Report per l'Esame di Fondameti di Machine Learning

Giovanni Scialla

132440
Corso di Laurea in Ingegneria informatica
255467@studenti.unimore.it

21/06/2021

Abstract

Il processo di noleggio biciclette è sicuramente correlato alle caratteristiche ambientali e stagionali. Per esempio il giorno della settimana, le condizioni atmosferiche, la stagione corrente, etc.
Questo elaborato pone come obbiettivo quello di analizzare il dataset relativo al noleggio bici di Washington D.C , USA negli anni 2011-2012 [1] e provare a fare delle stime circa il numero di biciclette noleggiate in una determinata giornata.

1 Introduzione

Il Bike Sharing System è un nuovo, innovativo processo di gestione noleggio biciclette in cui l'intero processo: noleggio, utilizzo e restituzione è del tutto automatico. Attraverso questi sistemi gli utenti, da qualsiasi posizione essi si trovino, possono tranquillamente noleggiare una bicicletta in modo semplice ed intuitivo. Al giorno d'oggi si è mostrato un grande interesse verso questi sistemi data la loro importanza nei problemi legati all'ambiente, al traffico e alla salute.
Il dataset che andremo ad analizzare proviene dal Capital Bikeshare system, Washington D.C., USA ed è pubblicamente reperibile sul sito http://capitalbikeshare.com/system-data.
L'obbiettivo di questo elaborato sarà quello di ottenere, tramite tecniche di Machine Learning e algoritmi di regressione, un modello che sia in grado , data una determinata giornata, con tutte le sue caratteristiche, di stimare al meglio il numero di biciclette possibilmente noleggiate in quel dato giorno.
Questo obbiettivo verrà raggiunto tramite:

Exploratory Data Analysis in cui verrà analizzato il dataset per intero, mettendo in evidenza eventuali caratteristiche lineari e correlazioni tra i dati eliminando eventuali valori mancanti o nulli e caratteristiche non rilevanti al fine di "pulire" quanto più possibile i dati. A tal proposito verrà utilizzata la libreria grafica di Seaborn [3]

Preprocessing dei dati in cui, il dataset, verrà suddiviso in tre parti: una per il training, una per la validazione e una per il testing. Inoltre si effettueranno modifiche di scaling, in particolare il dataset verrà normalizzato tramite Min Max Scaler e standardizzato tramite Standard Scaler.

Training dei dati e cross validation, in cui ai modelli utilizzati verranno forniti i dati di training per addestrarli e verranno scelti gli iperparametri tramite la funzione GridSearchCV di sklearn. I dati di training utilizzati verranno forniti secondo le tre varianti realizzate in preprocessing, quindi i modelli saranno addestrati su i dati originali, su dati normalizzati e su dati standardizzati.

Testing dei modelli, come ultimo passo verranno testati i modelli per determinare quale, e con quali caratteristiche, è quello che performa meglio per il compito assegnato. Si valuterà la "bontà" del modello utilizzando metriche quali:
Mean Squared Error
Mean Absolute Error
R^2-Score

2 Exploratory Data Analysis (EDA)

Inizio l'analisi leggendo il dataset e osservando le sue caratteristiche generali quali:
dimensione: df.shape = (731, 16) , in questo caso contiene 731 records, ciascuno con 15 features che posso analizzare leggendo la documentazione relativa al dataset.
con df.columns ottengo i nomi relativi alle colonne:

- 'instant' - indice di riga
- 'dteday' - data giornaliera
- 'yr' - anno corrente
- 'mnth' - mese corrente
- 'holiday' - se vale 1 è giorno di festa
- 'weekday' - giorno della settimana
- 'workingday' - se vale 1 è giornata lavorativa
- 'weathersit' - condizioni atmosferiche
- 'temp' - temperatura registrata in Celsius (normalizzata)
- 'atemp' - temperatura sentita dal cliente in Celsius (normalizzata)
- 'hum' - indice di umidità (normalizzata)
- 'windspeed' - velocità del vento
- 'casual' - numero di biciclette noleggiate da utenti non registrati nel sistema
- 'registered' - numero di biciclette noleggiate da utenti registrati nel sistema
- 'cnt' - numero totale di biciclette noleggiate

Con la funzione df.info() e df.describe() noto che i valori categorici sono già stati trasformati in valori numerici, non sono presenti valori nulli, ed effettivamente, i valori statistici quali umidità e temperatura risultano già normalizzati.

Column	**Non-null count**	**Dtype**
0 instant	731 non-null	int64
1 dteday	731 non-null	object
2 season	731 non-null	int64
3 yr	731 non-null	int64
4 mnth	731 non-null	int64
5 holiday	731 non-null	int64
6 weekday	731 non-null	int64
7 workingday	731 non-null	int64
8 weathersit	731 non-null	int64
9 temp	731 non-null	float64
10 atemp	731 non-null	float64
11 hum	731 non-null	float64
12 windspeed	731 non-null	float64
13 casual	731 non-null	int64
14 registered	731 non-null	int64
15 cnt	731 non-null	int64

Table 1: df.info()

Provo ed eliminare eventuali records duplicati, la dimensione non cambia, quindi non erano presenti duplicati
Posso eliminare subito alcune colonne quali l'indice di riga, la data e l'anno, siccome il mio obbiettivo è quello di realizzare un modello di regressione valido attualmente.
posso eliminare anche la colonna 'casual' e 'registered' in quanto la features obbiettivo sarà 'cnt' ovvero il numero totale di biciclette noleggiate. Fatto ciò, rileggo il dataset con solo le colonne rimanenti, rinominandole per procedere ad un analisi più dettagliata.
Incomincio dalle features di tipo numerico tramite una matrice di correlazione.

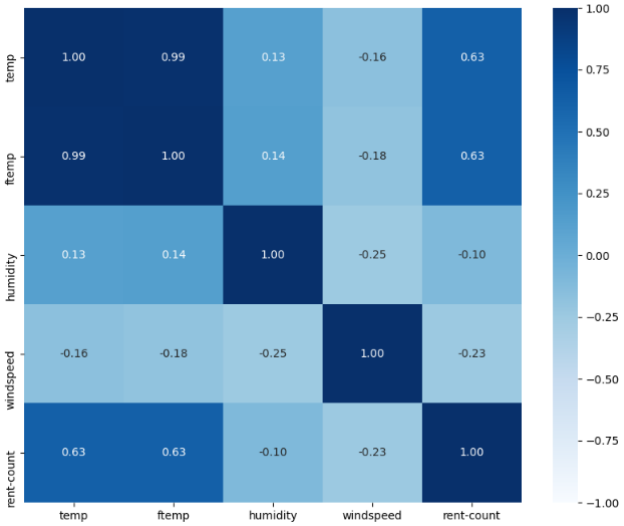

Figure 1: Matrice di correlazione

La matrice di correlazione ci dice quanto le rispettive features sono correlate tra di loro. In una scala, come nel nostro caso da $[-1, 1]$, più i valori di correlazione si avvicinano agli estremi dell'intervallo, e più le due features sono correlate fra loro.
Nel caso in esame, noto che le colonne riguardanti la temperatura registrata e la temperatura sentita dal cliente, sono ugualmente correlate con il numero di noleggio bici, questo è anche un caso di multicollinearità tra le due. Decido di eliminare la colonna riguardante la temperatura "sentita", mi tengo i dati per la temperatura registrata in quanto presenta un valore di correlazione molto alto di (0.63), significa che più la temperatura è alta e più si registrano noleggi di biciclette in giro per la città. Mentre, l'umidità e la presenza di vento, influiscono relativamente poco.
Ora passo in esame le variabili categoriche.

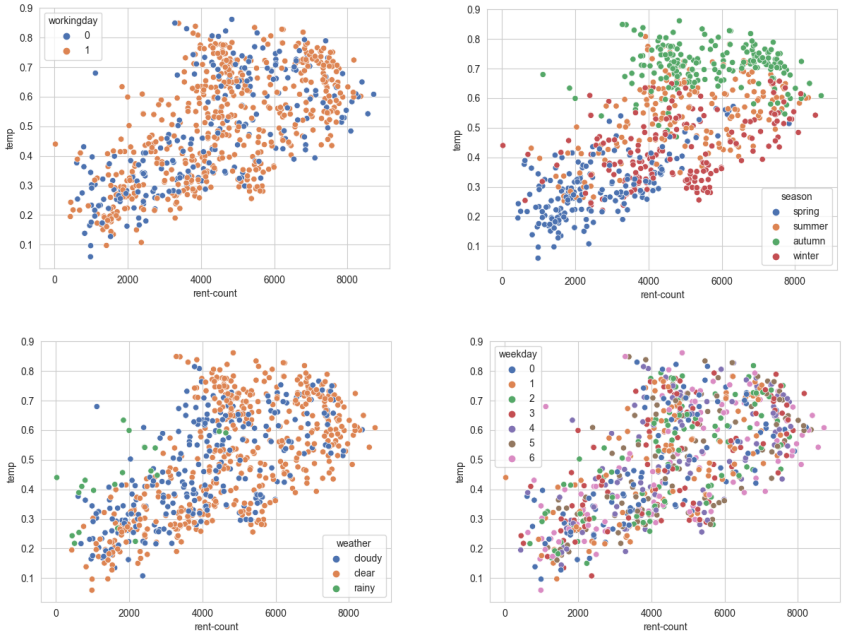

Figure 2: ScatterPlot

Inizio l'analisi delle variabili categoriche tramite degli scatterplot. Non c'è da meravigliarsi che le caratteristiche più notevoli, per il noleggio biciclette, siano la stagione e le condizioni atmosferiche, mentre per quanto riguarda le giornate lavorative e i giorni della settimana non si nota alcun pattern notevole. proseguo tramite dei box plot.
questi ultimi ci confermano le ipotesi fatte sopra, cioè che per le stagioni estate, autunno ed inverno si sono registrati più noleggi rispetto alla primavera. ovviamente la distribuzione di questi ultimi rispecchia quella dei mesi. I giorni della settimana risultano irrilevanti al fine di contare il numero di noleggi.

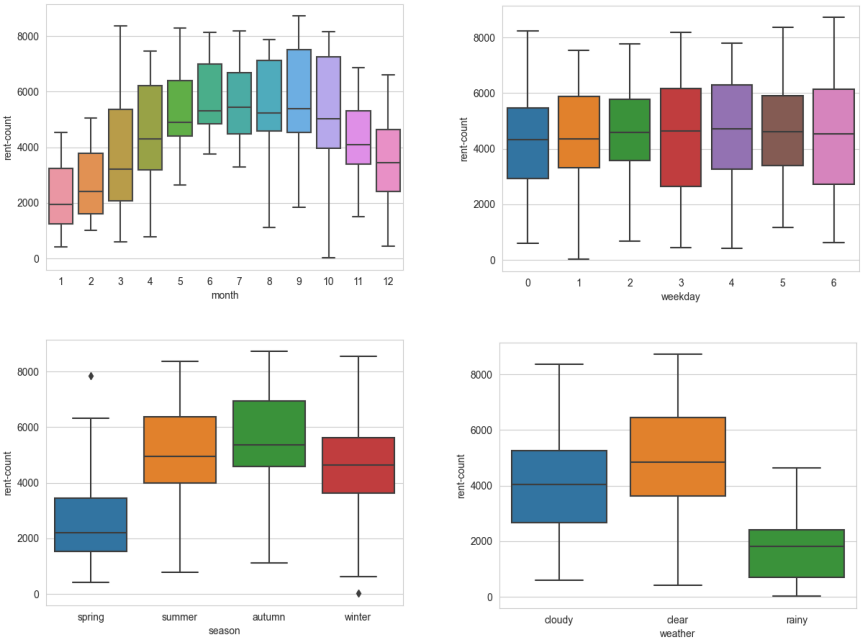

Figure 3: BoxPlot

Tramite dei bar plot noto che il numero di noleggi durante le giornate lavorative, al più delle volte, supera quelli effettuati durante i week-end e i giorni di festa, questo significa che molte persone noleggiano le biciclette come mezzo di trasporto per andare a lavoro.

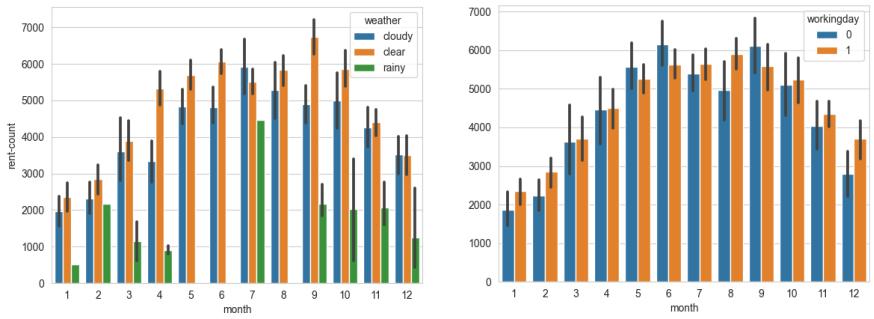

Figure 4: BarPlot

Per ultimo termino l'exploratory data analysis con un pairplot delle features numeriche, il quale ci mostra come le features formino dei cluster, questo vuol dire che sono altamente non lineari. l'unica features a tendenza lineare, come abbiamo già visto, è la temperatura.

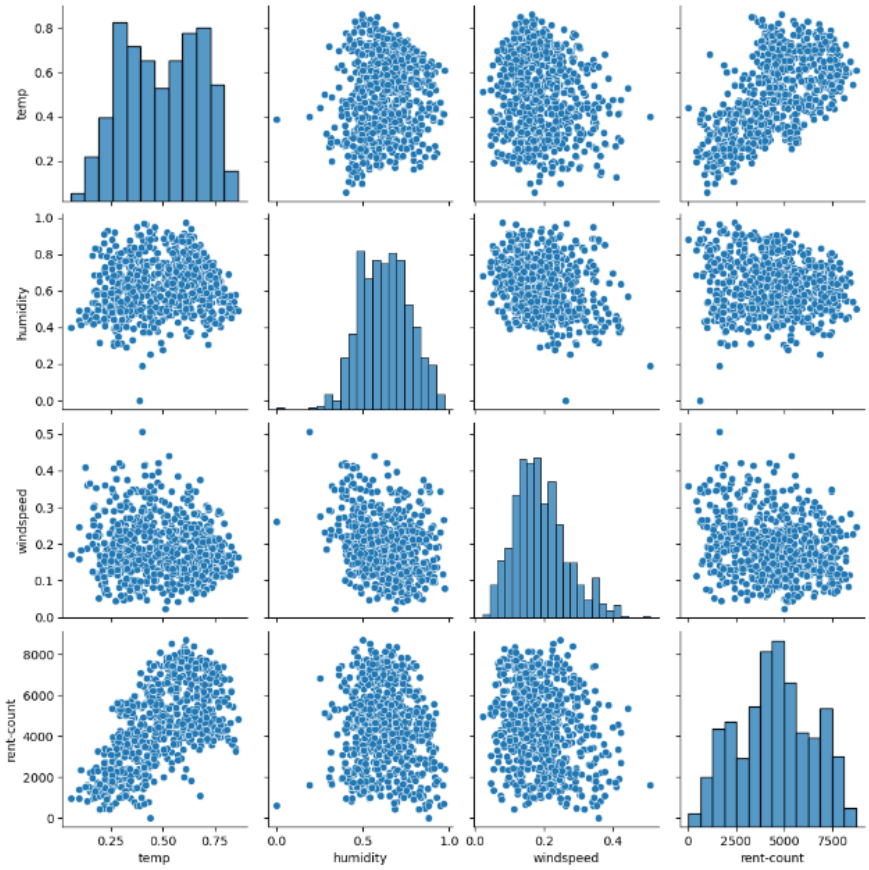

Figure 5: PairPlot

3 Discussione dei modelli

come visto nell'exploratory data analysis, le features si mostrano altamente non lineari con l'uscita, questo ci sconsiglia l'utilizzo di un modello di regressione lineare. Al suo posto decido di utilizzare un modello di regressione regolarizzato, in questo caso, mi ritrovo dopo l'eda, con un dataset abbastanza contenuto in termini di features e quindi utilizzo la ridge regression che ottimizza la funzione di costo.
Un altro modello che fa al caso nostro è quello fornito dall'impiego delle reti neurali, in cui, essendo il nostro un problema di regressione, dovremmo cercare di ridurre la loss function data dalla *Residual sum of squares*

$$\sum_{i=1}^{n}(y^i - y_p^i)^2 \tag{1}$$

Per questo problema decido di utilizzare come modelli, il Ridge regression e il MLPRegressor forniti dalla libreria Sklearn [2].
divido il dataset per l'80% in dati di training e il restante 20% in dati di testing.
I dati di training verranno nuovamente suddivisi, ed una parte di essi sarà utilizzata per model selection tramite cross validation. La ridge regression presenta la scelta di un iperparametro alfa e la scelta dell'algoritmo di convergenza.
Il MLPRegressor presenta svariati iperparametri come la dimensione dell'hidden layer, il parametro di regolarizzazione alpha, la funzione di attivazione etc..
entrambi questi due modelli, verranno testati nei seguenti modi:

1. utilizzando il dataset senza nessun tipo di pre-processing, quindi utilizzando i dati originali
2. utilizzando il dataset a seguito di pre-processing
3. in ensamble della variante migliore ottenuta

Nel Pre-processing utilizzo tecniche di normalizzazione tramite MinMaxScaler fornito da sklearn, per trasformare i dati in modo che il loro range di valori sia compreso tra (0,1). Il dataset utilizzato presenta già svariati dati normalizzati, su di essi questa trasformazione non avrà effetto. Mentre nella standardizzazione tramite StandardScaler, i dati verranno riarrangiati per avere media nulla e deviazione standard unitaria.
Nel model selection utilizzo GridSearchCV per cercare gli iperparametri più appropriati, molto utile soprattutto con la rete neurale, in quanto presenta svariati iperparametri quali: la dimensione dell' hidden layer, o l'ottimizzatore da utilizzare (Batch Gradient Descent o Stochastic Gradient descent), e anche che funzione di attivazione utilizzare all'interno dei nodi.
Alla fine di tutto ciò verranno testate le prestazione dei due modelli attraverso metriche di:

1. Mean Square Error (MSE)
2. Mean Absolute Error (MAE)
3. R^2 statistics

4 Implementazione

inizio creando il vettore delle features e il vettore delle labels, suddivido questi ultimi in training set, validation set e test set, tramite la funzione train test split.
Preparo anche i dati normalizzati e standardizzati , anche se, per la natura del dataset in esame, queste ultime trasformazioni non avranno un grosso impatto sul risultato finale.
Incomincio a definire il modello di ridge regression, inizialmente instanzio il modello utilizzando gli iperparametri di default. Dopo averlo addestrato, lo do in pasto a grid search per la ricerca del parametro di regolarizzazione alpha e della funzione di convergenza. Per la cross validation utilizzo un $k = 5$ e i dati presenti nel validation set.
Una volta terminato risulta:

parametro	Alpha	funzione di convergenza
	0.1	Regularized Least-Squares

Table 2: Iperparametri della Ridge regression

Ottenuti gli iperparametri ideali, posso procedere a riaddestrare il modello di ridge regression, sui dati di training e di validation insieme.
Ora utilizzo i dati di testing per testare i tre tipi di modelli con i tre tipi di dati preprocessati.
Ridge regression utilizzando i dati originali:

Metrica	Valore
Mean Squared Error	2392344,32
Mean Absolute Error	1328,66
Root Mean Squared Error	1546,72
R^2 Statistics	0.437

Table 3: Metriche di Ridge regression con dati non processati

In questa tabella noto come il Mean Squared Error, che penalizza maggiormente gli errori grandi, è una metrica fuorviante non essendo questo il caso. Mentre il Mean Absolute Error fa più al caso nostro in quanto rappresenta la distanza tra il valore predetto e quello effettivo. La sua interpretazione è abbastanza semplice, in questo caso con un MAE di 1328 possiamo dire che i valori predetti dal modello avranno un errore medio di +/- 1328
Il mio obbiettivo sarà proprio quello di ottenere un valore di MAE più basso possibile.
Il prossimo passo sarà di riaddestrare il modello su i dati Normalizzati tramite Min Max Scaler.

Metrica	Valore
Mean Squared Error	2398131.69
Mean Absolute Error	1329.21
Root Mean Squared Error	1548.59
R^2 Statistics	0.436

Table 4: Metriche di Ridge regression con dati normalizzati

I valori delle metriche sono rimasti praticamente identici a quelli ottenuti senza normalizzazione, questo è dovuto, come già notato durante l'EDA, alla natura del dataset in esame che presentava già di suo le features di tipo continuo normalizzate tra 0 e 1.
Vedo se riesco ad ottenere risultati migliori con dati standardizzati tramite Standard Scaler.

Metrica	Valore
Mean Squared Error	2400969.86
Mean Absolute Error	1329.30
Root Mean Squared Error	1549.50
R^2 Statistics	0.435

Table 5: Metriche di Ridge regression con dati standardizzati

Anche in questo ultimo caso in esame noto come utilizzato dati standardizzati, i valori delle metriche sono impercettibilmente diminuiti. Utilizzo come modello definitivo quello con i dati non preprocessati.
Il modello migliore presenta un R^2 score di 0.437, Ricordiamo che questa metrica varia tra 0 ed 1: quando è 0 il modello utilizzato non spiega per nulla i dati; quando è 1 il modello spiega perfettamente i dati. Nel caso in esame posso dire che quasi la metà dei dati sono spiegati dal modello di ridge regression.
Passo ora in esame il secondo modello, la rete neurale tramite MLPRegressor.

Inizio l'addestramente scegliento un numero massimo di iterazioni abbastanza elevato, e come per il primo modello, utilizzo gli iperparametri di default. Dopo averlo addestrato su i dati di training, passo il modello al grid search per effettuare cross validation e ottenere gli iperparametri migliori. In questa fase verranno scelti gli iperparametri riguardanti:

1. hidden layer sizes: cioè la dimensione dell'hidden layer
2. activation: la funzione di attivazione dell'hidden layer, la scelta sarà tra
   (a) Identità: funzione identità
   (b) Tanh: la funzione tangente iperbolica
   (c) ReLU: la funzione $f(x) = Max(0, x)$
3. solver: l'algoritmo di convergenza utilizzato, la scelta sarà tra
   (a) sgd: stochastic gradient descent
   (b) adam: stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba
   (c) lbfgs: un ottimizzatore della famiglia dei quasi-Newton methods
4. alpha: parametro di regolarizzazione
5. early stopping: se attivato è un metodo di regolarizzazione che ferma il training del modello quando la loss function è al suo minimo

Data la numerosità degli iperparametri da testare, accompagnato da un numero massimo di iterazioni abbastanza elevato, il gridsearchCV per completare la sua ricerca ha impiegato circa 40 minuti ed ha concluso la sua ricerca con:

iperparametro	**scelta**
hidden layer size	(60 , 30)
activation	identity
solver	lbfgs
alpha	0.1
early stopping	False

Table 6: Iperparametri del MLPRegressor

Utilizzando gli iperparametri forniti dalla cross validation, procedo con il riaddestramento su i dati di training e quelli di validazione. Ora verifico il comportamento del modello con i vari set di dati preprocessati.
Inizio con i dati non preprocessati

Metrica	**Valore**
Mean Squared Error	2401082.46
Mean Absolute Error	1329.30
Root Mean Squared Error	1549.54
R^2 Statistics	0.435

Table 7: Metriche di MLPRegressor con dati non preprocessati

Nonostante ho utilizzato un altro modello, ottengo circa gli stessi risultati, l'impiego della rete neurale non si sta rilevando una scelta vincente.
Proseguo a verificare i risultati ottenibili tramite preprocessing
Inizio con i dati forniti normalizzati

Metrica	Valore
Mean Squared Error	2401115.74
Mean Absolute Error	1329.30
Root Mean Squared Error	1549.55
R^2 Statistics	0.435

Table 8: Metriche di MLPRegressor con dati normalizzati

Nessun cambiamento rilevato.

Metrica	Valore
Mean Squared Error	2401095.19
Mean Absolute Error	1329.30
Root Mean Squared Error	1549.54
R^2 Statistics	0.435

Table 9: Metriche di MLPRegressor con dati standardizzati

Anche con i dati standardizzati , non c'è stato nessun cambiamento, provo ad analizzare gli output dei vari modelli per controllare se effettivamente cambia qualcosa.

Modello	Valore predetto	Valore effettivo
Ridge Regression	5775.707	
Ridge Regression (min max scaler)	5774.567	5923
Ridge Regression (standard scaler)	5777.932	
MLPRegressor	5778.119	
MLPRegressor (min max scaler)	5778.178	5923
MLPRegressor (standard scaler)	5778.122	

Table 10: Valori predetti dai modelli

Sembrerebbe esserci una sottile differenza tra i valori predetti dai modelli, anche se in linea generale, come visto dalle metriche, ottengono tutti gli stessi risultati.
Nonostante tutti gli sforzi non si riesce a superare la soglia di un R^2 superiore allo 0,5, il dataset fornito è estremamente vago.. le informazioni riguardanti l'ambiente nella maggioranza dei casi non sono riconducibili direttamente al numero di biciclette noleggiate.
Oltre all'eda provo ad attuare tecniche di features selection iterativo, come ad esempio l'applicazione di un warping method, che, per ogni n features ricerca iterativamente un sottoinsieme M_n la cui RSS (residual sum of squares) è la minore tra i sottoinsiemi possibili. Per ottenere ciò utilizzo la funzione di sklearn SequentialFeatureSelection. Questa mi permette di applicare una Forward selection o una backward selection, a seconda dei parametri forniti.
Questa funzione richiede una serie di parametri, tra cui l'estimator, cioè il modello su cui effettuare feature selection, il numero n di features da selezionare (ovviamente n deve essere minore o uguale al numero di features attualmente nel dataset). Poi bisogna fornirgli il wrapper method da utilizzare (forward selection o backward selection) , la metrica di scoring con cui valutare la performance del modello ed infine la strategia di k-fold cross validation da utilizzare.
per questo compito ho utilizzato una strategia di k-fold con k=5, forward selection con metrica di performance basata sull'R^2 score. Per quanto riguarda il numero di features ho eseguito un ciclo per testare tutte le possibili combinazione e ottenere quella che minimizza la loss function. Quindi l'algoritmo con questi parametri seleziona il sottoinsieme di features che forniscono lo score più elevato. Nel caso in esame ho ottenuto $n = 3$

indici selezionati	feature
[0 , 2 , 6]	['season', 'holiday', 'temp']

Table 11: Forward selection: index delle features selezionate

Dopo aver rimosso le features non selezionate dal wrapper method, riaddestro i modelli sui nuovi dati di training ottenuti

Ridge Regression	**Metrica**	**Valore**
	Mean Squared Error	2151436.66
	Mean Absolute Error	1273.07
	Root Mean Squared Error	1466.77
	R^2 Statistics	0.494
MLPRegressor	**Metrica**	**Valore**
	Mean Squared Error	2152435.73
	Mean Absolute Error	1273.19
	Root Mean Squared Error	1467.11
	R^2 Statistics	0.493

Table 12: Metriche dopo feature selection

Tramite features selection siamo riusciti, anche se non di molto, ad incrementare l'R^2 score e diminuire l'errore medio sui dati di testing. Come si poteva intuire dall' EDA le caratteristiche più rilevati sono state la temperatura(che aveva ottenuto un punteggio di correllazione pari a 0.63 con il target), la stagione, e se la giornata è una giornata di festa o di lavoro.

5 Risultati

La decisione del modello più performante nel task in esame si sta rivelando una scelta complicata, entrambi i modelli propongono metriche molto simili tra loro , come ultimo confronto provo ad instanziare un modello di regressione lineare per poi trarre le relative conclusioni.
Per trarre le conclusioni le metriche che vengono considerate saranno solo il Mean Absolute Error ed il R^2 statistics in quanto avendo una variabile di target dell'ordine delle migliaglia, il Mean square error, cioè la discrepanza quadratica media, risulta una metrica troppo elevata e fuorviante.
Anche per la linear regression i valori ottenuti sono gli stessi.

Metrica	**valore**
Mean Absolute Error	1273.19
R^2 Score	0.493

Table 13: Linear Regression Metriche

Utilizzo della visualizzazione grafica tramite dei residual plot e dei bar plot al fine di osservare eventuali differenze tra i modelli proposti.

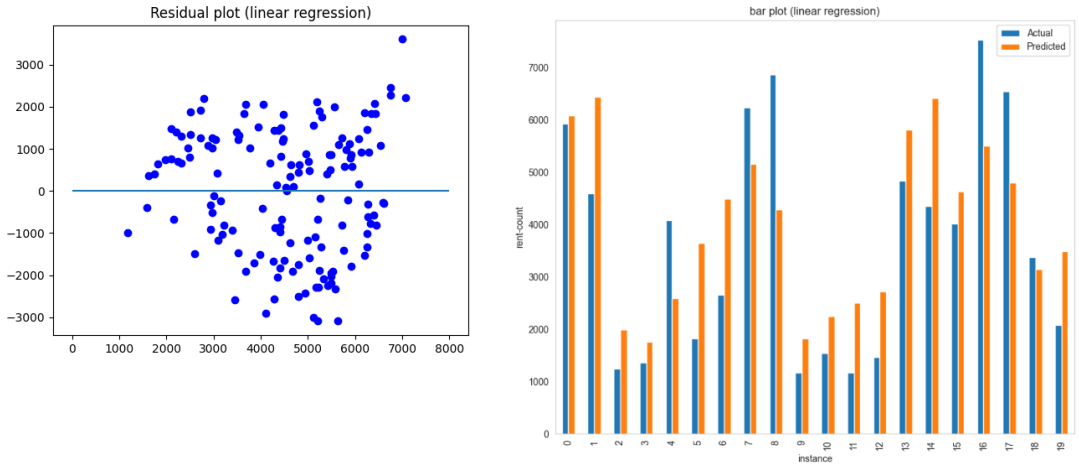

Figure 6: Linear regression plot

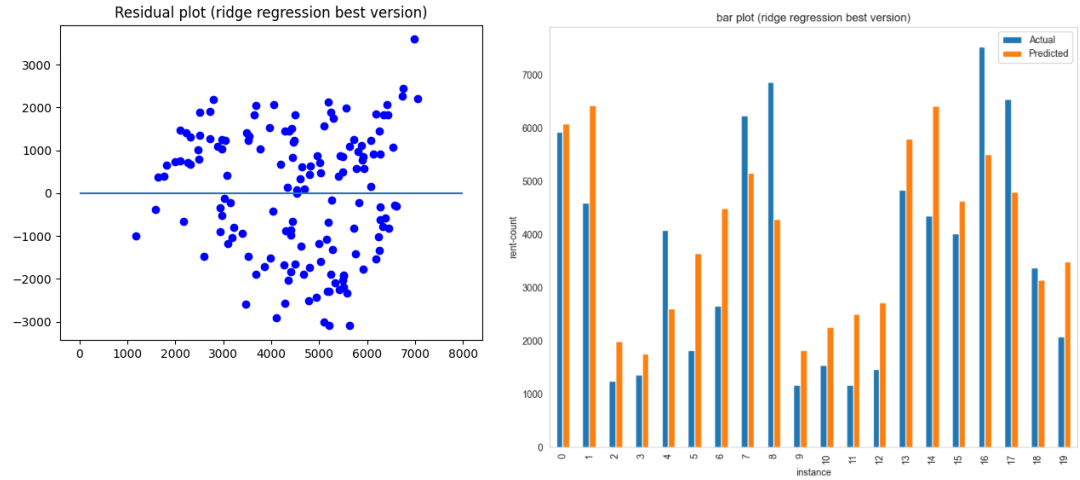

Figure 7: Ridge regression plot

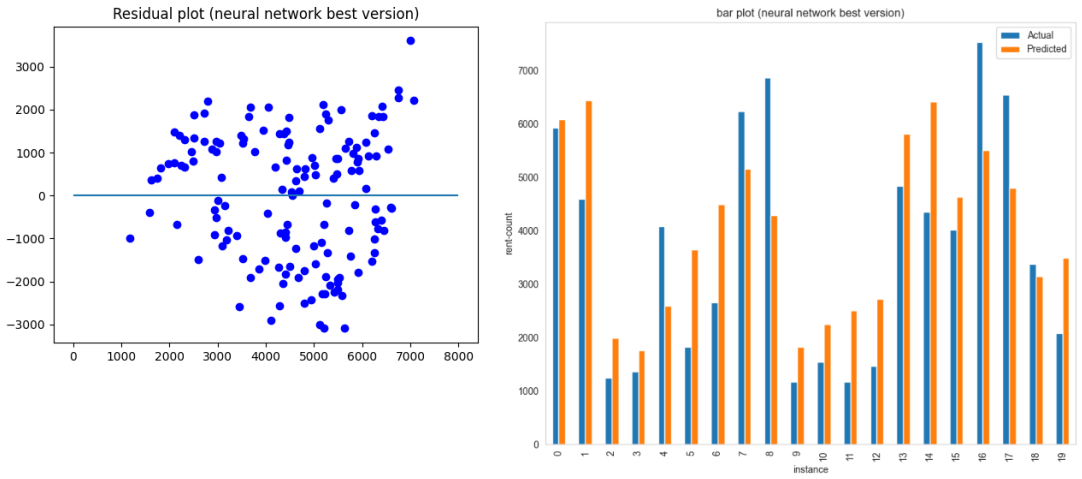

Figure 8: MLPRegressor plot

Il residual plot, mostra il valore dei residui, cioè di quanto i dati di test predetti si discostano dal valore effettivo, questo è reso più chiaro nel bar blot a fianco.
Per ultimo dò uno sguardo al tempo computazionale impiegato da ciascun modello per addestrarsi.

Modello	**Tempo di addestramento(ms)**
Linear Regression	$\simeq 0.0$
MLPRegressor	124.9704
Ridge Regression	$\simeq 0.0$

Table 14: Tempo di addestramento in millisecondi

Posso concludere dicendo che, il modello ottimale per la ridge regression è stato quello che si addestrava con i dati senza pre-processing. Mentre per la rete neurale l'ottimo lo dava quello con i dati in forma standardizzata.
Il migliore tra i due è il Ridge regression avendo un valore di R^2 score leggermente superiore ed un tempo di addestramento nettamente inferiore.

Nome	R^2 **score**	**Mean Absolute Error**	**Tempo di addestramento(ms)**
Ridge Regression	0.494	1273.07	$\simeq 0.0$

Table 15: Prestazioni modello migliore

Ottenuto tramite i seguenti parametri

1. Alpha = 0.1
2. solver = SVD(Singular Value Decomposition)

L'utilizzo di un modello di regolarizzazione come quello del ridge regression si è rivelata la scelta vincente. questo tipo di approccio aggiunge un vincolo per ottimizzare la funzione di costo, in questo modo le stime dei parametri tenderanno verso lo zero, sfavorendo le features meno importanti.

References

[1] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 1–15, 2013.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.