

Progetto di Data Analytics

A.A. 2022-2023

Daniele Perrella 1038893
Simone Boldrini 1038792

7 aprile 2023

Indice

Capitolo 1

Introduzione

L'obiettivo di questo report è presentare il progetto del corso di Data Analytics finalizzato alla predizione del voto medio di un film, date le sue caratteristiche, utilizzando un dataset proveniente da MovieLens, un recommendation system per contenuti video. Il dataset contiene rating e tag per oltre 60.000 film, raccolti da più di 150.000 utenti negli anni 1995-2019. Ogni file del dataset dispone di un genoma che identifica una caratteristica del film e la sua rilevanza.

Per raggiungere questo obiettivo, abbiamo utilizzato tecniche di Machine Learning supervisionate tradizionali quali Linear Regression, SVM, NB in seguito verranno illustrate nel dettaglio; tecniche di ML basate su Reti Neurali ed infine modelli deep per Tabular Data. In particolare, il report descrive il processo di acquisizione e preparazione dei dati, l'analisi esplorativa del dataset, la selezione delle feature rilevanti e la costruzione dei modelli predittivi.

Infine, il report conclude con una valutazione critica dei modelli creati, evidenziando i loro punti di forza e di debolezza, e suggerisce possibili sviluppi futuri per migliorare ulteriormente la predizione del voto medio dei film.

Lo studio è stato effettuato rispettando le componenti della pipeline studiata durante il corso, che è la seguente:

- Data Acquisition
- Data Visualization
- Data Preprocessing
- Modeling (e tuning degli iperparametri)
- Test & Evaluation

Capitolo 2

Metodologia

2.1 Data Acquisition

In uno studio di Data Analysis, lo step di Data Acquisition è il primo della pipeline da seguire. In questa fase abbiamo raccolto i dati necessari dal dataset [?, MovieLens]. I file sono stati raccolti in una cartella denominata `ml-25m` contenente un README.

Dataset

MovieLens è un Recommendation System per contenuti multimediali, quali film, serie tv, documentari ecc. MovieLens mette a disposizione degli sviluppatori un dataset open-source, generato dal database TMDB (The Movie Database). Questo dataset contiene recensioni con voti e tag di oltre 60.000 film, raccolte da oltre 150.000 utenti durante il periodo che va dal 1995 fino al 2019.

Il dataset è composto dai seguenti file:

- *genome-scores.csv*: contiene il *relevance score* di ogni tag per tutti i film (ovvero, quanto un tag è importante per il dato film).
- *genome-tags.csv*: contiene tutti i tag presenti all'interno del dataset.
- *links.csv*: contiene gli ID di ogni film per i due database TMDB e IMDB.
- *movies.csv*: contiene i titoli dei film, con i rispettivi generi.
- *ratings.csv*: contiene più di 25.000.000 votazioni provenienti dalle recensioni degli utenti.

- *tags.csv*: contiene i tag che sono stati assegnati ai film dagli utenti nelle rispettivi recensioni.

Per il nostro caso di studio, non abbiamo fatto uso di tutti i file disponibili, ma esclusivamente quelli che abbiamo considerato adatti per lo scopo. Di seguito viene mostrata la struttura dei tali.

Il dataset ***ratings*** è stato selezionato per poter far uso dei voti assegnati ai film dagli utenti, per poter quindi ottenere il voto medio di tali film. La struttura è la seguente

userId	movieId	rating	timestamp
1	296	5.0	1147880044
1	306	3.5	1147868817
1	307	5.0	1147868828
1	665	5.0	1147878820
1	899	3.5	1147868510
...

Tabella 2.1: ratings.csv

Il dataset ***genome-scores*** è stato selezionato per poter identificare le relazioni che ci sono tra un dato voto ed i valori dei tag assegnati

movieId	tagId	relevance
1	1	0.028749999999999998
1	2	0.023749999999999993
1	3	0.0625
1	4	0.075749999999999998
1	5	0.14075
...

Tabella 2.2: genome-scores.csv

Infine, si è fatto uso del dataset ***movies*** per ottenere le informazioni inerenti al genere di ogni film.

movieId	title	genres
1	Toy Story (1995)	Adventur Animation ...
2	Jumanji (1995)	Adventure Children ...
3	Grumpier Old Men (1995)	Comedy Romance
4	Waiting to Exhale (1995)	Comedy Drama ...
...

Tabella 2.3: movies.csv

2.2 Data Visualization

Nella fase di *data visualization* andremo a visualizzare alcune proprietà e peculiarità del dataset.

Distribuzione dei ratings

Nella seguente figura, è mostrata la distribuzione dei voti sul dataset.

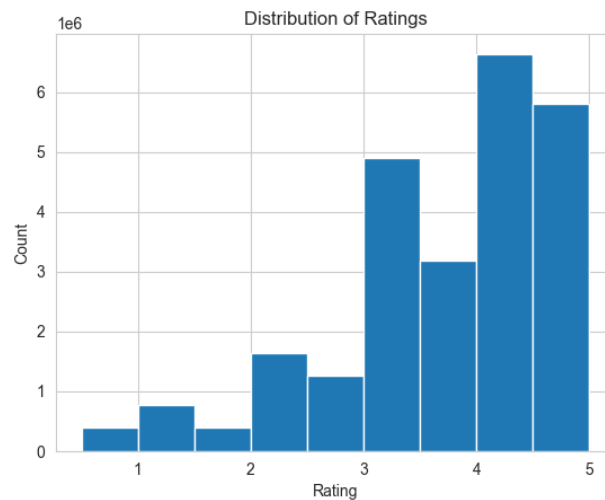


Figura 2.1: Distribution of Ratings.

Il grafico mostra la distribuzione dei voti sul dataset, evidenziando un chiaro sbilanciamento nella distribuzione. In particolare, si può osservare una concentrazione significativamente maggiore di voti con valore di 3 o superiore rispetto ai voti inferiori. Questo sbilanciamento è ulteriormente evidenziato dal fatto che i voti con valore di 3 o superiore sono 6 o più volte maggiori rispetto ai voti più bassi. Tale distribuzione suggerisce una maggiore prevalenza di valutazioni positive rispetto a quelle negative o neutre nel dataset.

Questo risultato potrebbe essere utile per comprendere meglio le caratteristiche del dataset, ad esempio potrebbe suggerire la presenza di una tendenza positiva nelle valutazioni, o la necessità di bilanciare meglio le categorie di voto.

Relazione tra numero di voti e voto medio

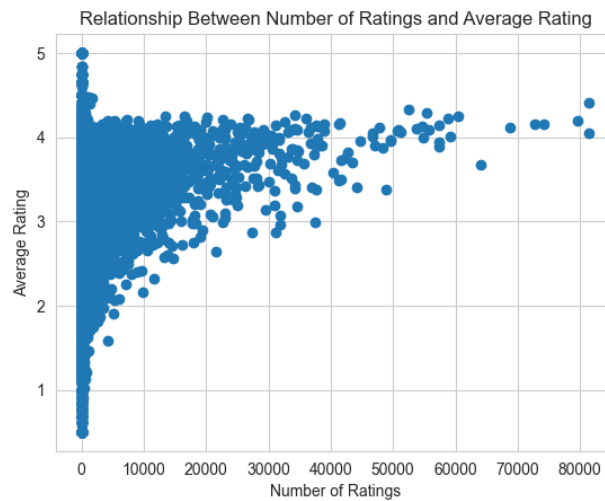


Figura 2.2: Relationship between Number of Ratings and Average Rating.

Il grafico mostra la relazione tra il numero di voti e il voto medio, evidenziando una tendenza simile ad una funzione esponenziale. In particolare, il grafico mostra un aumento costante del voto medio all'aumentare del numero di voti, con un picco raggiunto intorno al valore 4. Tuttavia, a partire da questo punto, si verifica una diminuzione improvvisa del voto medio per poi stabilizzarsi in prossimità del valore neutro. Questa tendenza suggerisce che il numero di voti ha un impatto significativo sul voto medio ricevuto, in particolare nei casi in cui il numero di voti è relativamente basso. Tuttavia, per i voti più alti, sembra che la quantità di voti ricevuti non abbia un impatto significativo sul risultato finale. Inoltre, la diminuzione improvvisa del voto medio tra 4 e 5 suggerisce che esiste una soglia critica oltre la quale i voti ricevuti possono avere un effetto negativo sul voto medio complessivo.

Voto medio per Genere

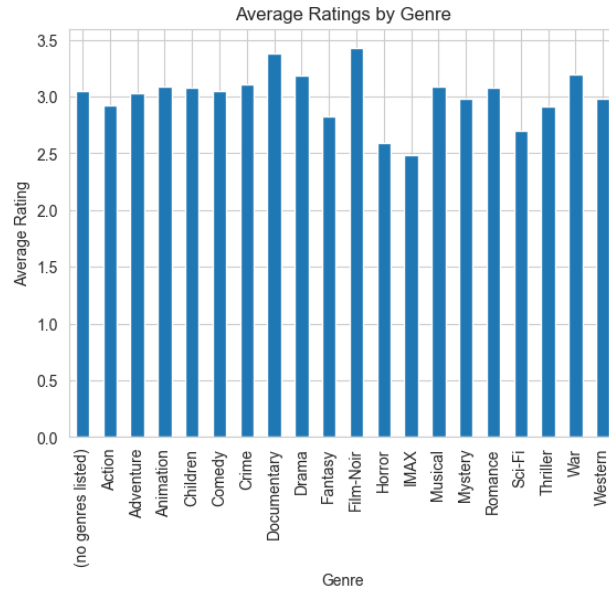


Figura 2.3: Average Ratings by Genre.

Il grafico mostra il voto medio per genere e rivela un andamento generalmente stabile, con tutti i valori che si attestano intorno al valore 3. Alcuni valori si avvicinano al 3.5, indicando una maggiore gradazione positiva, mentre altri si avvicinano al valore 2.5, indicando una maggiore gradazione negativa. Questa distribuzione dei voti suggerisce che il genere non sembra influenzare significativamente il voto medio complessivo, ma piuttosto, la valutazione sembra essere determinata in base alle caratteristiche specifiche della valutazione stessa. Tuttavia, esistono alcune differenze tra i voti attribuiti dai generi, con alcune categorie che mostrano una maggiore tendenza verso valutazioni positive rispetto ad altre. Ad esempio, il genere "commedia" sembra essere valutato in modo più positivo rispetto al genere "drammatico". Questi risultati potrebbero suggerire che le preferenze personali dei valutatori influenzano le loro valutazioni, oltre alle caratteristiche del film stesso.

2.3 Tabular Data

Capitolo 3

Implementazione

3.1 Data Preprocessing

La fase di preprocessing dei dati consiste nel preparare i dati per l'addestramento del modello. Una volta acquisiti i dati e averne studiato le peculiarità e le caratteristiche, si procede con la pulizia dei dati. Abbiamo usato il metodo *pivot_table* di Pandas per creare una tabella pivot che contiene i genome-score degli utenti per ogni film che lo possiede (andando così a ridurre la cardinalità da 60k a 13.816).

In seguito a ciò verrà effettuata una merge con un i generi dei film. Ci siamo inoltre andati a focalizzare sui voti da parte degli utenti, raggruppandoli per film ci abbiamo calcolato la media dei voti. Quest'ultimo capo sarà il nostro *target* per l'addestramento del modello.

3.2 Modeling

La nostra fase di modellazione include uno studio di regressione basato sulla predizione del voto medio di un film dati i suoi genome-score (ed in seguito anche i generi). Abbiamo inoltre applicato una tecnica di *PCA* per ridurre la cardinalità dei dati, ma senza ottenere risultati soddisfacenti, quindi è stato deciso di non farne uso ulteriormente.

Eseguiamo il train-test split con un rapporto 80-20 ed in seguito addestreremo il nostro modello.

3.2.1 Tecniche di ML Supervisionate con Approccio non-Deep

In questa sezione verranno descritte le tecniche di ML supervisionate con approccio non-Deep utilizzate per la predizione del voto medio di un film.

- **Linear Regression:** La regressione lineare è una tecnica di ML supervisionata che permette di predire il valore di una variabile dipendente a partire da una o più variabili indipendenti.
 - **Ridge:** Ridge è un algoritmo di regressione lineare che utilizza un termine di regolarizzazione per evitare problemi di multicollinearità. Il parametro di regolarizzazione controlla l'importanza del termine di regolarizzazione nella funzione di costo
 - **Lasso:** Lasso è un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un modello lineare.
- **Decision Tree:** Decision Tree è un algoritmo di ML supervisionato che permette di effettuare la classificazione o la regressione di un dato mediante la creazione di un albero di decisione.
- **Random Forest:** Random Forest è un algoritmo di ML supervisionato che permette di effettuare la classificazione o la regressione di un dato mediante la creazione di più alberi (Metodo di Ensemble Learning).
- **SVR:** Support Vector Regression è un algoritmo che cerca di trovare una funzione che minimizzi la distanza tra i dati di training e una fascia di tolleranza definita dall'utente.
- **KNN:** K-Nearest Neighbors è un algoritmo di ML supervisionato che permette di effettuare la classificazione o la regressione di un dato cerca di stimare il valore di una variabile dipendente su nuovi dati in base alla loro vicinanza ad altri dati di training.

Per ogni modello precedentemente citato è stato applicato il Tuning dei parametri per cercare di ottimizzare il modello. Attraverso la funzione *Grid-Search* di Scikit-Learn abbiamo cercato di trovare i migliori parametri per ogni modello.

Tuning dei Parametri

Per ogni modello precedentemente citato è stato applicato il Tuning dei parametri per cercare di ottimizzare il modello. Attraverso la funzione *GridSearch* di Scikit-Learn abbiamo cercato di trovare i migliori parametri per ogni modello, come segue:

```
1  def tuning(self):
2      search = GridSearchCV(estimator = self.svr, param_grid= self.
    params, cv = 3, n_jobs = 6)
3      search.fit(self.X_train_t, self.y_train)
4      print(search.best_params_)
5      print(search.best_score_)
6      self.svr = search.best_estimator_
```

3.2.2 Linear Regression

Il modello di Linear Regression ci permette di operare una regressione lineare tra i nostri dati (una regressione multivariable). In questo specifico modello non abbiamo bisogno particolari parametri da fare tuning. Di conseguenza andremo a vedere i risultati del tuning per i modelli che usano la regressione lineare: Lasso e Ridge

Ridge

Ridge é un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un modello lineare. Si basa su un valore di regolarizzazione che permette di evitare problemi di multicollinearità. Per cercare il valore α apposito abbiamo utilizzato la funzione *RidgeCV* di Scikit-Learn, che permette di trovare il valore ottimale di α , ed effettuare la Cross Validation.

Modello	senza PCA	PCA
Lambda	6.0	2.5

Lasso

Lasso é un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un modello lineare. Abbiamo cercato anche qui di ottimizzare il valore λ abbiamo utilizzato la funzione *LassoCV* di Scikit-Learn, che permette di trovare il valore ottimale di λ , ed effettuare la Cross Validation.

Modello	senza PCA	PCA
Alpha	7.425262873115622e-05	0.0005675190692206821

3.2.3 Decision Tree

Decision Tree é un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un albero di decisione. Abbiamo avuto bisogno di fare il tuning di piu parametri, in particolare:

```

1 self.params = {
2     'splitter' : ['best','random'],
3     'max_depth' : [None, 20,50,100],
4     'min_samples_split' : [10, 50, 100, 200],
5     'min_samples_leaf' : [2, 5, 10, 20],
6     'max_features' : ['auto', 'sqrt', 'log2'],
7     'max_leaf_nodes' : [None, 5, 10, 20],
8 }

```

Abbiamo utilizzato la funzione *GridSearchCV* di Scikit-Learn per trovare i migliori parametri per il modello, utilizzando un CV di 3 volendo mantenere coerenza con altre ricerche che con una CV maggiore avrebbero portato ad un incremento considerevole del tempo. Di seguito i risultati ottenuti:

Parametri	senza PCA	PCA
max_depth	None	None
max_features	'auto'	'auto'
max_leaf_nodes	None	None
min_samples_leaf	20	20
min_samples_split	50	100
splitter	'best'	'best'

3.2.4 Random Forest

Random Forest é un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di più alberi (Metodo di Emsable Learning). I parametri su cui effettueremo il tuning sono:

```

1 self.params = {
2     'n_estimators': [50,100,500],
3     'max_depth': [None ,10, 50, 100],
4     'min_samples_split': [2, 5, 10],
5     'min_samples_leaf': [1, 2, 5],
6     'max_features': ['auto', 'sqrt', 'log2'],
7 }

```

Ottendendo i seguenti risultati:

Parametri	senza PCA	PCA
max_depth	None	None
max_features	'auto'	'auto'
min_samples_leaf	1	1
min_samples_split	2	2
n_estimators	500	500

3.2.5 SVR

SVR é un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un modello lineare. Due parametri rilevanti sono C e ϵ : C é il parametro di regolarizzazione, mentre ϵ é il parametro che permette di definire la tolleranza per il modello.

```

1 self.params = {
2     'kernel': ['linear', 'poly', 'rbf', 'sigmoid'],
3     'C': [0.01, 0.1, 1, 10, 100],
4     'gamma': [0.001, 0.01, 0.1, 1, 10],
5     'epsilon': [0.01, 0.1, 1, 10]
6 }
```

Ottendendo i seguenti risultati:

Parametri	senza PCA	PCA
C	1	100
epsilon	0.01	0.01
gamma	0.01	0.001
kernel	'rbf'	'rbf'

3.2.6 KNN

KNN é un algoritmo di ML supervisionato che permette di effettuare una regressione sui dati trovando il target predetto da un interpolazione locale dei target associati ai k più vicini. Abbiamo effettuato il tuning di parametri:

```

1 self.params = {
2     'n_neighbors': [3, 5, 7, 13, 17, 25],
3     'weights': ['uniform', 'distance'],
4     'algorithm': ['ball_tree', 'kd_tree', 'brute'],
5     'leaf_size': [10, 30, 50, 80, 100],
6 }
```

Con i seguenti risultati:

Parametri	senza PCA	PCA
algorithm	ball_tree	ball_tree
leaf_size	10	10
n_neighbors	13	13
weights	'distance'	'distance'

3.2.7 Tecniche di ML Supervisionate con Reti Neurali

Mostriamo di seguito la nostra Implementazione della Rete Neurale Feed Forward, che abbiamo utilizzato per effettuare la regressione dei dati. Abbiamo utilizzato il framework Pytorch per la creazione della rete neurale, che ci ha permesso di utilizzare la GPU per l'addestramento della rete.

```

1  # Define the neural network architecture
2  self.model = nn.Sequential(
3      nn.Linear(X.shape[1], hidden_size1),
4      nn.ReLU(),
5      nn.Linear(hidden_size1, hidden_size2),
6      nn.ReLU(),
7      nn.Linear(hidden_size2, 1)
8  )
9
10 self.criterion = nn.MSELoss()
11 self.optimizer = optim.Adam(self.model.parameters(), lr=lr)

```

Abbiamo effettuato il tuning degli iperparametri della rete neurale utilizzando `itertools.product`, che ci ha permesso di effettuare il tuning di tutti i parametri in maniera semplice e veloce. I parametri che abbiamo utilizzato sono:

```

1  self.params = {
2      batchsize = [128,256,512,1024]
3      hidden_size1 = [64,128,256]
4      hidden_size2 = [64,128,256]
5      lr = [0.0001,0.001,0.01]
6      num_epochs = [200,400,600]
7  }

```

I risultati ottenuti sono:

Parametri	senza PCA	PCA
batchsize	1024	256
hidden_size1	256	128
hidden_size2	256	256
lr	0.01	0.001
num_epochs	400	600

3.3 Tabular Data

Per quanto riguarda i Tabular Data abbiamo fatto il tuning dei seguenti iperparametri(sempre utilizzando `itertools.product`):

```
1  batchsize = [512,1024,2048]
2  width = [8,16,32]
3  steps = [3,5,7]
4  learning_rate = [2e-2,1e-2,5e-3]
5  max_epochs = [70,120,150,210]
```

dove *width* corrisponde a n_d e n_a dove n_d é la Larghezza del livello di previsione delle decisioni(n_a nella documentazione viene consigliato sia uguale a n_d , per questo motivo utilizziamo un singolo valore). Viene fatto uso di Early Stopping per evitare l'overfitting. I risultati ottenuti sono:

Parametri	senza PCA	PCA
batchsize	2048	-
width	8	-
steps	5	-
learning_rate	0.02	-
max_epochs	210	-

Capitolo 4

Risultati

In questo capitolo verranno presentati e discussi i risultati ottenuti nelle diverse configurazioni descritte nei capitoli precedenti.

4.1 Performance Evaluation

Per valutare la qualità dei modelli di ML Supervisionati Deep e non Deep, sono stati utilizzati i seguenti metri di performance:

- **MSE: Mean Squared Error**
Mean Squared Error (MSE) é una media delle differenze al quadrato tra i valori predetti e quelli reali.
- **RMSE: Root Mean Squared Error**
Root Mean Squared Error (RMSE) é la radice quadrata della media delle differenze al quadrato tra i valori predetti e quelli reali.
- **R2: Coefficient of Determination**
Coefficient of Determination (R2) é una misura di quanto i valori predetti siano vicini ai valori reali.
- **MAE: Mean Absolute Error**
MAE è una media delle differenze assolute tra i valori predetti e quelli reali.

4.2 Tecniche di ML Supervisionate non Deep

Model	MSE	RMSE	R2	
Linear Regression	0.005277524569837709	0.0726465730082136	0.9779884023114186	0.0
Lasso	0.005486814781830245	0.07407303680712872	0.9771154908004298	0.0
Ridge	0.005173317239238808	0.0719257759029321	0.9784230321851388	0.0
K Neighbors	0.04014675063926314	0.20036654071791313	0.8325551853181739	0.1
SVR	0.004249807698698538	0.06519054915168715	0.9822748229629815	0.0
Decision Tree R.	
Random Forest R.	

Tabella 4.1: ratings.csv

Di seguito sono riportati i risultati dei modelli precedenti, ma con l'utilizzo della PCA

Model	MSE	RMSE	R2	
Linear Regression	0.006377020549959092	0.07985624928556996	0.973402604016331	0.0
Lasso	0.009433868123483865	0.09712810161577269	0.9606530472699164	0.0
Ridge	0.0063303117843782316	0.07956325649681661	0.9735974178050478	0.0
K Neighbors	0.03922817004507678	0.1980610260628698	0.836386418354838	0.0
SVR	0.004886523894752017	0.06990367583147554	0.9796191952034381	0.0
Decision Tree R.	
Random Forest R.	0.0120785827307925	0.109902605659704	0.949622422368631	0.0

Tabella 4.2: ratings.csv

Miglior Classic ML

Model	C	Epsilon	Gamma	Kernel
SVR	1	0.01	0.01	rbf

Tabella 4.3: ratings.csv

Miglior Classic ML con PCA

Model	C	Epsilon	Gamma	Kernel
SVR	100	0.01	0.001	rbf

Tabella 4.4: ratings.csv

4.3 Neural Network

Neural Network Results

Model	MSE	RMSE	R2	
NN	0.00664011668413877	0.0814869105815887	0.972389101982116	0.0629
NN With PCA	0.005938166752457619	0.07705949991941452	0.9753079414367676	0.0592

Tabella 4.5: ratings.csv

Miglior NN

Batch Size	Input Layer	Output Layer	lr	Epochs
1024	256	256	0.01	400

Tabella 4.6: ratings.csv

Miglior NN con PCA

Batch Size	Input Layer	Output Layer	lr	Epochs
256	128	256	0.001	600

Tabella 4.7: ratings.csv

4.4 TabNet

TabNet Results

Model	MSE	RMSE	R2	
TabNet	0.00795749657941388	0.0892048013248944	0.964775228814175	0.0
TabNet With PCA	0.005938166752457619	0.07705949991941452	0.9753079414367676	0.0

Tabella 4.8: ratings.csv

Miglior TabNet

Batch Size	width	step	lr	Max Epochs
...

Tabella 4.9: ratings.csv

4.5 Performance Evaluation

4.6 Performance Evaluation on Tabular Data