REPORT PROGETTO DATA & WEB MINING

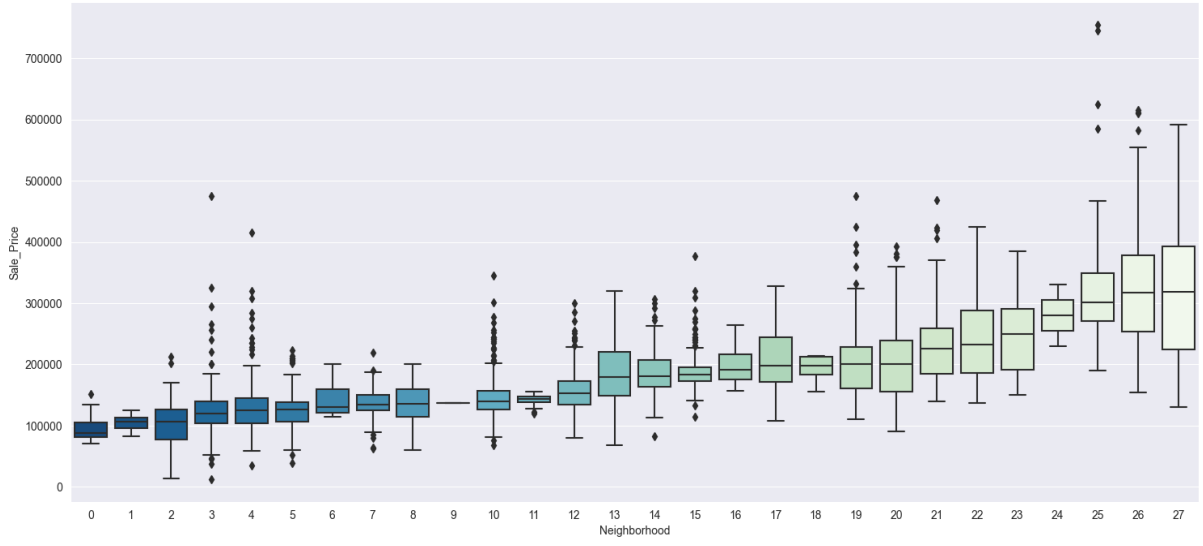
0. Introduzione

Il dataset preso in considerazione durante questo progetto vede coinvolte 2930 istanze che rappresentano ognuna una proprietà immobiliare nella città di Boston, Massachusetts. Ogni istanza è caratterizzata da 80 feature (35 numeriche e 45 categoriche) e il campo *Sale_Price* originariamente noto.
Il dataset è stato diviso fin da subito in train set e test set campionati casualmente, il train set contiene l'80% delle istanze totali e il test set contiene il 20% delle istanze totali. Il dataset di validazione non è stato creato in quanto per ogni modello è stata effettuata la Cross Validation.

1. Data-Engineering

1.1 Trasformazione feature categoriche in numeriche ordinali

Come primo passo della fase di data-engineering abbiamo trasformato i valori delle feature categoriche che mostravano un andamento crescente convertendoli da parole a valori numerici ordinali, un esempio calzante è la feature Neighborhood il cui risultato della trasformazione è mostrato nel seguente grafico.



1.2 Rimozione outliers

Successivamente abbiamo eliminato gli outliers dalle istanze del dataset controllando graficamente se vi erano valori estremi nel grafico che compara *Overall_Qual* con *Sale_Price* (in tutto sono stati individuati e rimossi tre outliers). Abbiamo riscontrato miglioramenti sui modelli successivamente alla rimozione.

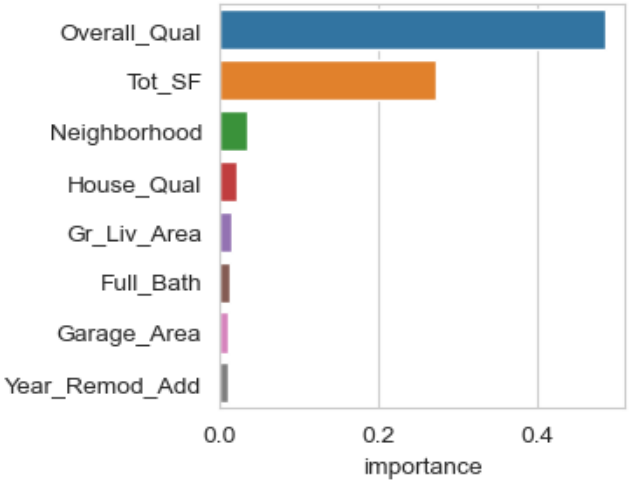
1.3 Normalizzazione dei prezzi

Prima di procedere con la creazione dei modelli abbiamo controllato la distribuzione della variabile Sale_Price e abbiamo visto che essa non seguiva una distribuzione normale, come si può vedere dal seguente grafico contenente il relativo distribution plot e il QQ-plot:

Per risolvere suddetto problema abbiamo deciso di trasformare *Sale_Price* usando la funzione logaritmo ottenendo come risultato il dato normalizzato. Quanto posto è illustrato nello stesso grafico successivo alla log-trasformazione di *Sale_Price*.

1.4 Introduzione di nuove feature

A questo punto abbiamo calcolato e introdotto nel dataset nuove feature da noi personalizzate di cui citiamo, tra le altre, *Tot_SF* (numero di metri quadri totale della proprietà immobiliare) e *House_Qual* (qualità complessiva della proprietà immobiliare). Si noti che la rilevanza di queste nuove feature e del loro contributo è confermata dallo studio dell'importanza di tutte le feature del dataset tramite l'utilizzo di un *RandomForestRegressor*.



1.5 One hot encoding

Sulle restanti features categoriche abbiamo applicato la tecnica di One Hot Encoding, il tutto sempre andando a lavorare sul train set.

1.6 Recursive feature elimination with cross-validation

Infine, dopo aver studiato la correlazione e l'importanza di tutte le feature del dataset, abbiamo utilizzato il metodo di eliminazione ricorsiva di feature con cross-validation (RFECV) della libreria *sklearn.feature_selection*. Quest'ultima modifica ci ha permesso di raggiungere performance molto soddisfacenti selezionando le feature con maggiore importanza predittiva. Abbiamo eseguito la RFECV prima su un DecisionTree e poi su un RandomForest per confrontare la bontà delle feature estrapolate dai due modelli.

Al termine della fase di data-engineering le feature che caratterizzano le istante sono diventate 142; nel test set ovviamente ne troveremo 141 in quanto *Sale_Price* non è nota per quelle istanze.

2. Predizione con modelli di classificazione

Inizialmente abbiamo cercato di approcciare il task di predizione come fosse un task di classificazione discretizzando i valori della colonna Sale_Price in diversi intervalli ordinati che costituiscono le "classi" da predire. Per fare questo abbiamo utilizzato *KBinsDiscretizer* della libreria *sklearn.preprocessing* con due principali strategie:

1. ***quantile*:** tutti gli intervalli hanno lo stesso numero di istanze al loro interno
2. ***kmeans*:** i valori di ogni intervallo hanno tutti lo stesso centro più vicino di un cluster k-means in una dimensione

Come previsto, il rendimento dei classificatori utilizzati non è stato soddisfacente e pertanto abbiamo iniziato ad approcciare il task di predizione come task di regressione.

3. Modelli di regressione

Nel processo di selezione del migliore modello di predizione abbiamo usato i seguenti modelli:

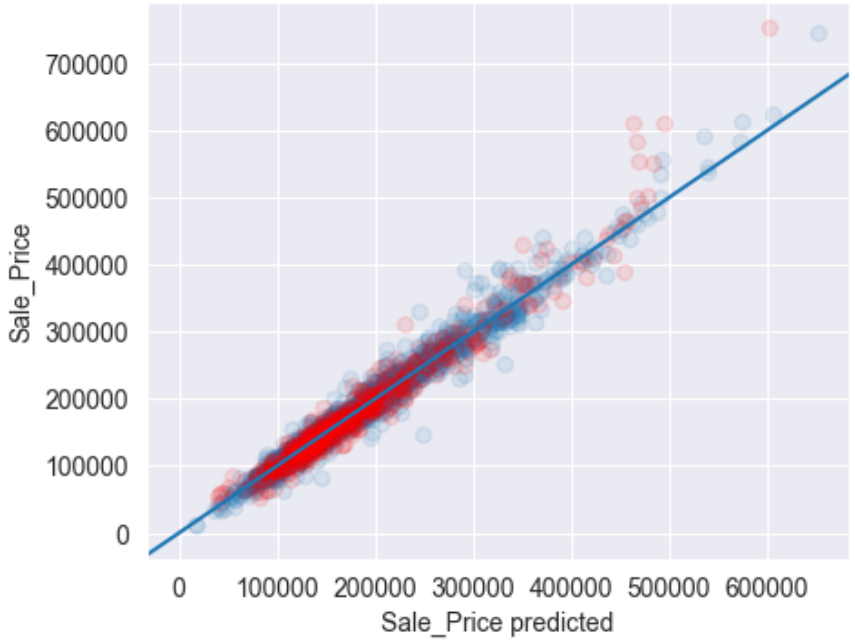
1. **DecisionTreeRegressor** (scikit learn)
2. **LinearRegression** (scikit learn)
3. **RandomForestRegressor** (scikit learn)
4. **SupportVectorMachineRegressor** (scikit learn)
5. **NeuralNetwork** (tensor flow: keras)
6. **XGBRegressor** (XGBoost)

Per i modelli (1, 3, 4, 6) abbiamo usato la GridSearchCV per andare a fare il tuning dei vari parametri del modello (quelli specificati), tramite l'uso della Cross-Validation. Per i rimanenti modelli è comunque stata usata la Cross-Validation tramite altre funzioni per tentare di ottenere modelli più corretti.
Dai modelli abbiamo derivato queste performance:

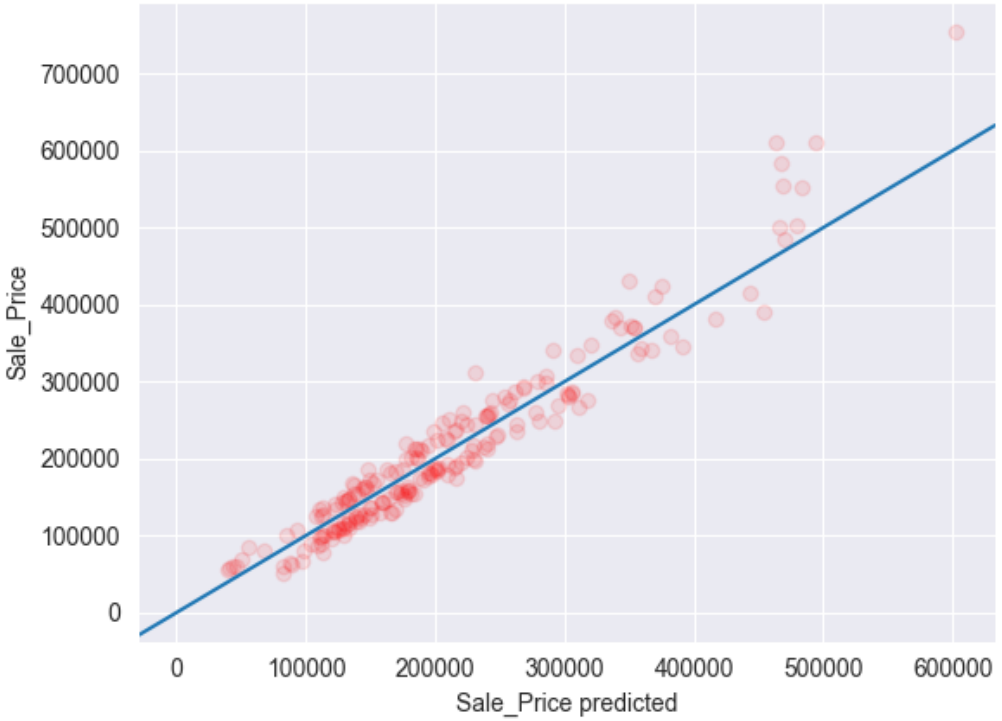
	Train MSE ($)	Test MSE ($)	Train R^2 (%)	Test R^2 (%)
DecisionTree	207.245.415	798.352.888	96,5	90
LinearRegression	409.212.375	1.167.523.567	93,1	85,4
RandomForest	115.535.634	795.859.674	98,1	90,1
SVM	673.973.639	1.278.982.259	88,7	84
NeuralNetwork	173.897.020	829.736.791	97,1	89,7
XGBRegressor	82.954.342	597.959.677	98,6	92,5

4. Miglior modello predittivo

Possiamo notare come il modello con il miglior Test MSE sia XGBRegressor, andiamo ora ad analizzare meglio cosa fa il modello. In particolare mostriamo il seguente grafico che illustra la differenza tra il vero valore del campo “Sale_Price” e la predizione del modello sia per tutte le istanze del training set (in blu) che per tutte le istanze del test set (in rosso).



E’ interessante investigare le istanze in cui la predizione era particolarmente sbagliata, pertanto abbiamo deciso di rappresentare graficamente le predizioni le cui differenze con i loro rispettivi valori reali sono maggiori della media di tutte le differenze tra “Sale_Price” predetta e reale.



Dal grafico si può notare che le istanze predette peggio della media si distribuiscono in maniera più o meno uniforme, quanto meno sino a una certa soglia di prezzo; l'aumentare dell'errore per proprietà il cui prezzo di vendita supera circa i 400000 USD è probabilmente motivato dal fatto che il dataset contiene poche istanze di questo tipo, rendendole poco rappresentate e pertanto più difficili da predire.

5. Conclusioni

In conclusione proponiamo di seguito una serie di possibili miglioramenti applicabili al nostro progetto con l'intento di migliorare le performance di predizione.
In primo luogo, in condizioni ideali, sarebbe appropriato aumentare il numero di istanze nel dataset, raccogliendo dati soprattutto su proprietà immobiliari il cui prezzo di mercato è poco rappresentato nel dataset attuale.
In secondo luogo si potrebbe approfondire il processo di tuning degli iper-parametri dei vari modelli illustrati con l'obiettivo di riscontrare un ulteriore miglioramento delle performance.
Inoltre uno studio approfondito degli outliers, ad esempio usando un oggetto *RandomForest*, potrebbe portare a delle nuove considerazioni utili per aumentare il potere predittivo dei modelli.
Infine si potrebbe sperimentare un approccio al processo di feature selection che coinvolge la Principal Components Analysis e analizzare l'eventuale cambiamento delle performance dei modelli per capire se sono state ottenute performance migliori di quelle attuali.