potervi accedere.

Knowledge Base(KB)

La rappresentazione formale della conoscenza è importante per consentire l'espressione della conoscenza in modo preciso, organizzato ed interpretabile da parte di sistemi informatici. Per gestire formalmente la conoscenza, facilitando la ricerca e l'accesso alle informazioni specifiche, si costruisce una knowledge base (base di conoscenza), cioè una raccolta strutturata di informazioni o dati che rappresenta la conoscenza su un determinato dominio o argomento.
La Knowledge Base implementata è utile per effettuare ragionamento automatico sfruttando la struttura di individui e relazioni, potendo inferire nuova conoscenza, e per ingegnerizzare nuove feature, sfruttabili per i task successivi. Per questa implementazione è stato utilizzato il linguaggio Prolog, mediante la libreria pyswip. Per la creazione della KB sono stati definiti i fatti e le regole nel modo seguente:
I fatti vengono rappresentati così
movie(0,'TheShining','Drama', 1980, '8.4', 927000,'StanleyKubrick', 'StephenKing', 'JackNicholson','UnitedKingdom',19000000,46998772.0,'WarnerBros',146,'June').

Regole

mentre le regole sono le seguenti:
get_movie_earn_perc(Id,Earn)

get_movie_earn(Id,Earn)
get_movie_category(Id, Category)
apply_get_movie_category(get_movie_category,[Element|Rest], [Result|RestResult])
get_movies_category(MoviesListS)
apply_get_movie_earn(get_movie_earn,[Element|Rest], [Result|RestResult])
get_movies_from_director(DirectorName,Movies)
get_earn_from_director(Director,Earn)
get_avg_earn_director(Director,Avgint)
convert_list_to_string([Atom|Rest],[String|RestStrings])
get_list_of_director(DirectorListS)
get_unique_directors(UniqueDirList)
get_id_from_director(Director,Id)

Ci sono regole che aggiungono nuova conoscenza:

get_movie_earn, calcola il guadagno di un film
get_movie_earn(Id,Earn) :- movie(Id,_,_,_,_,_,_,_,_,_,Budget,Gross,_,_,_),Earn is Gross-Budget

get movie category assegna la categoria di un film in base al guadagno avuto dallo stesso.

get_movie_category(Id, Category) :- get_movie_earn_perc(Id, Earn), (Earn > 50, Category = 'masterpiece'; Earn >= 25, Earn =< 50, Category = 'success'; Earn < 25, Earn >= 0, Category = 'flop'; Earn < 0, Category = 'big_flop')

Altre regole sono ricorsive:

apply_get_movie_earn serve a calcolare il guadagno per ogni film

apply_get_movie_category(get_movie_category,[Element|Rest], [Result|RestResult]):-
call(get_movie_category,Element,Result),
apply_get_movie_category(get_movie_category,Rest,RestResult)

apply get_movie_category, utile a calcolare la categoria per ogni film

apply_get_movie_earn(get_movie_earn,[Element|Rest], [Result|RestResult]):-
call(get_movie_earn,Element,Result), apply_get_movie_earn(get_movie_earn,Rest,RestResult)

Poichè la funzione ricorsiva apply_get_movie category produceva una lista di Atom invece che produrre una lista di stringhe abbiamo deciso di creare una funzione che convertisse gli atom della lista in stringhe.

convert_list_to_string([Atom|Rest],[String|RestStrings]):-
atom_string(Atom,String),convert_list_to_string(Rest,RestStrings)

La nuova conoscenza ricavata dalle regole, è stata aggiunta in un nuovo dataset.

Apprendimento supervisionato

L'apprendimento supervisionato consiste nell'addestrare un modello ad imparare dai dati ed a migliorare le proprie prestazioni nei compiti specifici senza essere esplicitamente programmato.

Regressione

Considerando le feature del dataset abbiamo ritenuto opportuno scegliere come feature target il guadagno di un film. Una volta che il sistema è allenato aiuterà i produttori di un ipotetico nuovo film a stimare il guadagno che quel film può fruttare in funzione dei valori attribuiti alle altre features.
L'obiettivo è individuare come le feature influiscono nel determinare il valore del guadagno, non considerando il gross e category, perché il valore del guadagno sarebbe poi facilmente ricavabile attraverso la sottrazione tra budget e gross, mentre category è una caratteristica direttamente collegata perché derivata proprio dal guadagno.
Per poter raggiungere questo obiettivo sono state analizzate le prestazioni di vari modelli di regressione in termini di:
Mean absolute error(MAE) che misura la media degli errori assoluti tra le previsioni del modello ed i valori osservati

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean squared error(MSE) che calcola la media dei quadrati degli errori tra le previsioni del modello ed i valori osservati

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

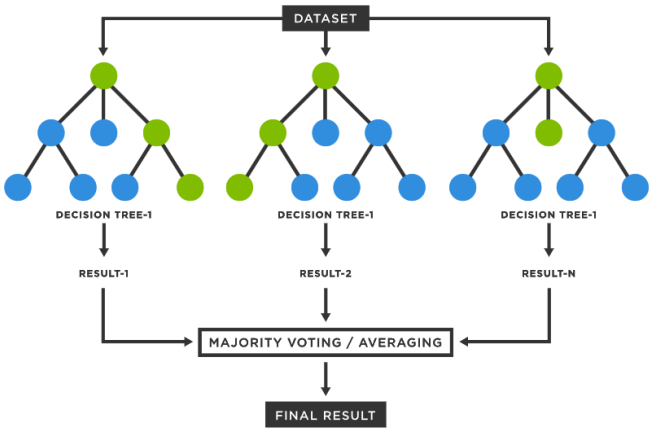
R^2 -score che indica quanto il modello spiega le variazioni nei dati

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

La MAE considera tutti gli errori assoluti uniformemente senza dare più importanza ad un errore assoluto più grande.
La MSE penalizza invece gli errori più grandi perché maggiore è l'errore maggiore sarà il quadrato di quest'ultimo.
L' R^2 score misura quanto i valori previsti si discostano dalla media degli stessi in funzione delle variabili indipendenti del modello.

Random Forest

Per questo caso specifico, si è usata la Random forest, modello di apprendimento automatico che combina molteplici alberi decisionali (decision trees), entrando così nella categoria di modelli ensamble, per migliorare la previsione e la generalizzazione.
La random forest fa parte della gamma dei modelli di apprendimento supervisionato, cioè viene addestrato su un insieme di dati che includono sia le caratteristiche (input features) che le risposte corrette (output features). Il modello impara a fare previsioni basate su questi esempi etichettati.



In particolare la Random Forest funziona creando un insieme di alberi decisionali (regressione/classicazione in base al task), ognuno addestrato su un subset casuale dei dati e su un subset casuale delle caratteristiche. Questo processo introduce variabilità e riduce il rischio di sovradattamento (overtting) cioè quando il modello si lega troppo ai dati di training risultando incapace su dati mai visti. Alla fine ogni albero fornisce una sua previsione, la Random Forest le combina in vari modi (media, moda, etc. . .) per ottenere un risultato più accurato e stabile. Nel nostro caso si è optato per una media delle predizioni di ogni albero.
Dopo varie sperimentazioni si riporta la migliore configurazione del modello, in particolare attraverso la libreria scikit-learn si è definita una random forest le cui caratteristiche peculiari sono:
1. n_estimators = 35, rappresenta il numero di alberi su cui lavorare;
2. criterion = "mean_squared_error", rappresenta il criterio di split degli alberi, in particolare il criterio è l'errore quadratico medio, una misura che quantifica la differenza tra i valori previsti da un modello e i valori osservati o reali puntando a minimizzare la varianza, si calcola:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

dove:

n è il numero di osservazioni;
yi rappresenta il valore osservato (reale);
yˆi rappresenta il valore predetto.
Il criterio squared_error è stato scelto per la sua efficacia e chiarezza di espressione, infatti è espresso nella stessa unità dei dati reali, in questo caso il guadagno, rendendo più comprensibile di quanto le previsioni del modello si discostano, in media, dai valori reali.

Gradient Boosting

Si tratta di un metodo di apprendimento ensemble, il che significa che combina le previsioni di diversi stimatori base (tipicamente alberi decisionali) per migliorare l'accuratezza del modello.

Ecco come funziona:
Inizializzazione: Viene addestrato un modello semplice, spesso un albero decisionale con un solo nodo (detto anche "tronco"), ai dati come punto di partenza. Questo modello iniziale è di solito la media della variabile target.
Addestramento sequenziale di weak learners: I modelli successivi vengono addestrati in sequenza, ognuno concentrato sugli errori commessi dai modelli precedenti. Qui entra in gioco il "gradient" in Gradient Boosting. L'algoritmo minimizza una funzione di perdita (l'errore quadratico medio per la regressione) aggiungendo weak learners (alberi decisionali) che riducono gli errori residui dei modelli precedenti.
Combinazione pesata di weak learners: Le previsioni di tutti i weak learners vengono combinate per fare la previsione finale. A ciascun weak learner viene assegnato un peso che indica il suo contributo alla previsione finale. Questi pesi vengono determinati durante il processo di addestramento, con pesi più elevati assegnati ai modelli che performano meglio nel ridurre gli errori residui.
Il criterio di errore utilizzato di default dal Gradient Boosting regressor di scikit-learn è l'errore quadratico medio (Mean Squared Error, MSE). Questo criterio valuta la discrepanza tra i valori predetti dal modello e i valori effettivi nel dataset di addestramento. L'algoritmo cerca di minimizzare questo errore durante il processo di addestramento per migliorare le prestazioni del modello.

Risultati

Di seguito sono illustrati i risultati dei due modelli

Tabella relativa al random forest

Metrica	Valore
MAE	48012744
MSE	95984168
R^2	0.703

Tabella relativa al gradient boosting

Metrica	Valore
MAE	48200334
MSE	95016898
R^2	0.709

Ci siamo accorti che il MAE e MSE sono decisamente troppo elevati, considerando anche che la media del guadagno è pari a 67220482 per cui la maggior parte delle previsioni sono in larga parte errate.
Pertanto abbiamo deciso di optare per la classificazione, per cui andremo a predire la categoria di successo del film(bigflop,flop,success,masterpiece).

Classificazione

Le metriche utilizzate per valutare i modelli di classificazione sono:
precision: è un valore che indica quanto il modello è accurato nel predire le istanze positive rispetto a tutte le istanze predette.

$$\text{Precisione} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

recall: è una misura di quanto il classificatore sia in grado di identificare correttamente le istanze positive rispetto a tutte le istanze reali positive nel dataset.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

accuratezza: Rappresenta le predizioni corrette rispetto al totale delle predizioni fatte dal modello.

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{True Positives} + \text{True Negatives} + \text{False Positives} + \text{False Negatives}}$$

Il F1-score è una metrica che combina precisione e recall in un'unica misura, fornendo una valutazione complessiva delle prestazioni del modello di classificazione. È particolarmente utile quando hai un insieme di dati sbilanciato, in cui una classe è molto più comune dell'altra.

Il F1-score è calcolato come la media armonica di precisione e recall e viene solitamente calcolato utilizzando la seguente formula:

$$F1\text{-}score = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

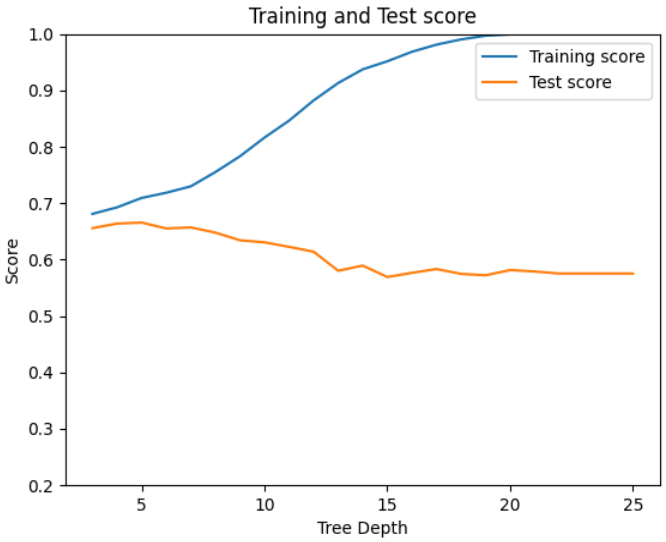
Utilizzo di diversi modelli:
Random Forest, Ada Boost e decision tree.

Decision Tree
Il Decision Tree o albero di decisione è un algoritmo di machine learning che si basa sulla creazione di un albero di decisione. Per il task corrente il dataset considerato si divide in test set e train set, con i quali si sono misurate le performance del modello, al variare della profondità dell'albero. Inoltre, sono state considerate le misure di impurità: log-loss, gini, entropy; poiché non sono emerse

differenze significative tra le prestazioni, è stato utilizzato il criterio entropy.
La formula per calcolare l'entropia di un insieme di dati è:

$$H(X) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

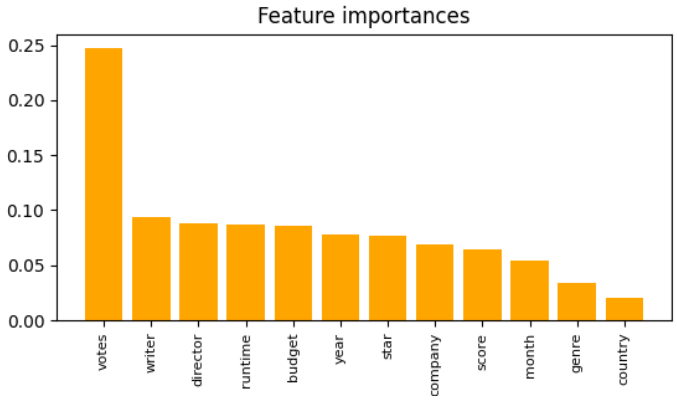
Dove:
H(X) è l'entropia dell'insieme di dati X
c è il numero di classi nel dataset
p_i è la proporzione delle istanze della classe i nell'insieme di dati.
Il grafico sottostante indica lo score, quindi i valori di precision, recall, accurancy ed f1, all'aumentare del numero di alberi



Dal grafico si può notare come all'aumentare della profondità le performance del set di test diminuiscano paradossalmente, mentre per il set di training si adattino troppo ai dati.

Metrica	Valore
Accuratezza	0.6066914299457525
Precisione	0.35003899048306275
richiamo	0.3561357084328441
F1-score	0.37215991580987123

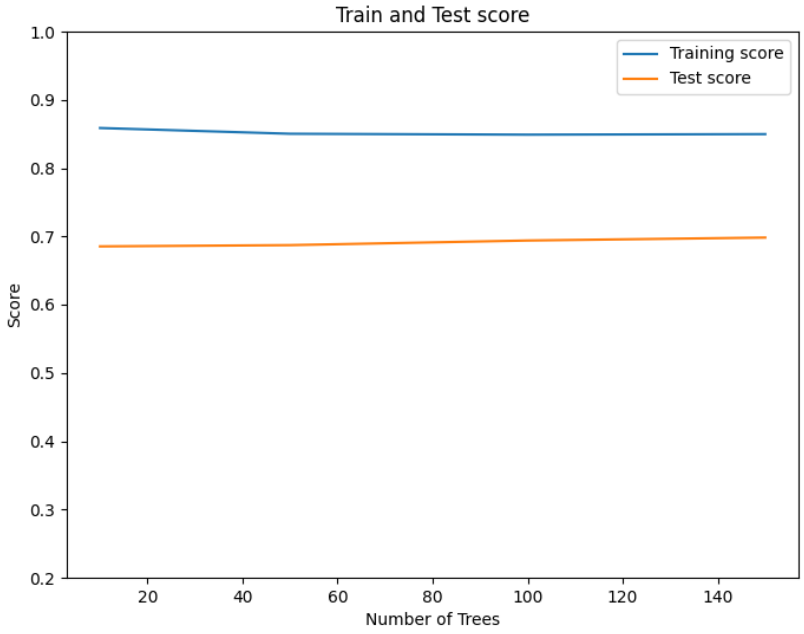
Di seguito è riportato il grafico delle feature più importanti derivate dall'addestramento del modello.



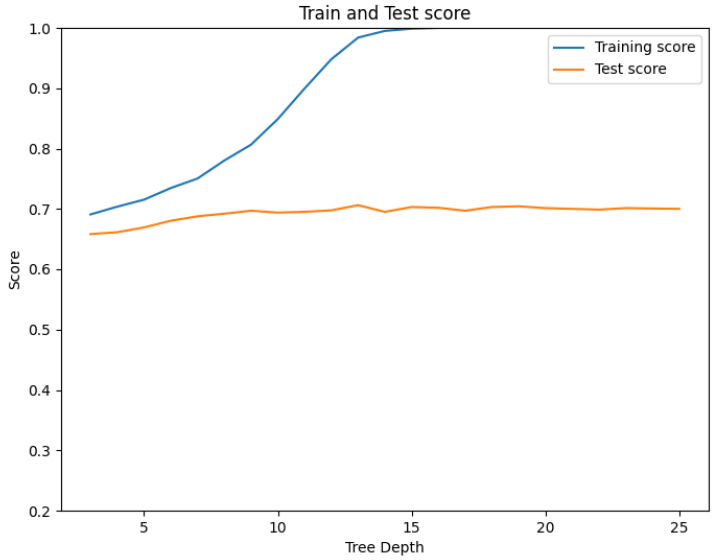
Conclusioni: dal grafico 'Train e Test score' si può ipotizzare che il modello stia diventando troppo complesso e stia iniziando a soffrire di overfitting

Random forest

Come già precedentemente descritto, è un modello che divide il dataset,attraverso la tecnica del bagging, e genera diversi alberi decisionali tutti con un sottoinsieme di feature differente.
Per il random forest di classificazione utilizziamo il criterio di entropia, utilizzata per valutare la purezza di un nodo, ovvero quanto bene i dati in quel nodo sono separati in classi.

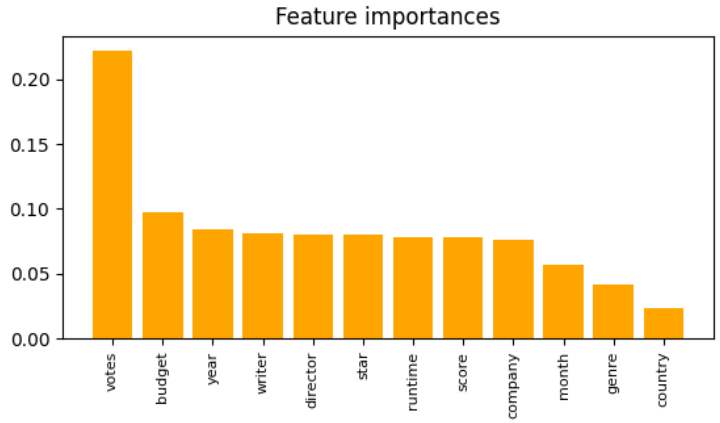


Come si può notare lo score rimane più o meno costante sia per il test set che per il training set.
Ora invece vediamo come cambia il grafico se nelle ascisse poniamo la profondità



Metrica	Valore
Accuratezza	0.6933005531653349
Precisione	0.3386651031189568
richiamo	0.37678282732393237
F1-score	0.35627166010982275

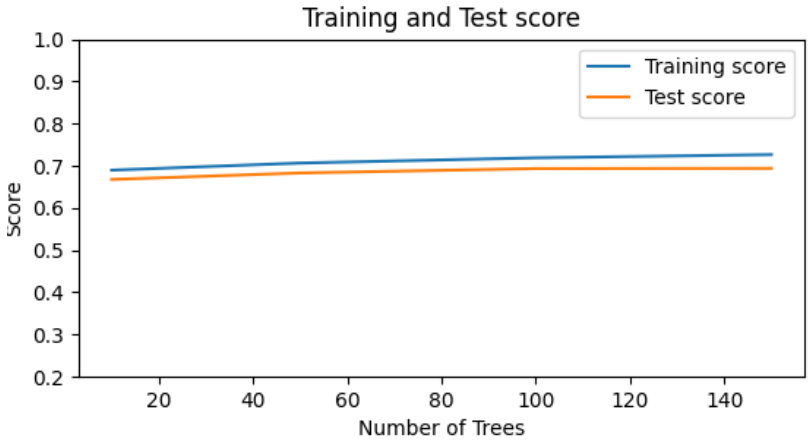
Di seguito è riportato il grafico delle feature più importanti derivate dall'addestramento del modello.



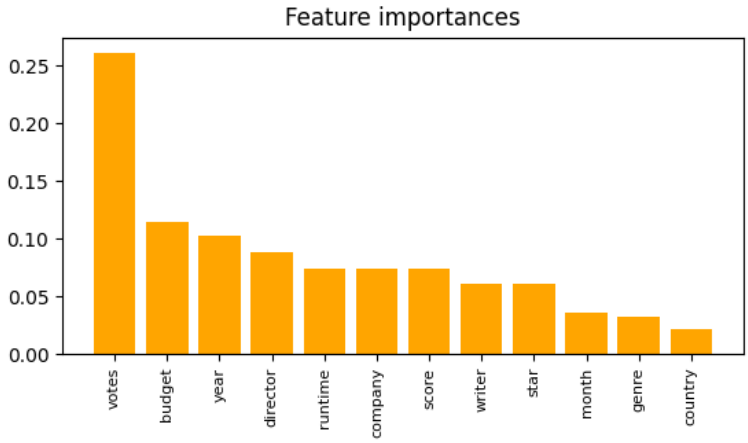
Conclusioni: osservando i grafici, soprattutto nel secondo, si nota come nel training set il modello soffra di overfitting. Per quanto riguarda il training set invece il modello non migliora di tanto le sue performance al variare della profondità e del numero di alberi, come evidenziato anche dalle metriche.

Ada Boost

L'idea alla base di AdaBoost è quella di combinare diversi classificatori deboli in un classificatore forte. Il dataset considerato è stato suddiviso in test set e train set, con i quali si sono misurate le performance del modello, all'aumentare del numero di alberi. La massima profondità è 3, una scelta comune, per evitare un overfitting del modello



Metrica	Valore
Accuratezza	0.6843884449907806
Precisione	0.7096008061209896
richiamo	0.36923496013629753
F1-score	0.35011121368277437

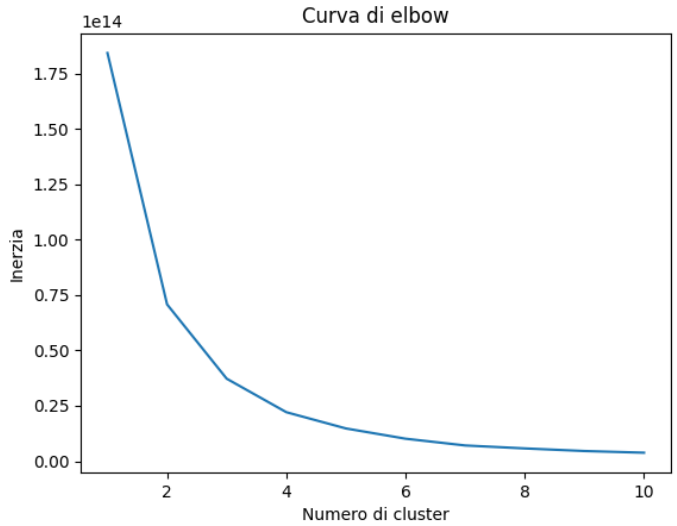


Conclusioni: le metriche di valutazione in questo modello sono leggermente migliori rispetto ai modelli precedenti. Inoltre, nel grafico 'Train e Test score' sono molto vicine, e ciò indica che il modello è in grado di adattarsi adeguatamente ai dati di addestramento e al contempo generalizzare bene su nuovi dati. Questo è un segno positivo, indicando che l'AdaBoost sta ottenendo buone prestazioni sui dati di test senza soffrire di overfitting.

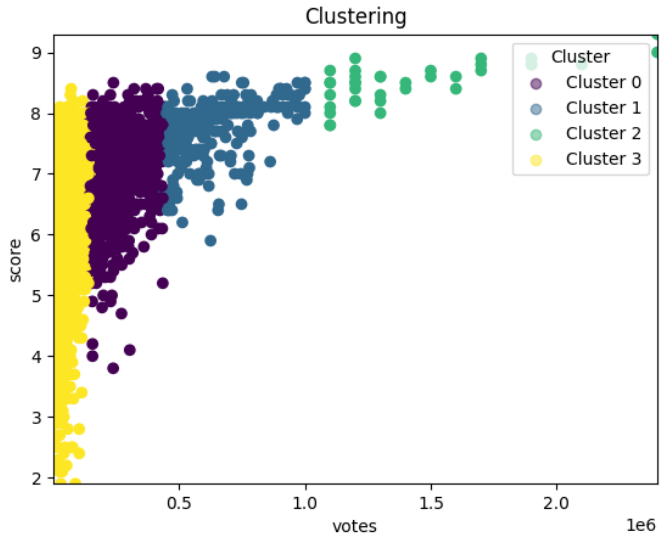
Conclusioni generali Osservando le conclusioni dei modelli si evince che tutti i modelli, tranne l'AdaBoost, soffrono di overfitting. In aggiunta, nel classificatore AdaBoost le metriche di valutazione hanno valori più elevati rispetto agli altri e quindi questo risulta il modello migliore. Quindi, si deduce che le feature più significative che influiscono nel determinare sulla categoria di successo del film sono: voti, budget, director, anno e writer.

Apprendimento non supervisionato

Un altro obiettivo del caso di studio consiste nella creazione di un grafico che evidenzia i film con caratteristiche simili, permettendo al sistema di consigliare nuovi film agli utenti considerando le loro preferenze.
Per fare ciò è stato applicato il clustering, una tecnica di analisi dei dati utilizzata per raggruppare un insieme di oggetti in sottoinsiemi (cluster) in base alle loro somiglianze. Ci sono diversi algoritmi di clustering, tra questi è stato utilizzato il K-Means. Questo algoritmo cerca di suddividere il dataset in un numero prefissato di cluster (k) in modo che la distanza tra le osservazioni all'interno di ciascun cluster sia minima e la distanza tra i centroidi dei cluster sia massima. L'algoritmo inizia assegnando casualmente i punti ai cluster e poi ripete il processo di assegnazione finché i centroidi dei cluster non si stabilizzano.
È stato utilizzato il dataset generato in seguito all'elaborazione della knowledge base ovvero generated dataset.csv e le feature selezionate sono state divise in categoriche e numeriche.
Ovviamente le variabili categoriche sono state convertite in numeriche attraverso l'ordinalencoder.
Le feature categoriche sono genre,director,writer,star,country,company,month.
Le feature numeriche sono invece queste: year,score,votes,runtime.
Per determinare il valore più appropriato di k (numero di cluster) si è utilizzata l'elbow rule: all'interno di esso vi è il punto di "gomito" sul grafico indica il numero di cluster in cui l'aggiunta di un ulteriore cluster non produce un miglioramento significativo dell'inerzia (cioè la somma dei quadrati delle distanze tra ogni osservazione e il centroide del suo cluster).



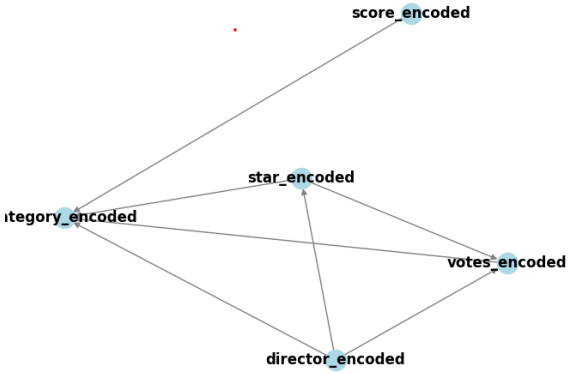
Si è optato per la scelta k = 4, che sembra fornire una migliore suddivisione.
Per visualizzare graficamente i film suddivisi in cluster è stato realizzato il grafico scatter seguente:



Per valutare la qualità del clustering, è stato calcolato l'indice di silhouette medio, che valuta quanto ogni osservazione all'interno di un cluster sia simile ad altre osservazioni nello stesso cluster rispetto a quelle presenti in cluster diversi. L'indice di Silhouette medio restituisce un valore compreso tra -1 e 1, dove un valore più vicino a 1 indica che i cluster sono ben separati e che le osservazioni all'interno di ogni cluster sono molto simili, mentre un valore più vicino a -1 indica che i cluster sono poco separati e che le osservazioni all'interno di ogni cluster sono poco simili.
Nel nostro caso equivale a 0.7038066908366415 un valore vicino ad 1 per cui, come si evince anche dal grafico, i cluster sono ben separati.

Belief Network

Infine, l'ultimo task sviluppato è il calcolo della probabilità che un film possa essere un big_flop, un flop, un success o masterpiece.
Per fare ciò è stata realizzata una Belief Network, anche nota come rete bayesiana o rete di probabilità grafica, cioè un tipo di modello grafico probabilistico utilizzato per rappresentare e ragionare su relazioni di dipendenza probabilistica tra diverse variabili. Consiste in un grafo diretto aciclico (DAG) in cui i nodi rappresentano le variabili e gli archi rappresentano le relazioni di dipendenza probabilistica tra di esse. Ogni nodo ha associata una distribuzione di probabilità condizionata, che descrive la probabilità della variabile in funzione dei suoi genitori, ovvero le variabili a cui è direttamente collegata.
Le feature categoriche sono director e star, mentre le numeriche sono score e votes.
Queste feature sono state aggiunte come nodi alla rete e sono stati definiti gli archi tra 'category' e tutte le altre feature, oltre che tra le feature stesse. Dopo aver aggiunto i nodi alla rete, sono state create le tabelle di distribuzione di probabilità condizionata per ciascun nodo (CPT o TabularCPD), che rappresentano le probabilità condizionate delle variabili, cioè la probabilità che una variabile si verifichi dato lo stato delle sue variabili genitore.



Per ogni feature è stato definito il numero di possibili valori per ogni variabile, utilizzato in seguito per la creazione delle CPT.
Per i nodi senza genitori, come ad esempio il nodo 'director_encoded', sono state semplicemente calcolate le frequenze del nodo corrente basandosi sui valori osservati nel dataset. Per il nodo con genitori, cioè 'category_encoded', sono state calcolate le frequenze delle diverse combinazioni di valori dei genitori e del nodo corrente nel dataset.
La rete bayesiana terrà conto solo delle combinazioni di valori osservate nel dataset durante il ragionamento. Questo procedimento è stato realizzato mediante la classe 'TabularCPD' fornita dal modulo 'pgmpy', che consente di definire le distribuzioni di probabilità condizionata utilizzando tabelle di probabilità. Le CPT create sono state poi aggiunte alla rete bayesiana.
Viene eseguita l'inferenza esatta, utilizzando le evidenze per calcolare la probabilità condizionata dell'evento 'category_encoded'. Nel caso di studio, l'evidenza considerata è la seguente: 'score_encoded': 1, 'star_encoded': 3,'director_encoded':5,'votes_encoded': 2(i valori sono scelti a caso)
La probabilità risultante è

category_encoded	phi(category_encoded)
category_encoded(0)	0.3988
category_encoded(1)	0.0343
category_encoded(2)	0.4984
category_encoded(3)	0.0685

Conclusioni

Il caso di studio comprende quattro task principali che spaziano tra diversi argomenti del corso di Ingegneria della Conoscenza. Questi task potrebbero essere utilizzati, per possibili estensioni per valutare nuovi film o serie tv con lo stesso criterio. Oppure si potrebbe aggiungere al dataset nuove feature per fornire uno strumento più completo utile ai produttori per valutare se investire su un film o meno.