Progetto di Data Analytics
A.A. 2022-2023

Daniele Perrella 1038893
Simone Boldrini 1038792

21 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 [5], 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 [5]. I file sono stati raccolti in una cartella denominata `ml-25m` contente un README.

Dataset

Movielens è un Recomendation System per contenuti multimediali, quali film, serie tv, documentari ecc. Movielens mette a disposizione degli sviluppatori un dataset opensource, generator dal database TMDB (The Movie Database)[3]. Questo datasedt 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 TMBD e IMDB[4].
- *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 mediio 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	0.07574999999999998
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.

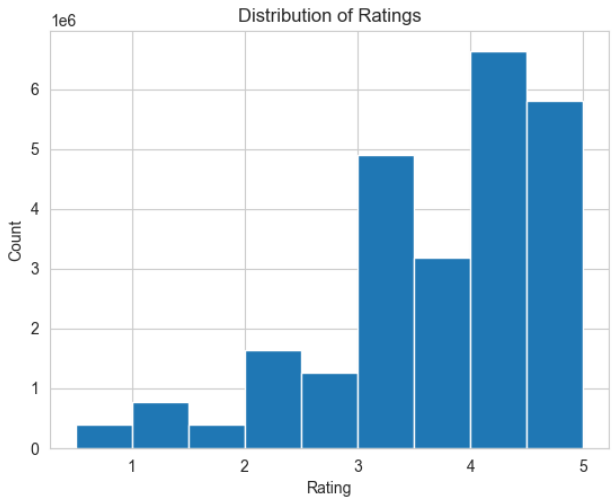


Figura 2.1: Distribution of Ratings.

Il gafico mostra la distribuzine dei voti sul dataset, evidenziando un chiaro sbilanciamento nella distribuzione. In particolare, si può osservare una concentrzione 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

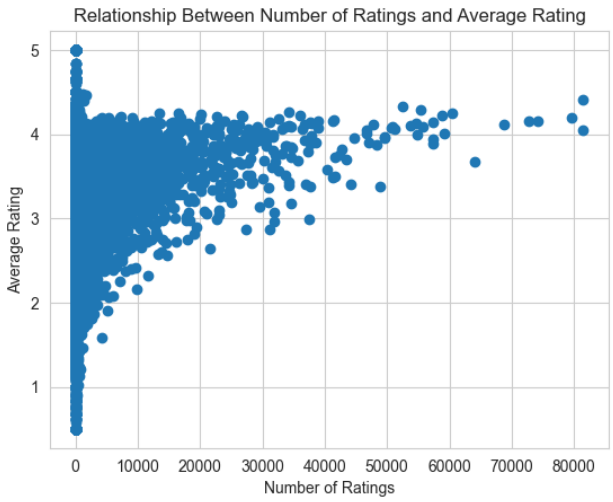


Figura 2.2: Relationship between Number of Ratings and Average Rating.

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 votio 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

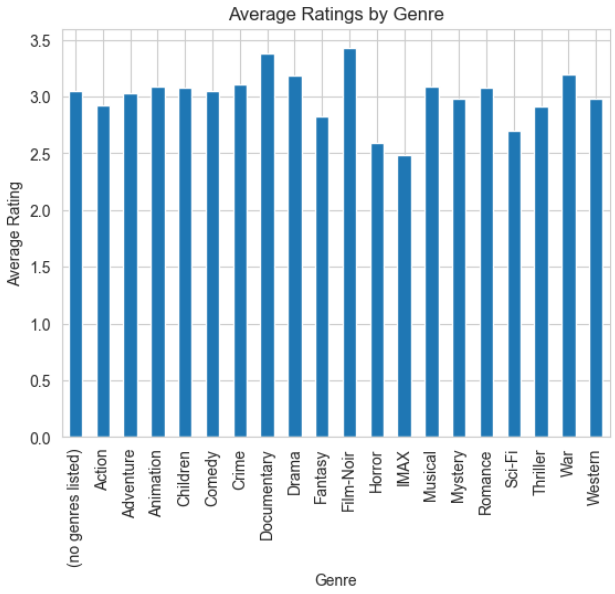


Figura 2.3: Average Ratings by Genre.

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 infuenzano le loro valutazioni, oltre alle caratteristiche del film stesso.

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 cardialità 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 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. Abbiamo inoltre applicato una tecnica di *PCA* con l'obiettivo di ridurre la cardinalità dei dati, eseguendo l'addestramento sia con che senza la PCA su tutti i modelli al fine di confrontarne i risultati.

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 eviare 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.
- **Deicision 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 Emsable 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 *GridSearch* 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 nel seguente esempio:

```
    def tuning(self):
        search = GridSearchCV(estimator = self.svr, param_grid= self.
    params, cv = 3, n_jobs = 6)
        search.fit(self.X_train_t, self.y_train)
        print(search.best_params_)
        print(search.best_score_)
        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 particolari parametri per cui fare tuning. Di conseguenza andremo a vedere i risultati del tuning per i modelli che usano la regressione lineare: Lasso e Ridge

Ridge

Ridge è 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**
α	6.0	2.5

Lasso

Lasso è 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 α abbiamo utilizzato la funzione *LassoCV* di Scikit-Learn, che permette di trovare il valore ottimale di α, ed effettuare la Cross Validation.

Modello	senza PCA	PCA
Alpha	7.425262873115622e-05	0.0005675190692206821

3.2.3 Decision Tree

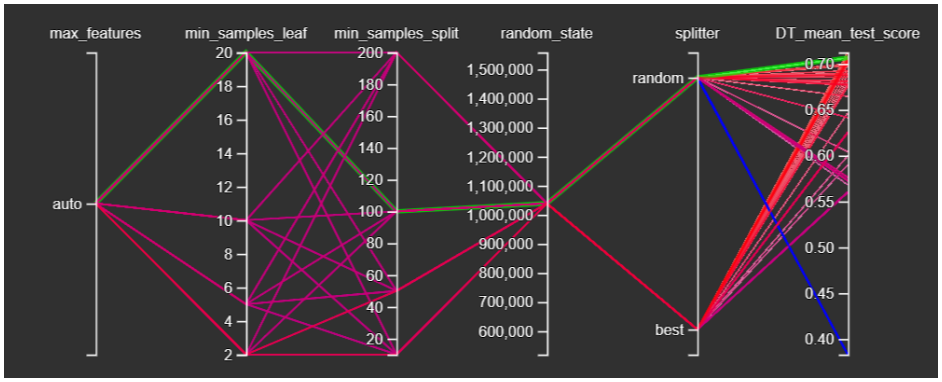
Decision Tree è un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un albero di decisione. Abbiamo avuto bisogno di fare il tuning di piu parametri, in particolare:

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

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'

Mostriamo inoltre il grafico che mostra l'andamento del Mean Test Score in funzione dei parametri, ricordando che nei modelli successivi verrá mostrato il grafico solo con dati senza PCA:



3.2.4 Random Forest

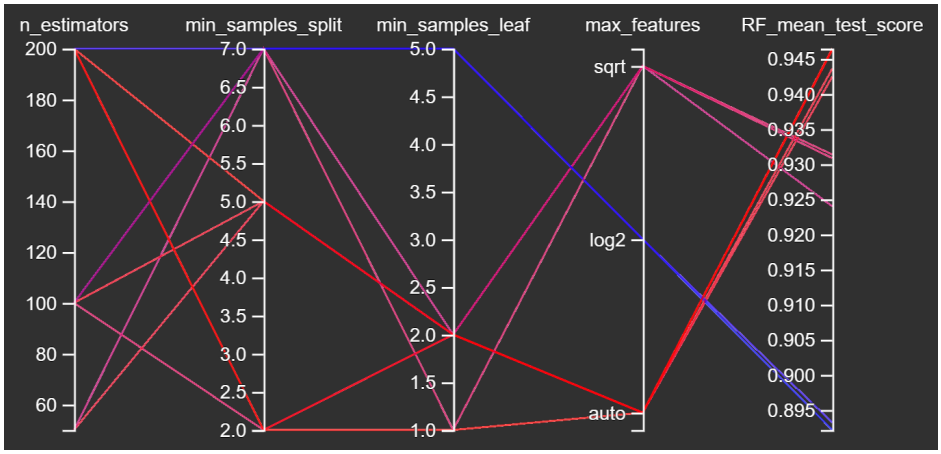
Random Forest è un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di più alberi (Metodo di Emsable Learning). I parametri su cui effettueremo il tuning sono:

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

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

Qui di seguito inoltre riportiamo un grafico che mostra la distribuzione dei parametri e del Mean Test Score ottenuto:



3.2.5 SVR

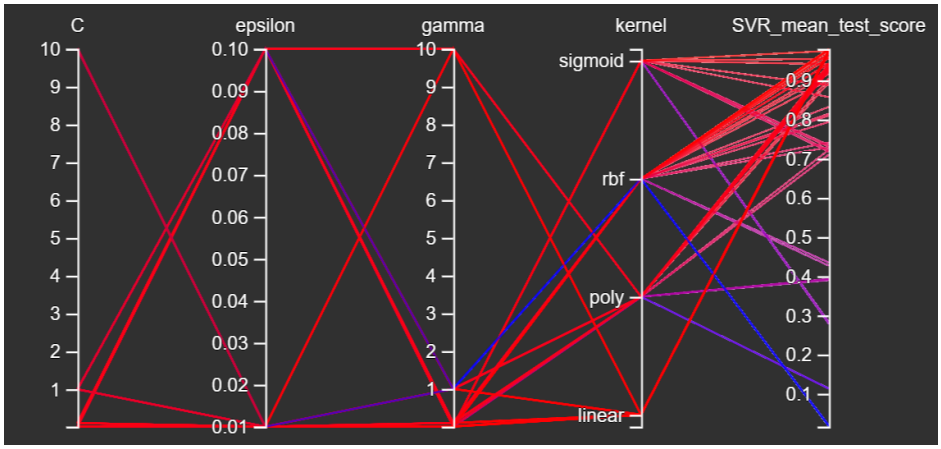
SVR è un algoritmo di ML supervisionato che permette di effettuare la regressione di un dato mediante la creazione di un modello lineare. Due parametri rilvanti sono C e ϵ: C è il parametro di regolarizzazione, mentre ϵ è il parametro che permette di definire la tolleranza per il modello.

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

Ottendendo i seguenti risultati:

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

Come per il modello precedente riportiamo il grafico dell'andamento dei parametri e del Mean Test Score:



3.2.6 KNN

KNN è 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 piu vicini. Abbiamo effettuato il tuning di parametri:

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

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.

```
    # Define the neural network architecture
    self.model = nn.Sequential(
        nn.Linear(X.shape[1], hidden_size1),
        nn.ReLU(),
```

```
        nn.Linear(hidden_size1, hidden_size2),
        nn.ReLU(),
        nn.Linear(hidden_size2, 1)
    )

    self.criterion = nn.MSELoss()
    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:

```
    self.params = {
        batchsize = [128,256,512,1024]
        hidden_size1 = [64,128,256]
        hidden_size2 = [64,128,256]
        lr = [0.0001,0.001,0.01]
        num_epochs = [200,400,600]
    }
```

I risultati ottenuti sono:

Parametri	senza PCA	PCA
batchsize	1024	256
hidden_size1	64	128
hidden_size2	256	256
lr	0.01	0.001
num_epochs	400	600

Visualizziamo il grafico dell'andamento dei parametri e del Mean Test Score:

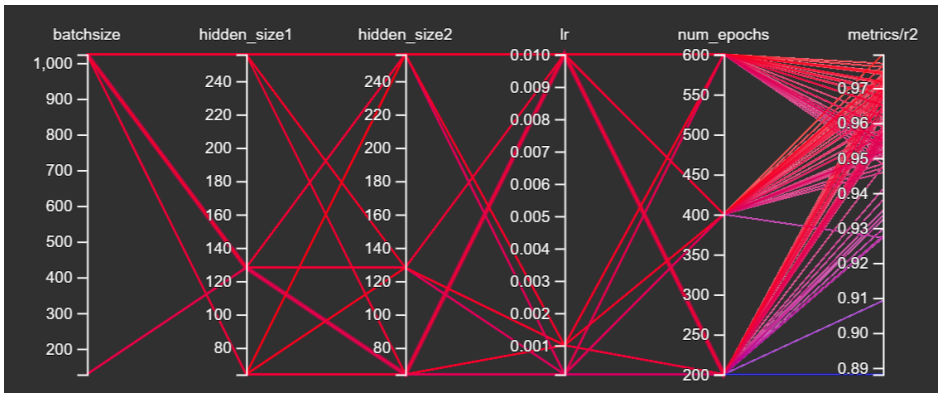


Figura 3.1: Andamento dei parametri e del Mean Test Score per la Rete Neurale (senza PCA)

3.3 Tabular Data

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

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

dove *width* corrisponde a n_d e n_a dove n_d è 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	512	512
width	32	32
steps	7	5
learning_rate	0.02	0.02
max_epochs	70	150

Con un andamento degli iperparametri e del Mean Test Score come segue:

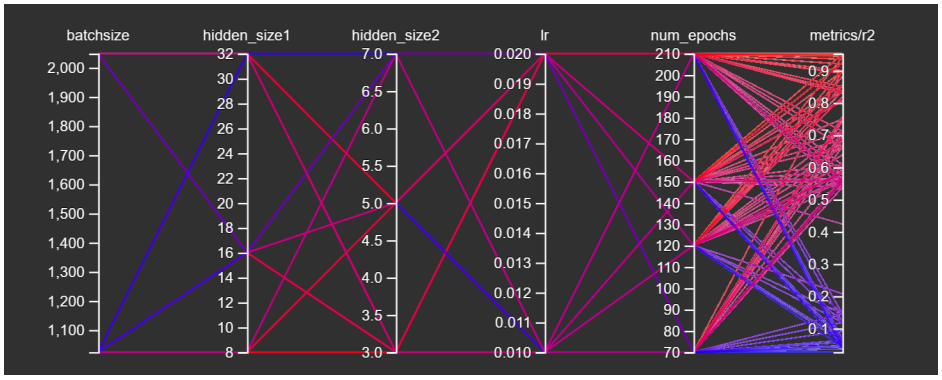


Figura 3.2: Andamento dei parametri e del Mean Test Score per i Tabular Data (senza PCA)

Capitolo 4

Risultati

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

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) è una media delle differenze al quadrato tra i valori predetti e quelli reali.
- RMSE: Root Mean Squared Error
  Root Mean Squared Error (RMSE) è la radice quadrata della media delle differenze al quadrato tra i valori predetti e quelli reali.
- R2: Coefficient of Determination
  Coefficient of Determination (R2) è 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	MAE
Linear Regression	0.005277524569837709	0.0726465730082136	0.05598534700766051
Lasso	0.005486814781830245	0.07407303680712872	0.0568421300459931
Ridge	0.005173317239238808	0.0719257759029321	0.05530627637007967
K Neighbors	0.04014675063926314	0.20036654071791313	0.15758359542154865
SVR	0.004249807698698538	0.06519054915168715	0.049401864388909957
Decision Tree R.	0.02518975360339929	0.15871280226685966	0.12227319185366432
Random Forest R.	0.012127163985177493	0.11012340343985694	0.0845742608309641

Model	R2
Linear Regression	0.9779884023114186
Lasso	0.9771154908004298
Ridge	0.9784230321851388
K Neighbors	0.8325551853181739
SVR	0.9822748229629815
Decision Tree R.	0.8949381068993166
Random Forest R.	0.9494197987687645

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

Model	MSE	RMSE	MAE
Linear Regression	0.006377020549959092	0.07985624928556996	0.06110574164990556
Lasso	0.009433868123483865	0.09712810161577269	0.07491622588172611
Ridge	0.006330311784378231 6	0.07956325649681661	0.060759651674386066
K Neighbors	0.03922817004507678	0.1980610260628698	0.15566020188713492
SVR	0.004886523894752017	0.06990367583147554	0.05318109432495814
Decision Tree R.	0.060707019543795586	0.24638794520794963	0.19116443494466778
Random Forest R.	0.012127163985177493	0.11012340343985694	0.0845742608309641

Model	R2
Linear Regression	0.973402604016331
Lasso	0.9606530472699164
Ridge	0.9735974178050478
K Neighbors	0.836386418354838
SVR	0.9796191952034381
Decision Tree R.	0.7468020331524539
Random Forest R.	0.9494197987687645

Miglior Classic ML

Model	C	Epsilon	Gamma	Kernel
SVR	1	0.01	0.01	rbf

Miglior Classic ML con PCA

Model	C	Epsilon	Gamma	Kernel
SVR	100	0.01	0.001	rbf

4.3 Neural Network

Neural Network Results

Model	MSE	RMSE	MAE
NN	0.00664011668413877	0.0814869105815887	0.0629449188709259
NN With PCA	0.005938166752457619	0.07705949991941452	0.05929824709892273

Model	R2
NN	0.9796283841133118
NN With PCA	0.9753079414367676

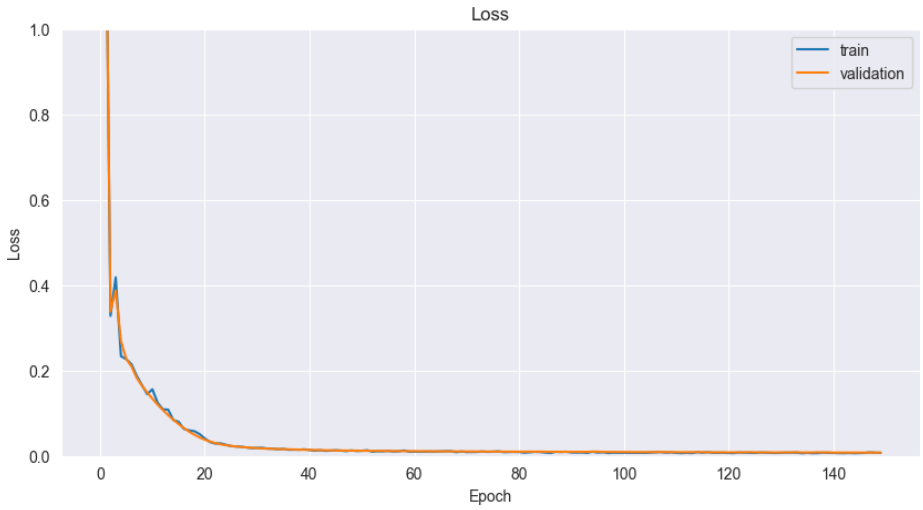
Miglior NN

Batch Size	Input Layer	Output Layer	lr	Epochs
1024	64	256	0.01	400

Miglior NN con PCA

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

Di seguito è riportato un grafico contenente l'andamento della funzione loss del training e validation del miglior modello di Neural Network



4.4 TabNet

TabNet Results

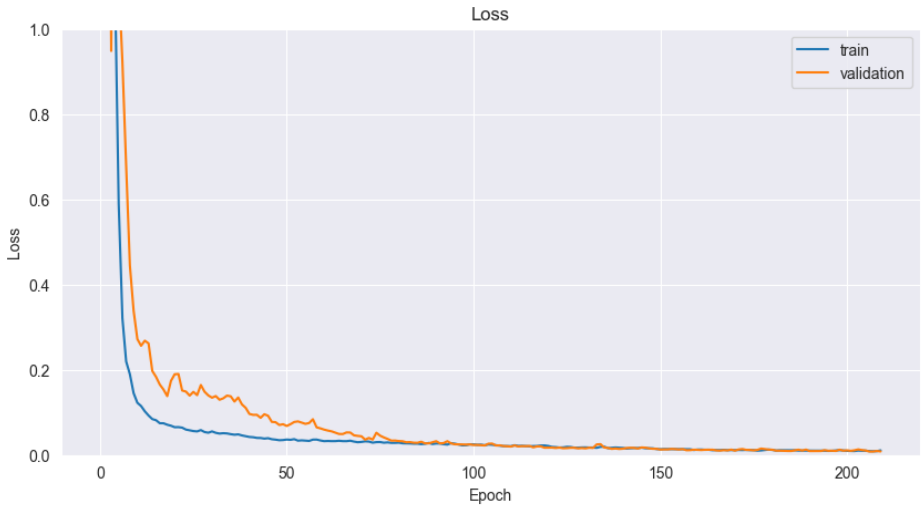
Model	MSE	RMSE	MAE
TabNet	0.00795749657941388	0.0892048013248944	0.0689039959039718
TabNet With PCA	0.00813222261181444	0.0901788368289059	0.964001783993923

Model	R2
TabNet	0.964775228814175
TabNet With PCA	0.964001783993923

Miglior TabNet

Batch Size	width	step	lr	Max Epochs
2048	8	5	0.02	210
512 with PCA	32	5	0.02	150

Di seguito è riportato un grafico contenente l'andamento della funzione loss del training e validation del miglior modello di TabNet



4.5 Conclusioni

Nel presente lavoro si è eseguita la pipeline descritta nell'Introduzione, in seguito ad aver acquisito il dataset abbiamo addestrato i modelli sia con dati grezzi sia applicando la PCA confrontando i risultati. Abbiamo notato che in tutti i modelli i risultati con PCA (n_components = 0.95) hanno portato a una perdita di informazioni e non un risparmio in termini di tempo sostanziale. Per tutti i modelli di Machine Learning è stato fissato il Random State per rendere possibile la riproducibilità dei risultati. Il tuning degli iperparametri è stato effettuato con il metodo GridSearchCV con 3-fold cross validation per ottimizzare proprio il tempo ed avere un risultato congruo in termini di Mean Test Score tra tutti i modelli di Machine Learning. Il modello dei Tabular Data è risultato molto più efficiente in termini di tempo rispetto alle Neural Network nonostante teniamo sempre in considerazione che i parametri (e di conseguenza i risultati) sono differenti. Possiamo concludere che tutti i modelli (eccezzione fatta per KNN e DT) riescono a generalizzare bene sui dati in modo da fare previsioni accurate su dati mai visti in precedenza.

Bibliografia

[1] pytorch-tabnet
https://pypi.org/project/pytorch-tabnet/

[2] TabNet: Attentive Interpretable Tabular Learning
https://arxiv.org/abs/1908.07442/

[3] TMDB
https://www.themoviedb.org/

[4] IMDB
https://www.imdb.com/

[5] MovieLens
https://grouplens.org/datasets/movielens/