Relazione

Università Ca'Foscari Venezia

11 giugno, 2021

Data and Web Mining

Sofia Crudu (876335)

Sommario

1. Progetto TMBD Box Office Prediction Kaggle

Tale Progetto è incentrato sulla predizione delle entrate di un set di film, i cui dati sono presentati sul sito al seguente link: https://www.kaggle.com/c/tmdb-box-office-prediction.

Il dataset contiene 7398 esemplari di film, suddiviso in due dataset separati: train.csv e test.csv.

2. Glossario

Features = colonne del dataset, contenenti le caratteristiche di ogni singolo film.

Linear Regression = modello di Machine Learning che presuppone una relazione lineare tra le variabili di input (X) e la variabile target (y).

Decision Tree Regression = modello di regressione che assume la forma di un albero di decisione.

Random Forest Regression = un altro modello di regressione che opera costruendo una moltitudine di alberi decisionali al momento dell'apprendimento e fornendo la media (regressione) dei singoli alberi.

Label-Encoding = implica la conversione di ogni valore in una colonna in un numero.

One-Hot-Encoding = qui è dove la variabile categoriale, invece di assumere una codifica numerica, viene rimossa e viene aggiunta una nuova variabile binaria per ogni valore univoco assunto dalla variabile originale.

RMSE (Root Mean Squared Error) = errore nella predizione di un modello.

Score = basso errore, in altre parole minimo RMSE, oppure a volte segnato come valore di accuratezza basato sull'RMSE più basso.

3. Data Loading

3.1 Initial exploration of the data

La parte di data loading è stata quella più corposa data la diversità del formato dei dati presenti nelle singole colonne.

Una volta caricati i dati si può notare che ci sono colonne che contengono valori NAN. Questi ultimi verranno sostituiti nella fase di feature engeeniring con valori più opportuni, come ad esempio con la mediana nella colonna 'budget' per non rischiare di sporcare il resto dei dati con questi outliers oppure con il valore 0.0 nelle colonne che sono candidate ad essere trasformate in variabili categoriali binarie (tagline, bellongs_to_colection, homepage, keywords ecc). Altre colonne invece, diverse da 'id', come 'title, 'original_title', 'imdb_id' e 'poster_path' non presentano alcuna informazione utile in quanto univoche, motivo per cui verranno tolte. Le colonne che hanno un valore costante inoltre sono ugualmente da scartare poiché toglierle o lasciarle non influisce in nessun modo sulla qualità della predizione.

Si nota subito che ci sono colonne che contengono liste di dizionari, 'production_companies', 'production_countries', 'cast', 'crew', 'Keywords', 'belongs_to_collection', strutture che necessitano di una lavorazione a se stante per riuscire a dedurne pattern utili.

Altre features che sembrano espressive ai fini di una analisi migliore del dataset sono i campi categoriali, come per esempio le colonne 'genres' e 'original_language'. Questi possono essere trasformati in numeri attraverso il Label-Encoding oppure usando l'One-Hot-Encoder.
Inoltre la colonna della data di release necessita a sua volta di uno split in più features nel formato giorno/mese/anno/quadrimestre/weekday, che si pensa possano essere significative.

4. Data analysis and feature engineering

In questa parte verranno presentate le funzioni utili a rendere i dati 'grezzi' più espressivi e più adatti alla manipolazione da parte di strumenti di machine learning. In seguito verranno descritte quelle più importanti, esaminandone le funzionalità caratteristiche.

4.1 Funzione data_processed

Come già anticipato, molte delle colonne presenti contengono dati sotto forma di json, in quanto tali, sicuramente hanno bisogno di essere trattate a parte. Si è scelto di riempire i valori mancanti in tali colonne con la stringa vuota, dopo di che andare alla ricerca tramite una regexp delle parole chiave in queste lunghissime liste di dizionari. All'estrazione segue subito il loro inserimento in liste, posizionate in ordine in un'unica nuova colonna. A questa colonna viene applicato il conteggio degli elementi presenti in ogni lista, ottenendo infine una colonna di numeri che rispecchiano la lunghezza di tutte le liste. Come parametri la funzione prende un dataframe e la colonna di interesse. Le colonne per cui si è utilizzata la funzione sono: cast, crew, genres, production companies, production countries e spoken languages.

4.2 Funzione genres_by_mean_revenue

Con l'aiuto di questa funzione viene calcolata la media dei revenues per ogni genere di film. Innanzitutto si utilizza una funzione ausiliaria, **unpackCol**, che serve a 'srotolare' la lista di generi caratteristici di ogni film, ossia fare in modo che i generi diventino colonne autonome con valore 1 per i film in cui sono presenti, 0 altrimenti. Dopo questa fase di one-hot-encoding viene chiamata la funzione **genres_by_mean_revenue**, che restituisce un dizionario avente come chiave tutti i generi del dataset e come valore la media delle entrate, in ordine decrescente rispetto alla media. Ottenuto tale dizionario, nel dataframe sono lasciate solo le colonne dei primi 5 generi che hanno ottenuto la media più alta, tutte le altre sono cancellate.
Sempre nella colonna 'genres' vengono effettuate altre modifiche. Si provvede a realizzare un label-encoding delle liste di generi di ogni film utilizzando a tale scopo la funzione **genre_encoding**.

4.3 Funzioni top_actors e check_top_actors

In particolare, quello che fa la funzione **top_actors** è calcolare per ogni attore la media dei revenues dei film di cui ha fatto parte, in ordine crescente rispetto al valore della revenue totalizzata. La struttura dati usata è il dizionario, che per chiave ha il nome dell'attore e per valore la media dei revenues. Una volta applicata la funzione e ottenuto il dizionario, tramite la funzione **check_top_actors** viene estrapolato un 'rank' che indica il numero degli attori presenti nel top dei primi 100 con la media più alta.

In questo modo, nel training set avremo questa feature aggiuntiva che dà uno specifico 'punteggio' ad ogni film in base al principio spiegato nelle ultime righe. Inoltre, si prende l'occasione per far notare che oltre al numero dei partecipanti del cast e del crew per ogni film, si è pensato di distinguere rispetto al 'gender' della persona, per cui si contano il numero dei maschi e il numero delle femmine rispettivamente nelle colonne 'num_female_cast/crew' e 'num_male_cast/crew'. Per la colonna 'crew' altre features interessanti che si è pensato valesse la pena di aggiungere sono 'num_directors', 'num_producers' e 'num_editors'.

4.4 Funzione date_features

La funzione **date_features** si prefigge di suddividere la data di release in più features. Questa operazione è resa semplice grazie al package datetime. Le nuove colonne ottenute tramite questo split sono: 'release_month', 'release_day', 'release_year', 'release_dayofweek', 'release_quarter'. Inoltre viene fatto un fix sull'anno in quanto abbiamo valori in un formato scorretto, come ad esempio '2011' con 4 cifre e '98' con solo 2 cifre. In seguito ad una ulteriore analisi, vediamo come i revenues crescono con gli anni a passare, questo è il motivo per cui è importante implementare queste features nel modello.

Inoltre sembra che ci sia una correlazione tra il quadrimestre e la revenue, per cui teniamo queste informazioni come features aggiuntive con l'one-hot-encoding.

4.5 Altre funzioni

Per altre features viene utilizzata la stessa tecnica menzionata prima di label-encoding. Prima vengono riempiti i valori mancanti con la stringa '0'. Ogni feature ha la sua funzione ben specifica.

Altre features interessanti che si è pensato valesse la pena di aggiungere sono 'in_english', in quanto si è osservato che i film in inglese raggiungono ricavi molto più alti, quindi questa potrebbe essere una buona feature da includere. Dal momento che la maggioranza dei film sono stati originariamente rilasciati in inglese e tutte le altre lingue costituiscono solo una piccola parte dei film, creeremo questa nuova variabile dummy, specificando semplicemente se il film è stato rilasciato in inglese o meno. Sembra che non sia importante quale sia l'altra lingua specifica, ma piuttosto il fatto che il film non sia inglese è più significativo per il nostro modello.

Possiamo notare dalla colonna 'production_countries' che la stragrande maggioranza dei film ha gli Stati Uniti elencati come paese di produzione. A causa di questa osservazione, non esamineremo tutti i diversi paesi di produzione, ma includeremo invece una feature per determinare se un film è stato prodotto o meno negli Stati Uniti, 'usa_produced'.

Altre features che assumono la forma di variabili categoriali binarie sono 'tagline', 'homepage', 'Keywords' e 'belongs_to_collection'. Per le colonne che contengono già dati numerici, come 'budget' e 'runtime', si è pensato di sostituire i valori NAN con la mediana, che essendo meno sensibile agli outliers rispetto alla media si è rilevata la migliore opzione da adottare. Come per la colonna 'budget' è stata creata una colonna aggiuntiva, 'budget_is_median', per indicare se la tupla corrispondente del budget è stata inserita come mediana o meno, così anche per la colonna 'runtime' è stata creata la colonna 'runtime_is_median' con lo stesso scopo.

Alla colonna overview si è pensato di applicare il conteggio delle parole presenti in ogni riga.

Colonne come ‘id’, ‘title’, ‘imbd’ e ‘status’ vengono droppate, in quanto risultano univoche o costanti, di conseguenza non utili ai fini dell’apprendimento da parte del modello che andiamo a costruire.

4.6 Conclusioni

Tutte queste funzioni, ognuna specifica per la propria feature, vengono raggruppate e chiamate nella funzione **prepare_data**, che permette di essere chiamata con pochi parametri, dando in output un dataset più pulito ed espressivo, su cui si potrà applicare in seguito gli strumenti di machine learning.

5. Training e Modeling

Si sono utlizzati principalmente tre strumenti per la ricerca del modello di predizione più adatto: la **regressione lineare**, gli **alberi di decisione** e la **random forest**.

Innanzitutto dobbiamo prendere il log della variabile target, poiché la metrica della competizione è RMSLE (Root Mean Squared Log Error).

5.1 Regressione Lineare

La regressione lineare è stata scelta perché intrinsecamente il task è di regressione, cioè richiede che la predizione sia un valore continuo. Alla fine la regressione ci darà un modello che verrà usato solo come benchmark per verificare che l’albero di regressione superi la regressione. La regressione lineare viene implementata usando la libreria scikit-learn. Infatti verrà fatto un confronto grafico tra l’andamento del RMSE della regressione e quello dell’albero di decisione. Si potrà comunque notare che la regressione funzionerà degnamente.

5.2 Alberi di Decisione

Il vantaggio di usare un albero di decisione è che i dati vengono divisi in sotto-nodi a sinistra e sotto-nodi a destra in base a certe misure di selezione degli attributi, per semplicemente diminuire l’RMSE. Si è utilizzato il validation set per prendere decisioni sui parametri da utilizzare (es. numero di foglie) per fare le simulazioni sul test set in seguito. Inoltre si è ricorso anche al k-fold Cross-validation, con k=5, per attenuare l’effetto di dati specifici presenti nel dataset. Si è potuto osservare che un aumento eccessivo del numero di foglie causava un aumento graduale dell’RMSE. Ciò è dovuto al fatto che alberi con un elevato numero di foglie sono soggetti a overfitting, in quanto il modello, più attraversa input diversi, più va incontro alla perdita stabilità sulle predizioni delle varie istanze. Questo è possibile mitigare tramite la tecnica di bagging, che serve proprio a diminuire la varianza.

5.3 Random Forest

Le Random Forest sono degli **ensemble methods** molto utilizzati per migliorare l’accuratezza della predizione. Un ensemble, combina una serie di k alberi M1,M2,..,Mk, allo scopo di creare un predittore migliore. Un dataset D , viene utilizzato per creare k training set D1,D2,…,Dk.

È un'estensione dell'algoritmo di bagging. Gli stimatori di base nella random forest sono alberi decisionali. A differenza del bagging, la random forest seleziona casualmente un insieme di features che vengono utilizzate per decidere la migliore costruzione del nodo successivo dell'albero decisionale.

Passo dopo passo, questo è ciò che fa un algoritmo di random forest:

1. I sottoinsiemi casuali di features vengono creati a partire dal set di dati originale (bootstrap);

2. Ad ogni nodo dell'albero decisionale, viene considerato solo un insieme casuale di features per decidere lo split migliore;

3. Un modello di albero decisionale viene simulato su ciascuno di tali sottoinsiemi. La previsione finale viene calcolata facendo la media delle previsioni provenienti da tutti gli alberi decisionali.

In particolare, il modello di random forest della libreria sklearn utilizza tutte le features per l'albero decisionale e un sottoinsieme di features viene selezionato in modo casuale per determinare il criterio di splitting.

Per riassumere, la random forest utilizza il bagging insieme a una selezione random degli attributi.

5.4 Conclusioni

Confrontando l'RMSE ottenuto con l'algoritmo di random forest con quelli degli altri modelli descritti sopra, possiamo concludere di aver raggiunto un buon risultato, in particolare il migliore fra tutti.

Inoltre, alla fine viene fatta un'analisi di feature importance simulando un random forest regressor su un numero crescente di features, calcolando man mano l'RMSE ottenuto. L'andamento della variazione dell'RMSE insieme al numero di features associato viene mostrato graficamente.