Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Progetto Finale

Taxi Trip Duration

Autori:
Valentina Nelli - 860613 - v.nellu@campus.unimib.it
Andrea Piancone - 812250 - a.piancone@campus.unimib.it
Marcello Pichini - 860855- m.pichini@campus.unimib.it

2020-2021

Sommario

Il seguente Report comprende un insieme di metodologie di Deep Learning, finalizzate alla previsione del tempo di percorrenza di un viaggio di un taxi. Dopo un'accurata preparazione dei dati a disposizione ed una procedura di selezione degli iperparametri, sono stati ottenuti dei modelli con buone performance e sufficientemente leggeri da poter funzionare in contesti dove le risorse sono limitate, aprendo le porte a potenziali sviluppi futuri per applicazioni mobile o integrate all'interno di un veicolo.

1 Introduzione

Obiettivo di questo elaborato è quello di realizzare dei modelli che siano in grado di prevedere il tempo di percorrenza di un viaggio in taxi, a partire da un insieme di variabili di input, tali da risultare efficaci ma al tempo stesso sufficientemente leggeri da poter essere impiegati in situazioni di risorse limitate.

Nello specifico, il focus dell'analisi è stato sui viaggi effettuati nella città di New York nel primo semestre del 2016. A partire dal **Dataset 2016 NYC Yellow Cab trip record data**, contenente i dati relativi a circa 1 milione e mezzo di dati e prendendo spunto dalla Challenge Kaggle **New York City Taxi Trip Duration**, sono stati allenati diversi modelli di Deep Learning.

Il lavoro si apre con l'attività di preprocessing, che ha visto diverse fasi. Durante una prima fase di feature engineering sono state create nuove features a partire da quelle esistenti e sono state eliminate quelle ritenute superflue ai fini dell'analisi. Successivamente, a causa della presenza di valori anomali tali da suggerire possibili errori strumentali derivanti dalla fase di rilevazione, si è deciso di eliminare una parte ridotta delle osservazioni. In ultima istanza si è proceduto ad una integrazione del dataset principale con i dati meteorologici della città di New York facenti riferimento all'anno 2016, al fine di includere ulteriori variabili in grado di spiegare la durata di un viaggio.

Sono state poi allenate tre diverse **Fully Connected Neural Networks**, ovvero una per ogni tecnica di regolarizzazione L1, L2 e Dropout. Questi modelli sono stati addestrati sia con iperparametri scelti discrezionalmente che con iperparametri scelti sulla base di un pipeline di ottimizzazione basata su un approccio *Random Search*, con lo scopo di migliorare l'efficacia e l'efficienza di questi modelli.

2 Datasets

2.1 Dataset New York Taxi

Il dataset sui cui è stata svolta l'analisi è disponibile su Kaggle[1]. Tale dataset è stato rilasciato da NYC Taxi and Limousine Commission e presenta la suddivisione training set e test set. Ogni record del dataset rappresenta un viaggio compiuto da un taxi nella città di New York. Le variabili di input del dataset sono le seguenti:

- **id**: identificatore univoco del viaggio;
- **vendor_id**: codice che identifica la compagnia del taxi o della limousine che ha compiuto il viaggio;
- **pickup_datetime**: data e ora di inizio del viaggio;
- **dropoff_datetime**: data e ora del momento in cui è finito il viaggio (non presente nel test set);
- **passenger_count**: numero di passeggeri presenti nel veicolo;
- **pickup_longitude**: longitudine del punto in cui è iniziato il viaggio;
- **pickup_latitude**: latitudine del punto in cui è iniziato il viaggio;
- **dropoff_longitude**: longitudine del punto in cui è finito il viaggio;
- **dropoff_latitude**: latitudine del punto in cui è finito il viaggio;
- **store_and_fwd_flag**: variabile binaria (Y/N) che indica se le informazioni relative al viaggio sono state inviate direttamente o meno a causa della assenza/presenza della connessione al server.

La variabile di output da predire è **trip_duration**, che indica la durata in secondi di un viaggio. Questa variabile nel test set non è disponibile.

[1] https://www.kaggle.com/c/nyc-taxi-trip-duration

2.2 Dati meteo

Data l'influenza delle condizioni meteo sulla viabilità e di conseguenza sui tempi di percorrenza, si è deciso di arricchire i dati disponibili con il dataset **weather-data-nyc-centralpark-2016** [2], anch'esso presente su Kaggle. Esso contiene una serie di variabili meteorologiche, relative alla città di New York, rilevate su base giornaliera nell'anno 2016. In particolare le variabili sono:

- **date**: data di rilevazione;
- **maximum temperature**: temperatura massima in gradi fahrenheit;
- **minimum temperature**: temperatura minima in gradi fahrenheit;
- **average temperature**: temperatura media in gradi fahrenheit;
- **precipitation**: quantità di pioggia caduta misurata in pollici;
- **snow fall**: quantità di neve caduta misurata in pollici;
- **snow depth**: profondità dello strato nevoso misurato in pollici;

Questo dataset ha richiesto una breve fase di preprocessing. Le variabili **precipitation**, **snow fall** e **snow depth**, presentavano il valore "T", per indicare tracce di questi fenomeni. Si è deciso di sostituire questo valore con lo zero. Il dataset è stato poi integrato tramite una operazione di inner join con il dataset principale, dove la chiave esterna di tale operazione è la *data*.

2.3 Preprocessing

Accertata l'assenza di valori mancanti nei dati, si è proceduto ad eliminare le variabili **id** e **vendor-id**, in quanto ritenute superflue per l'analisi predittiva. È inoltre stata eliminata la variabile **dropoff-datetime** in quanto ai fini di un modello predittivo non ha senso utilizzare come variabile di input l'esatto momento in cui il taxi giunge a destinazione. Infatti, se fosse noto a priori il modello predittivo sarebbe inutile.

Successivamente, si è deciso di sfruttare la presenza delle coordinate geografiche di partenza e destinazione per calcolare la distanza in chilometri

[2] https://www.kaggle.com/mathijs/weather-data-in-new-york-city-2016

compiuta durante il viaggio. In prima buttata si era pensato di ricorrere alle API di Google Maps, per ricavare una distanza che tenesse conto della conformazione urbana della città (sensi unici, svolte...) in maniera tale da ottenere una distanza estremamente realistica e adatta allo scopo. Tuttavia, dato che queste API sono a pagamento, si è deciso di ricorrere a misure di distanza che non sono in grado tenere in considerazione questi aspetti critici per la durata di un viaggio percorso in automobile. In particolare si è fatto ricorso alla distanza di Harvesine, definita dalla seguente formula:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (1)$$

dove φ_1 e φ_2 sono la latitudine del punto 1 del punto 2 in radianti, mentre λ_1 e λ_2 sono la longitudine del punto 1 e del punto 2 in radianti. Invece r è il raggio della Terra in chilometri. In questo modo si è ottenuta una nuova variabile chiamata **distance**. Un'altra variabile estratta grazie alla presenza delle coordinate geografiche di partenza e di arrivo è la **direction**, definita dalla seguente formula:

$$direction = \text{atan2}(\sin(\Delta\lambda)\cdot\cos\varphi_2, \cos\varphi_1\cdot\sin\varphi_2 - \sin\varphi_1\cdot cos\varphi_2\cdot cos(\Delta\lambda)) \quad (2)$$

dove φ_1,λ_1 sono le coordinate del punto di partenza e φ_2,λ_2 sono le coordinate del punto di arrivo, $\Delta\lambda$ è la differenza in longitudine.

Una volta calcolata la distanza in chilometri e la direzione in gradi, le variabili **pickup-longitude**, **pickup-latitude**, **dropoff-longitude**, **dropoff-latitude** sono state eliminate. Successivamente, a partire dall'attributo **pickup-datetime** sono state estratte una serie di variabili dummies per rappresentare l'ora in cui è iniziato il viaggio, il giorno della settimana, se il giorno era feriale o meno ed il mese, al fine di includere delle features in grado di fornire informazioni temporali potenzialmente utili per stimare la durata di un viaggio. Una volta ottenute queste variabili dummies, la variabile di input, **pickup-datetime** è stata eliminata. Infine si è binarizzata la variabile **store_and_fwd_flag**. Al termine di questa fase di feature engineering, si passa da 10 variabili di input a 48.

Successivamente, a seguito di un'analisi esplorativa, si è notata la presenza di valori anomali. Osservandoli è emerso che alcuni di essi suggerivano la

presenza di possibili errori strumentali nella rilevazione delle variabili **distance**, **trip_duration** e **passenger_count**, si è deciso dunque di eliminare i viaggi che presentavano alcune caratteristiche ritenute non ragionevoli:

- quelli con durata minore di 60 secondi e maggiore di 21600 secondi;
- quelli con zero passeggeri;
- quelli con una distanza minore di 100 metri e maggiore di 100 chilometri.

3 Approccio metodologico

3.1 Validation set approach

Come anticipato, nel test set disponibile su Kaggle non è presente la variabile di output. Dunque si è deciso di ignorare il test set e dividere il training set nel seguente modo: 60% training set, 10% validation set e 30% test set.

Successivamente si è proceduto a standardizzare le variabili numeriche, trasformandole in nuove variabili a media zero e varianza unitaria ed aventi la medesima scala di misurazione.

3.2 Modelli

Data la natura del problema e dei dati a disposizione si è deciso di ricorrere alle *Feed Forward Neural Netwroks*, in particolare si è fatto affidamento sulle *Fully Connected Neural Networks*, in quanto le osservazioni non presentano un aspetto sequenziale. Infatti i dati non solo non sono ordinati cronologicamente, ma numerosi timestamp si ripetono più volte. Pertanto, può essere ragionevole ritenere che gli input sono tra loro indipendenti.

Sono state addestrate tre diverse reti neurali, utilizzando per ognuna di esse un differente metodo di regolarizzazione per evitare il fenomeno dell'overfitting, al fine di individuare il metodo che porta alle migliori prestazioni. I metodi di regolarizzazione impiegati sono dropout, regolarizzazione L2 e L1. I modelli sono stati ottimizzati con il gradiente stocastico discendente con momentum, fissando un learning rate costante pari a 0.002 ed un momentum pari a 0.3. La funzione di perdita minimizzata è il *Mean Squared Error* (MSE), questa funzione è stata scelta in quanto il problema è di regressione

e sulla base di essa sono state valutate le performance dei modelli. I pesi delle reti neurali sono stati inizializzati con il metodo *GlorotUniform*. Le reti neurali sono state addestrate per 100 epoche, adottando un callback di early stopping che interrompe l'addestramento, qualora la perdita sul validation set non migliori dopo 10 epoche. La dimensione del batch è stata fissata pari a 128, in maniera tale che sia abbastanza grande da avere un andamento più smussato della curva di Loss. I modelli sono stati realizzati sfruttando le librerie Keras e Tensorflow.

Inizialmente, i tre modelli sono stati addestrati con i seguenti iperparametri scelti discrezionalmente: quattro hidden layers, dove il numero di neuroni per ciascun hidden layer è rispettivamente pari a 256 per il primo, 128 per il secondo ed il terzo e 32 per il quarto. La funzione di attivazione impiegata negli hidden layers è la *ReLu*, al fine di evitare il problema del vanishing gradient. L'output layer ha un solo neurone e la funzione di attivazione è quella lineare. Per i modelli regolarizzati rispettivamente con la penalizzazione L1 e L2, si è impiegato un parametro lambda pari a 0.0005, mentre per il rate dei layers di dropout si è fissato un valore pari a 0.2.

Successivamente, al fine di migliorare le prestazioni dei tre modelli si è deciso di effettuare un tuning dei loro iperparametri. La selezione degli iperparametri è stata realizzata sfruttando la libreria Optuna[3], per due motivi: il primo è che consente di implementare un *Pruning* dei tentativi in base al valore della funzione di perdita sul validation set, troncando quelli che non sono promettenti consentendo un notevole risparmio di tempo nella procedura di ottimizzazione. Il secondo invece consiste nella possibilità di salvare il risultato di ogni tentativo all'interno di un database SQLite in maniera tale da poter interrompere la procedura e riprenderla in un momento successivo e allo stesso tempo avere un backup della procedura di ottimizzazione. Come *Sampler*, ovverosia il campionatore del set dei parametri di ricerca, si è deciso di ricorrere al *Random Search*, con 50 tentativi, scegliendo infine la combinazione di iperparametri che ha portato al minor MSE sul validation set dopo 30 epoche. In particolare gli iperparametri ottimizzati sono il numero di layers, il numero di neuroni per layer, il valore del parametro lambda per i modelli regolarizzati con la penalizzazione L1 e L2 e ed il rate dei layers di dropout per il modello regolarizzato con la omonima tecnica. Il learning rate, il momentum ed il batch size non sono stati oggetto di ottimizzazione. Anche in questo caso si è utilizzata la *ReLu* come funzione di attivazione negli

[3]https://optuna.org/

hidden layers. Lo spazio di ricerca degli iperparametri è stato così definito:

- numero di hidden layers: tre o quattro;
- numero di neuroni nel primo layer: range di interi $[32, 256]$ con passo 16;
- numero di neuroni nel secondo layer: range di interi $[32, 128]$ con passo 16;
- numero di neuroni nel terzo layer: range di interi $[32, 128]$ con passo 16;
- numero di neuroni nel quarto layer: range di interi $[16, 64]$ con passo 16;
- parametro lambda per la penalizzazione L1 e L2: intervallo continuo $(0.0001; 0.1]$;
- rate per i layers di dropout: range $[0.1, 0.5]$ con passo 0.1.

In Tabella 1 è riportata, per ciascun modello, la combinazione ottimale di iperparametri che ha prodotto il minor MSE sul validation set.

Tabella 1: Iperparametri ottimali per modello

Iperparemetri	**L1 Regularization**	**L2 Regularization**	**Dropout**
N. di hidden layers	3	3	3
N. di neuroni nel primo layer	64	144	240
N. di neuroni nel secondo layer	112	112	112
N. di neuroni nel terzo layer	64	48	128
N. di neuroni nel quarto layer	NA	NA	NA
Lambda	0.00043	0.0001	NA
Probabilità dropout	NA	NA	0.1

Dall'osservazione della Tabella 1 si nota che la procedura di ottimizzazione ha selezionato 3 come numero di hidden layers ideale per tutti i modelli, a differenza dei modelli non ottimizzati, i quali erano costituiti da 4 hidden layers. Per quanto riguarda i modelli regolarizzati con la penalizzazione L1 e L2, l'ottimizzazione ha portato ad una minore intensità della regolarizzazione, specie per la L2 penalty. Una volta trovati gli iperparametri ottimali,

per ciascuno dei tre modelli si è proceduto a riaddestrare le reti neurali con gli iperparametri ottimali, per 100 epoche con l'early stopping, il batch size e l'ottimizzatore descritto in precedenza.

4 Risultati e valutazione

Nella Figura 1 sono mostrati i risultati della procedura di ottimizzazione degli iperparametri, per i tre modelli escludendo il primo tentativo del modello regolarizzato con la tecnica dropout, a causa di un MSE molto elevato che rendeva complessa la lettura della figura.

Nella Tabella 2 sono riportate le performance, misurate in termini di MSE, registrate sul training set, validation set e test set, sia dei modelli i cui iperparametri sono stati oggetto di ottimizzazione che dei dei modelli con gli iperparametri scelti discrezionalmente.

Tabella 2: MSE sul training, validation, test per modello

	Modelli non ottimizzati			**Modelli ottimizzati**		
	Train	**Validation**	**Test**	**Train**	**Validation**	**Test**
L1 regularization	0.2858	0.2702	0.2814	0.2853	0.2701	0.2808
L2 regularization	0.2659	0.2541	0.2638	0.2598	0.2497	0.2592
Dropout	0.2682	0.2541	0.2649	0.2520	0.2396	0.2496

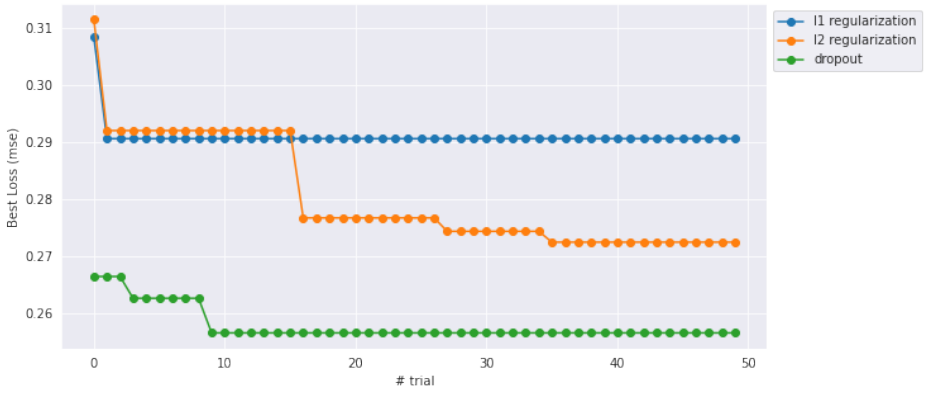

Figura 1: Risultati dell'ottimizzazione degli iperparametri

Nella Figura 2 sono riportate le curve di Loss dei modelli addestrati, confrontando la curva generata dal modello senza iperparametri ottimizzati che

la curva generata dal modello i cui iperparametri sono stati ottimizzati. Infine, nella Figura 3 si mostra il confronto tra il MSE dei modelli sul validation test e test set ed il numero di parametri addestrabili.

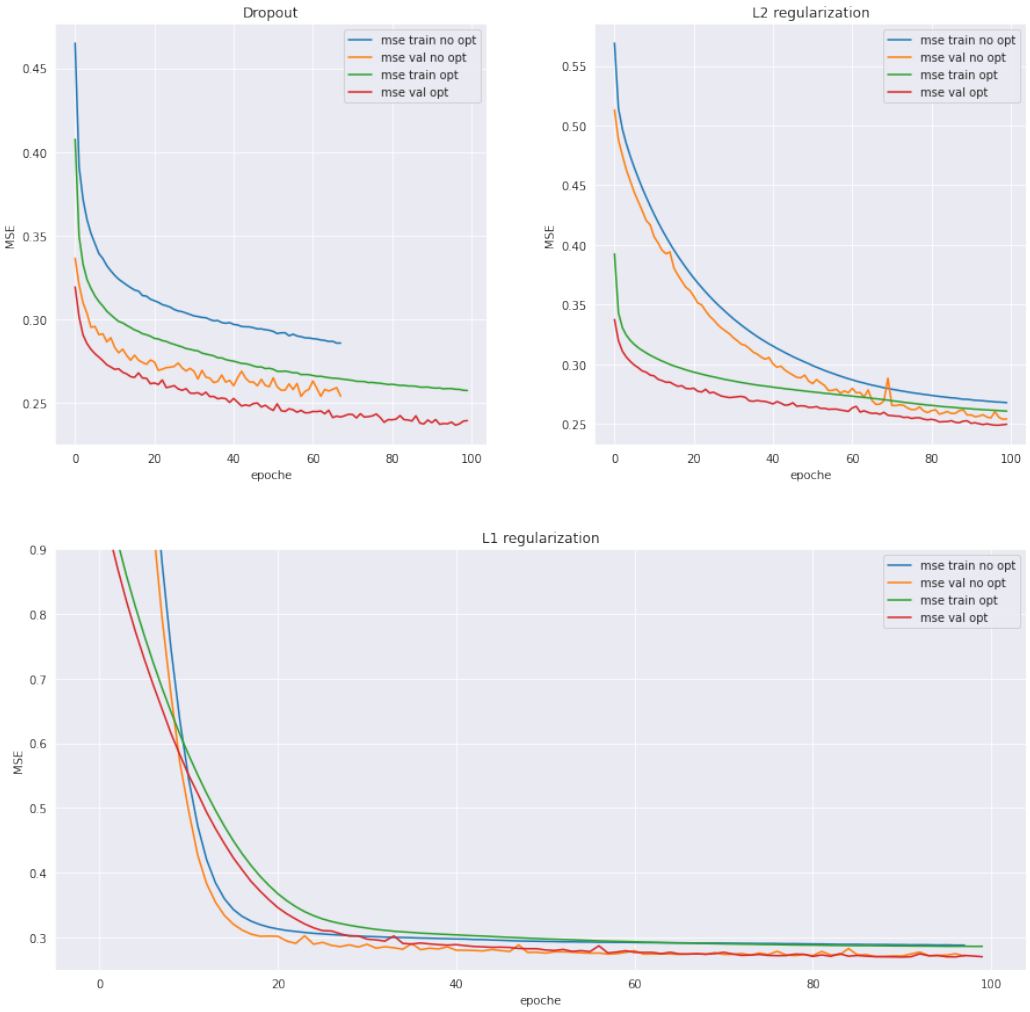

Figura 2: Curva di Loss dei modelli

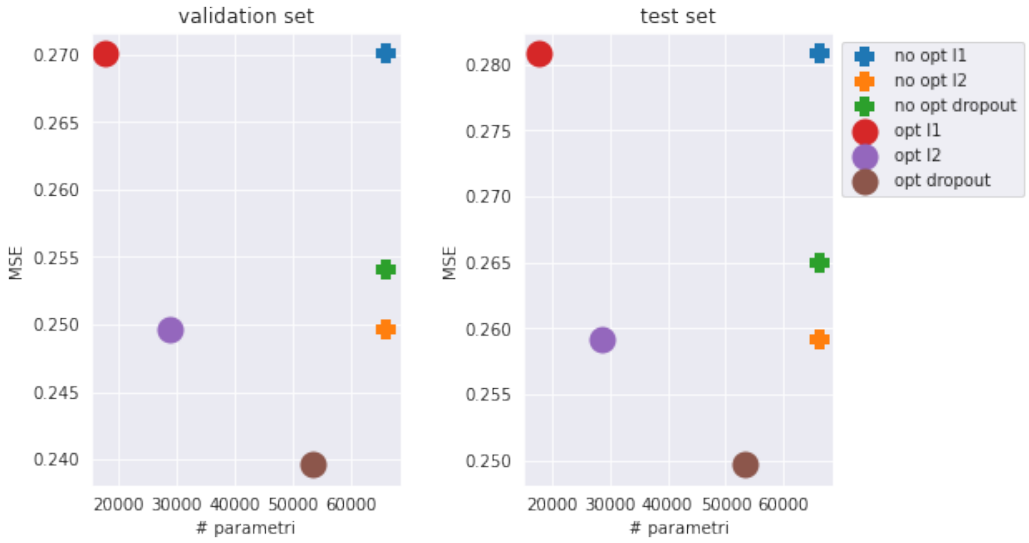

Figura 3: MSE vs numero di parametri

5 Discussione dei risultati

Come mostra la Tabella 2, nè i modelli con gli iperparametri non ottimizzati nè i modelli con iperparametri ottimizzati portano ad una situazione di overfitting, infatti il MSE sia sul validation set che sul test set è inferiore rispetto a quello riscontrato sul training set, mentre si nota che sistematicamente il MSE sul validation set è di poco inferiore rispetto a quello registrato sul test set. Tuttavia, si può osservare che, grazie alla procedura di ottimizzazione degli iperparametri, si è ottenuto un miglioramento delle performance di tutti e tre i modelli.

Effettuando un confronto tra modelli, emerge che tra quelli senza ottimizzazione degli iperparametri, il miglior modello trovato è quello con *L2 regularizer*, seguito da quello regolarizzato con il metodo dropout, mentre il peggiore è quello con *L1 regularizer*. Al contrario, nei modelli con gli iperaparametri ottimizzati la gerarchia muta leggermente: il modello con *L1 regularizer* rimane sempre il peggiore, mentre il modello con dropout sorpassa il modello regolarizzato con la L2 penalty.

Questo cambiamento della gerarchia, può essere spiegato dal fatto che, come mostra la Figura 2, l'addestramento del modello con dropout senza iperparametri ottimizzati, si interrompe prima delle 100 epoche a causa del-

l'early stopping. Pertanto la procedura di ottimizzazione degli iperparametri per il modello con dropout ha consentito anche di aumentare ulteriormente la sua capacità di generalizzazione.

Tuttavia, considerare solo le performance del modello rispetto alla funzione di perdita potrebbe non essere sufficiente, in quanto un modello potrebbe non essere abbastanza leggero da poter essere impiegato all'interno di una app che deve funzionare ad esempio su uno smartphone e quindi con risorse limitate. Pertanto, si è deciso di confrontare le performance dei modelli con il numero di parametri addestrabili. Questo confronto è riportato nella Figura 3. Dall'osservazione della Figura 3, emerge che in generale grazie all'ottimizzazione degli iperparametri, i modelli hanno raggiunto performance migliori sia sul test set che sul validation set, con un minor numero di parametri. Quindi grazie al tuning degli iperparametri, non solo i modelli trovati sono più efficaci ma sono anche più efficienti e parsimoniosi.

Il miglior modello è quello regolarizzato con dropout e iperpartametri ottimizzati (pallino marrone), tuttavia esso ha un numero di parametri elevato che si avvicina a quello dei modelli i cui iperparametri non sono stati ottimizzati. Dunque questo modello potrebbe essere considerato il migliore nel caso in cui le risorse computazionali non siano particolarmente limitate. Il modello con iperparametri ottimizzati e regolarizzato con la L2 penalty (pallino viola), invece presenta un miglior compromesso tra efficacia ed efficienza, infatti con un numero di parametri contenuto raggiunge buone performance e quindi potrebbe essere il miglior modello nel caso in cui debba funzionare in un contesto di risorse molto limitate.

6 Conclusioni

Il lavoro ha portato, dopo una attenta fase di preprocessing, a dei modelli aventi una buona capacità di generalizzazione e delle buone prestazioni, che sono state ulteriormente migliorate grazie all'attività di tuning degli iperparametri. La pipeline di ottimizzazione degli iperparametri ha portato ad un incremento sia dell'efficacia che dell'efficienza dei modelli. Questi risultati aprono le porte alla possibilità di replicare l'esperimento anche in altre città e la possibilità di utilizzare questi modelli, con le opportune modifiche, come l'utilizzo delle API di Google Maps per ricavare distanze più accurate. Inoltre, un ulteriore sviluppo futuro potrebbe essere quello di adottare tecniche di model compression per ridurre ulteriormente la dimensione dei

modelli, mantenendo buone performance, facilitando l'utilizzo di tali modelli in contesti caratterizzati da una scarsità di risorse.

Riferimenti bibliografici

[1] https://www.kaggle.com/c/nyc-taxi-trip-duration/overview.

[2] M. Cools, E. Moons, and G. Wets, "Assessing the impact of weather on traffic intensity," *Weather, Climate, and Society*, vol. 2, 01 2010.

[3] https://www.kaggle.com/mathijs/weather-data-in-new-york-city-2016.

[4] https://www.movable-type.co.uk/scripts/latlong.html.

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 07 2019, pp. 2623–2631.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.