Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Progetto Finale

New York City Taxi Trip Duration

Autori:
Apa Marco - 848154 - m.apa1@campus.unimib.it
Cavenati Laura - 864000 - l.cavenati1@campus.unimib.it
De Cola Francesca - 819343 - f.decola1@campus.unimib.it

8 febbraio 2021

Indice

Sommario

Il presente report si pone l'obiettivo di descrivere l'approccio metodologico al fine di risolvere la sfida pubblicata su Kaggle[1] denominata New York City Taxi Trip Duration. L'obiettivo di questa sfida è prevedere la durata di un viaggio in taxi nella città di New York partendo da una serie di informazioni rilasciate dalla NYC Taxi and Limousine Commission. Tra le caratteristiche a disposizione sono presenti le coordinate geografiche di partenza e arrivo, l'orario di prelievo ed il numero di passeggeri.
Si precisa che per tale fine sono stati implementati più algoritmi di machine learning e deep learning nel linguaggio di programmazione *Python*[2], utilizzando soprattuto la libreria scikit learn[3], che hanno restituito dei risultati poi confrontati nella parte finale del presente report in termini di MSE.

Keywords— Reti neurali, Machine Learning, Predizione durata viaggio, Ottimizzazione, Deep Learning

1 Introduzione

Riuscire a prevedere la durata di un viaggio all'interno di città densamente popolate è molto importante e può essere di aiuto sia al cliente, che al momento della prenotazione può sapere quando arriverà a destinazione, sia per gli odierni strumenti di prenotazione che possono così gestirle in maniera ottimale e distribuirle ai tassisti in maniera efficiente.
Il problema appena presentato è abbastanza complesso da descrivere e analizzare a causa di tutte le variabili che si presentano durante uno spostamento in auto in una grande e popolosa città tra cui traffico, semafori, attraversamenti pedonali, incidenti e così via.
La metodologia seguita inizia con l'analisi delle variabili esplicative e della variabile target (durata del viaggio). Successivamente si costruiscono delle reti neurali, particolarmente indicate per via della numerosità del dataset. Infine vengono ottimizzati gli iperparametri e si confrontano i risultati con quelli ottenuti con altre tipologie di modelli.

2 Dataset

Il Dataset in analisi è ottenibile dalla pagina della competizione[1] e risulta già diviso in training set e test set. Il training set contiene 1458644 osservazioni, il test set ne contiene 625134 ovvero il 30% delle osservazioni totali.
Le variabili presenti nel training set sono le seguenti:

- *id*: id univoco per ogni viaggio;
- *vendor_id*: codice univoco che identifica il provider del viaggio;
- *pickup_datetime*: data e ora di quando è stato fatto partire il contatore temporale della corsa;
- *dropoff_datetime*: data e ora di quando è stato fermato il contatore temporale della corsa;
- *passenger_count*: numero di passeggeri inserito dal tassista;
- *pickup_longitude*: longitudine di quando è stato fatto partire il contatore temporale della corsa;
- *pickup_latitude*: latitudine di quando è stato fatto partire il contatore temporale della corsa;
- *dropoff_longitude*: longitudine di quando è stato fermato il contatore temporale della corsa;
- *dropoff_latitude*: latitudine di quando è stato fermato il contatore temporale della corsa;
- *store_and_fwd_flag*: flag che indica se le informazioni sono state inviate al provider in un momento successivo alla fine della corsa e quindi prima salvate in memoria (per mancanza di connessione) (Y) o inviate direttamente senza salvataggio in memoria (N);
- *trip_duration*: durata della corsa in secondi.

Sono state applicate diverse tecniche di analisi per ottenere informazioni sui dati e per determinare come le diverse variabili dipendono dalla durata della corsa (*trip_duration*), ovvero la variabile target.

2.1 Analisi datatset e preprocessing

Eseguite le prime analisi è possibile dire che non sono presenti valori mancanti, non sono presenti record duplicati e che tutte le corse sono valide, ovvero che la data e ora di inizio della corsa è sempre minore della data e ora di fine della stessa. Il passo successivo è stato quello di analizzare le variabili a disposizione.

2.1.1 Durata della corsa

La variaible target *trip_duration* presenta una forte asimmetria a destra e quindi degli outliers come è possibile notare dalla figura 1. Per questo motivo è stato scelto di rimuovere dal dataset lo 0.3% dei viaggi con maggior durata e tutti i viaggi con durata inferiore ai 30 secondi.

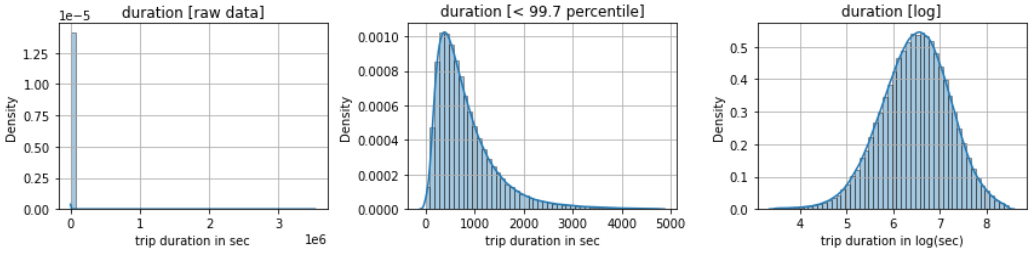

Figura 1: Statistiche della variabile trip_duration

Sempre dalla figura 1 si può notare che, applicando una trasformazione logaritmica alla variabile target, questa presenta una distribuzione simile alla normale.

2.1.2 Dati geospaziali

Analizzando i dati geospaziali (*pickup_latitude*, *dropoff_latitude*, *pickup_longitude* e *dropoff_longitude*), dopo aver rimosso gli outliers, si è deciso di utilizzare l'algoritmo di clustering KMeans[4] per raggruppare in 8 diversi cluster le zone di pickup e di dropoff relative ai viaggi.
Dalla figura 2 è possibile visualizzare i vari cluster geografici.

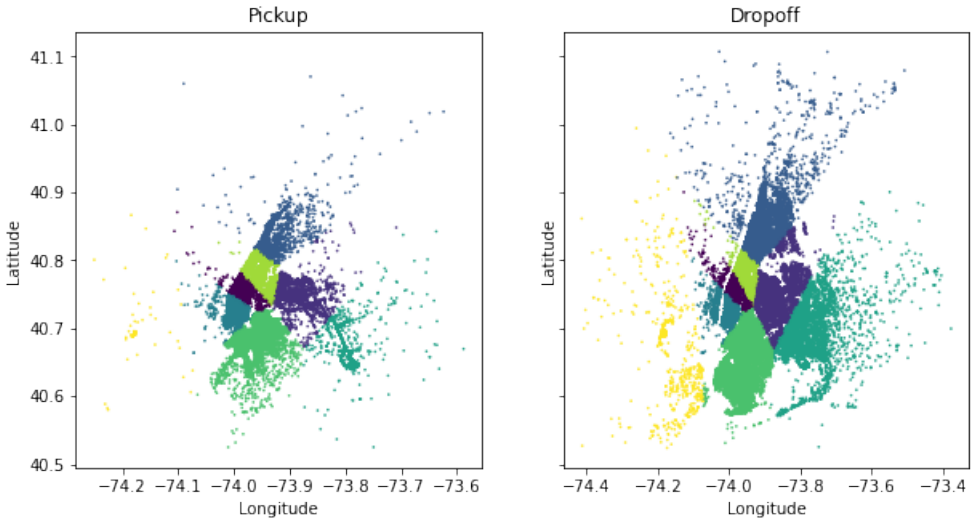

Figura 2: Cluster zone di pickup a sinistra e dropoff a destra

Distanza

Una variabile utile alla predizione è la distanza percorsa durante il viaggio. Successivamente alla rimozione delle corse che hanno lo stesso punto di partenza e di arrivo, probabilmente associate ai giri turistici della città, si è proceduto al calcolo della distanza percorsa durante il viaggio. A partire dalle variabili *pickup_latitude*, *dropoff_latitude*, *pickup_longitude* e *dropoff_longitude* è stata calcolata la distanza tramite la metrica dei taxi, o distanza di Manhattan[5], che misura la distanza tra due punti come somma del valore assoluto delle differenze delle loro coordinate. La nuova variabile è utilizzata per rimuovere i viaggi con distanza di Manhattan maggiore agli 80 km.
Facendo uso della nozione di distanza sopra citata viene creata una nuova variabile categorica (*distance_group*) come segue:

- Categoria *short* per distanza di Manhattan minore di 3 km;
- Categoria *medium* per distanza di Manhattan compresa tra 3 km e 10 km;
- Categoria *medium-long* per distanza di Manhattan compresa tra 10 km e 30 km;
- Categoria *long* per distanza di Manhattan maggiore di 30 km.

2.1.3 Dati temporali

Analizzando i dati temporali è possibile notare come il dataset contenga osservazioni comprese tra il 01/01/2016 e il 30/06/2016.
Oltre ad utilizzare la variabile con la granularità fornita (minuto), viene utilizzato l'algoritmo di clustering denominato KMeans[4] per raggruppare in 6 diversi cluster la variabile *pickup_datetime*. Dalla figura 3 è possibile visualizzare la distribuzione degli orari all'interno dei 6 gruppi.

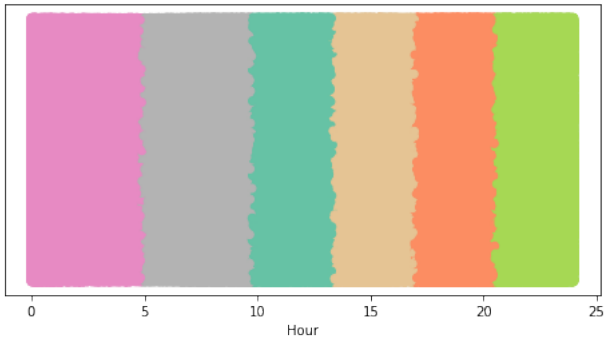

Figura 3: Cluster orari

Ultima aggiunta che concerne i dati temporali si riferisce alla definizione di una nuova variabile che assegna al giorno della settimana le seguenti etichette:

- *Weekend*: se il giorno fa parte del weekend;
- *Businessday*: se il giorno è lavorativo;
- *Holiday*: se il giorno non è lavorativo.

2.1.4 Dati meteorologici

Dall'analisi della variabile *pickup_datetime*, come è possibile notare nel grafico superiore della figura 4, è presente un valore anomalo nel numero di viaggi in corrispondenza del giorno 23/01/2016.

Per spiegare tale valore si è deciso di integrare i dati meteorologici riferiti allo stesso periodo reperiti su Kaggle[6].
Analizzando la quantità di neve precipitata, grafico inferiore dalla figura 4, si nota come nella giornata del 23 gennaio del 2016 ci sia stata una forte nevicata a New York.

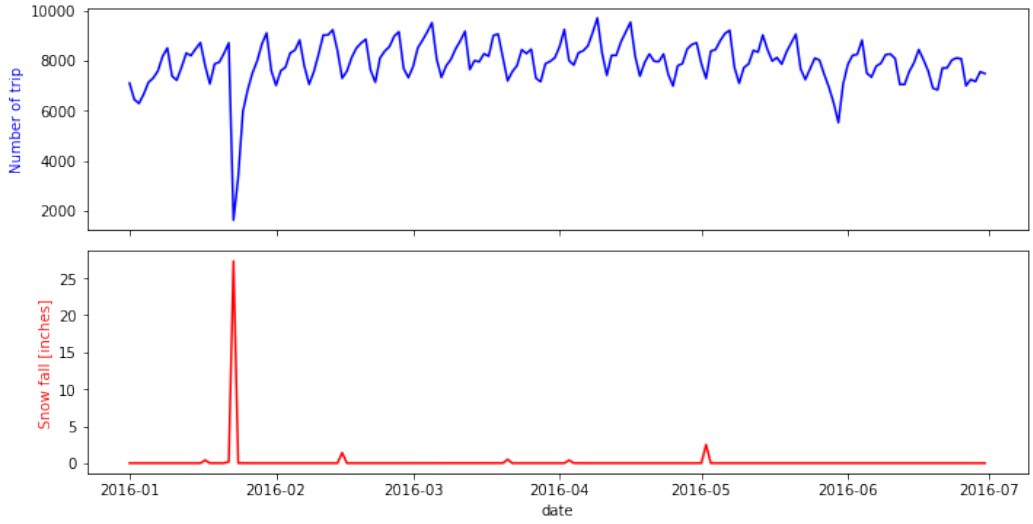

Figura 4: In alto il numero di corse per giorno, in basso la quantità di neve caduta

Per questo motivo è stato deciso di integrare nel dataset variabili che indicano la temperatura media, le precipitazioni, la quantità di neve caduta e quella accumulata.

2.1.5 Ulteriori considerazioni

- La variabile *vendor_id* presenta due categorie ed è bilanciata;
- La variabile *store_and_fwd_flag* viene rimossa per via della non influenza con la variabile target;
- Per quanto riguarda la variabile *passenger_count*, è stato deciso di considerare i viaggi con numero di persone compreso tra 1 e 6, eliminando le casistiche con numero di passeggeri maggiori o uguali a 7 e gli errori, ovvero le casistiche con 0 passeggeri.

Ultima considerazione viene fatta riguardo la velocità di percorrenza rispetto alla durata del viaggio. Come si può vedere dalla figura 5, sono presenti dei viaggi con velocità (calcolata a partire dalla distanza di Manhattan e dalla durata del viaggio) maggiore di 200 km/h. Questi sono considerati outliers ed eliminati dal dataset.

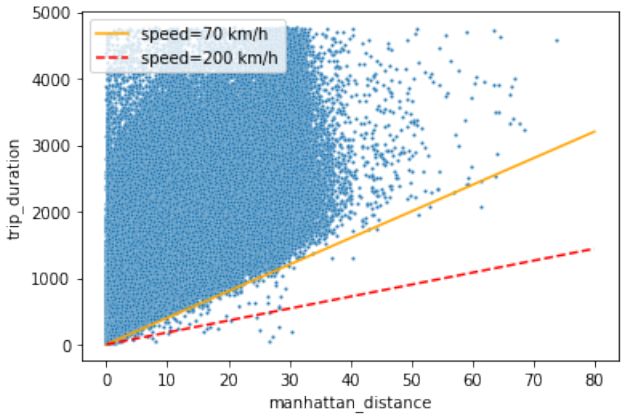

Figura 5: Distanza di Manhattan rispetto alla durata del viaggio

3 Approccio metodologico

Si è applicata una trasformazione logaritmica alla variabile target, così da ottenere una distribuzione riconducibile ad una normale. Successivamente si è diviso il dataset in train e validation (30% delle osservazioni).

3.1 Modelli di regressione

Prima di procedere alla costruzione delle reti neurali, si è provato ad utilizzare un set di diversi modelli: regressione lineare, albero decisionale, AdaBoost, XGB, LGBM, CatBoost. Nessuno di questi ha portato a risultati soddisfacenti e per questo non sono stati trattati.
Si menziona solamente l'importanza delle features che viene rilevata dal modello CatBoost, visibile in figura 6.

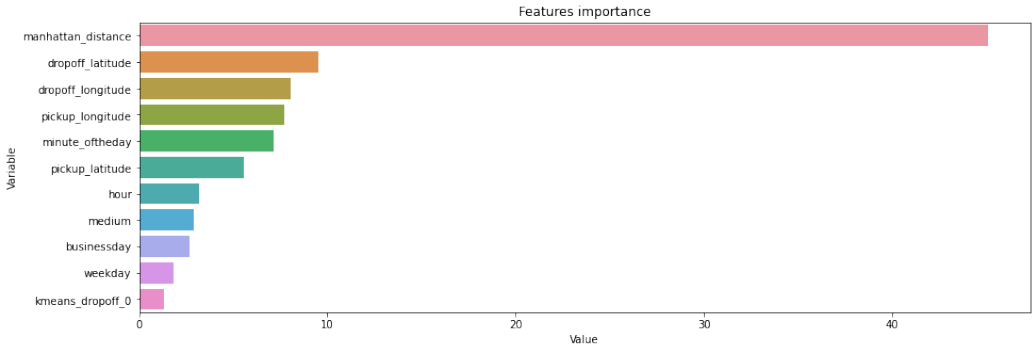

Figura 6: Importanza delle feature - CatBoost

3.2 Rete neurale

Non avendo ottenuto risultati ottimali dai modelli sopra citati, tramite l'utilizzo della libreria keras[7] sono state costruite tre architetture di reti neurali con diversa profondità. Sono stati infine confrontati i risultati ottenuti, si è scelto il modello migliore e si sono ottimizzati gli iperparametri.
Le reti neurali costruite sono le seguenti:

- Modello 1: 3 layer dense con rispettivamente 1024, 512, 1 nodi;
- Modello 2: 6 layer dense con rispettivamente 1024, 512, 256, 128, 64, 1 nodi;
- Modello 3: 8 layer dense con rispettivamente 1024, 512, 256, 128, 64, 32, 16, 1 nodi.

3.2.1 Architetture modelli e comparazione

Per tutte le reti neurali si sono usate 20 epoche per la fase di training dato l'eccessivo tempo computazionale con valori superiori. Grazie a questa scelta non è servito utilizzare metodi di early stopping. E' stato scelto un batch size pari a 256 per sfruttare le capacità di elaborazione parallela delle GPU. Come ottimizzatore è stato usato l'algoritmo *Adam*[8], algoritmo di learning adattivo, come loss function l'MSE e come metriche di misura della performance l'MSE e il MAE.
Dai risultati ottenuti il modello migliore in termini di MSE calcolato nel validation set risulta essere quello con 6 layer, ovvero il modello 2, la cui architettura è riportata in figura 7.

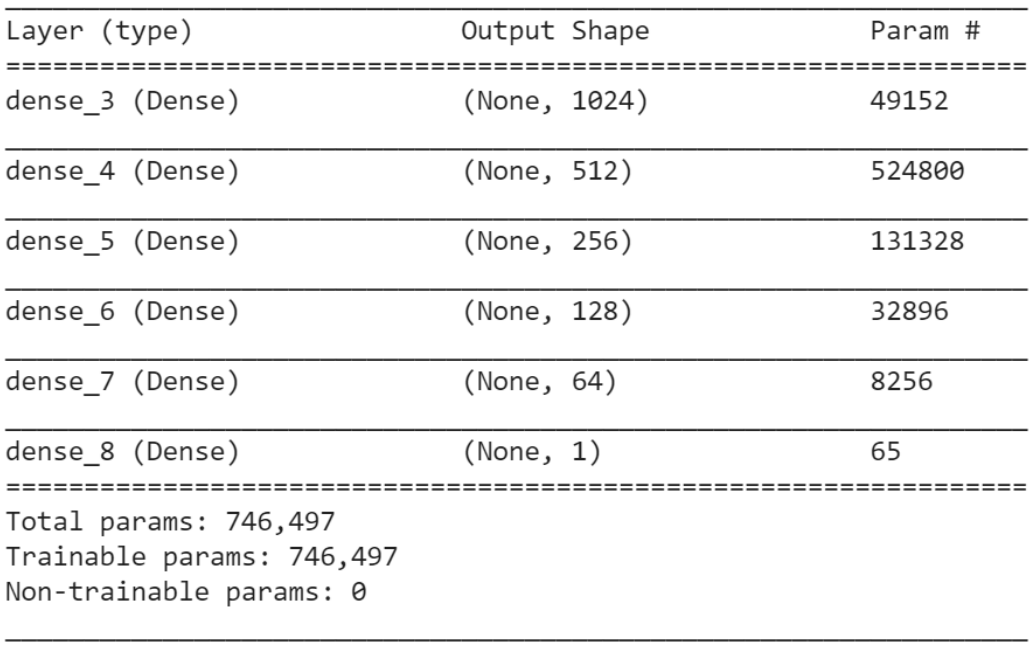

Figura 7: Architettura modello 2

3.3 Ottimizzazione

Successivamente alla scelta del modello 2, si è posto come obiettivo l'ottimizzazione degli iperparametri.

3.3.1 Funzione di attivazione

Sono state provate diverse funzioni di attivazioni: Relu, Elu e Gelu. Le performance migliori si ottengono con la Relu. Si è deciso quindi di utilizzarla in tutti i layer.

3.3.2 Regolarizzazione

Normalizzazione batch
Aggiungendo un layer di normalizzazione batch dopo ogni layer dense si standardizza la distribuzione degli output così da limitare lo spostamento della covarianza interna all'output dei layer ogni qual volta si verifica un cambiamento nella rete riuscendo così a generalizzare l'apprendimento[9]. In figura 8 è mostrata l'architettura del modello 2 con i layer di normalizzazione batch.

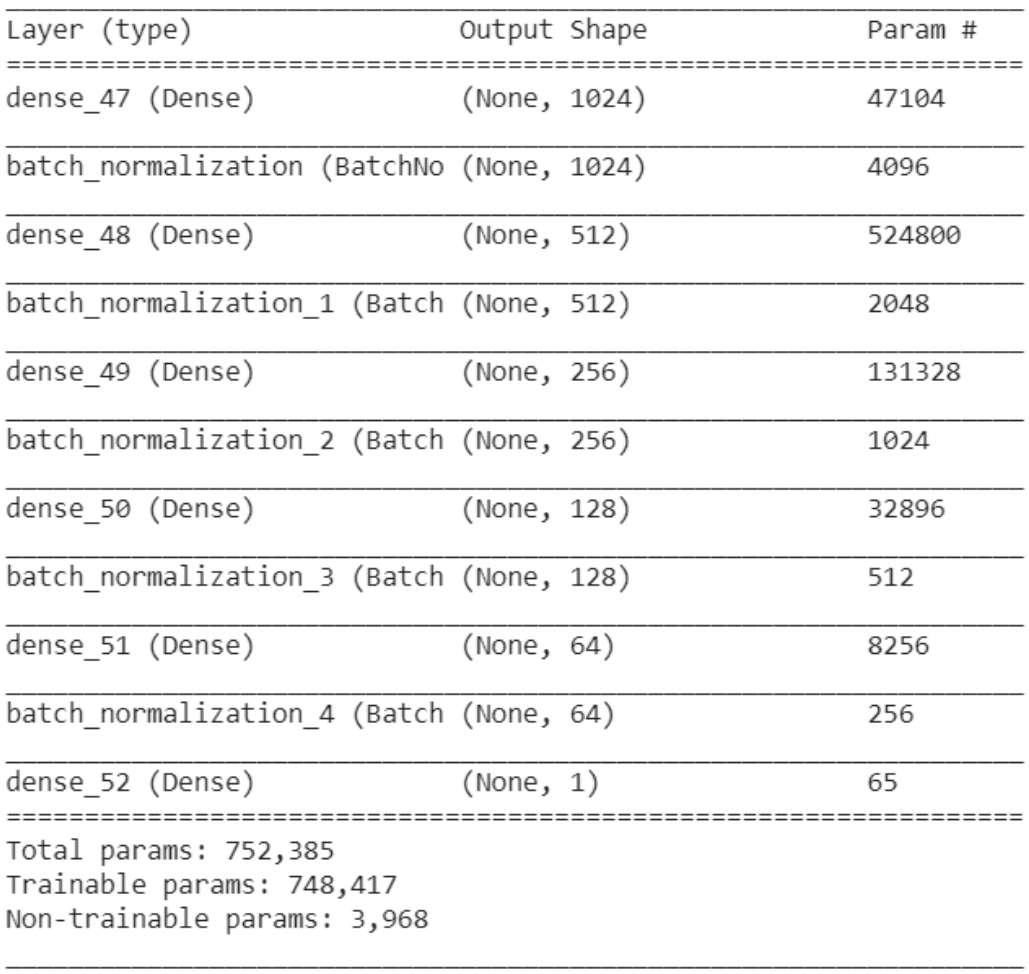

Figura 8: Architettura modello 2 + Normalizzazione batch

Rumore Gaussiano

Aggiungendo un rumore gaussiano con media pari a 0 e deviazione standard pari a 0.001 si impone una penalità sulla norma dei pesi. In questo caso si sono ottenuti risultati in termini di MSE leggermente migliori rispetto a quelli ottenuti col modello 2 ma non migliori dello stesso con l'aggiunta della normalizzazione batch.

Regolarizzazione L2 (Kernel-Bias)

Si è applicato ai pesi e al bias di ogni layer dense una regolarizzazione L2 andando così a penalizzare la norma dei parametri. I pesi sono stati inizializzati attraverso il Glorot uniform[10]. Ciò non ha portato a miglioramenti delle performance.

Dropout
Si è provato ad aggiungere un layer di dropout dopo ogni layer dense, ma ciò non ha portato a performance migliori.

4 Risultati e valutazioni

Per l'interpretazione dei risultati in termini di MSE sul validation set, è bene ricordare che la variabile target è stata trasformata attraverso una trasformazione logaritmica.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}(true_i - pred_i)^2 \tag{1}$$

E' possibile prendere nota dei risultati in tabella 1. Come già discusso nel paragrafo precedente, il valore di MSE calcolato sul validation set risulta migliore scegliendo il modello 3 (6 layer) a cui si aggiungono dei layer di normalizzazione batch dopo ogni layer dense.

Tabella 1: MSE per modello

Modello	**MSE**
Modello 1	0.1026
Modello 2	0.1005
Modello 3	0.1006
Modello 2 + Rumore gaussiano	0.1002
Modello 2 + Dropout	0.1035
Modello 2 + Normalizzazione batch	0.0965
Modello 2 + Regolarizzazione L2	0.1191

Riguardo il modello migliore, ovvero il modello 2 con l'aggiunta della normalizzazione batch, si può notare in figura 9, come il valore della loss sul validation set è stabile e piccolo già dopo le prime epoche.

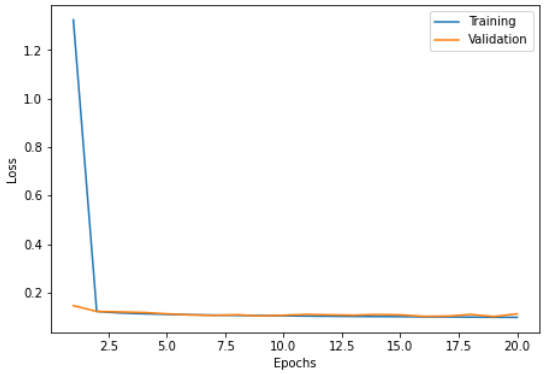

Figura 9: Valore di loss relativa al modello 2 + Normalizzazione batch

5 Discussione

Si è visto come i modelli citati nel paragrafo 3.1 non hanno portato a risultati ottimali; la motivazione è da ricondurre alla grossa mole di dati. Mentre le reti neurali si sono comportate abbastanza bene. Si nota come il modello 2 ottiene performance di poco migliori rispetto al modello con più layer, ovvero il modello 3. Come è stato presentato nel paragrafo 3.3, le varie ottimizzazioni provate non hanno portato a miglioramenti delle performance, a meno dell'aggiunta della normalizzazione batch. Molto probabilmente ciò è dovuto al fatto che questo tipo di layer porta ad una generalizzazione dell'output riuscendo a limitare lo spostamento della covarianza interna.

6 Conclusioni

Il lavoro svolto ha condotto a buoni risultati. La prima parte di analisi e preprocessing del dataset si è rivelata fondamentale per aumentare le performance dei modelli. In futuro, oltre a migliorare la fase di ottimizzazione, sarebbe opportuno sia provare architetture diverse, sia ad arricchire il dataset a disposizione con nuove variabili provenienti da dataset esterni. Esempi potrebbero essere informazioni relative alla velocità del veicolo, al traffico in tempo reale durante la corsa, ai semafori e alla reale distanza percorsa.

Riferimenti bibliografici

[1] "New york city taxi trip duration," https://www.kaggle.com/c/nyc-taxi-trip-duration.

[2] Wikipedia contributors, "Python (programming language) — Wikipedia, the free encyclopedia," 2021, [Online; accessed 6-February-2021]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1005152737

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[4] Wikipedia contributors, "K-means clustering — Wikipedia, the free encyclopedia," 2021, [Online; accessed 7-February-2021]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=K-means_clustering&oldid=1002047831

[5] ——, "Taxicab geometry — Wikipedia, the free encyclopedia," 2021, [Online; accessed 7-February-2021]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Taxicab_geometry&oldid=1004476192

[6] "Weather data in new york city - 2016," https://www.kaggle.com/mathijs/weather-data-in-new-york-city-2016?select=weather_data_nyc_centralpark_2016\%281\%29.csv.

[7] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[9] Cos'è la normalizzazione batch? [Online]. Available: https://ichi.pro/it/post/4811851625859

[10] Glorotuniform. [Online]. Available: https://www.tensorflow.org/api_docs/python/tf/keras/initializers/GlorotUniform