Relazione del corso di "Data and Web Mining"

Autori: Andrea Giacomello (884910), Serena Zanon (887050)

28 gennaio 2023

1 Introduzione al progetto

Il progetto proposto consiste nel predire i prezzi di abitazioni situate in una zona degli Stati Uniti d'America. Lo scopo, oltre a quello di individuare un modello che possa predire in maniera ottimale i prezzi, è principalmente quello di approcciarsi alla Data Mining e di provare a sperimentare diversi modelli e valutare e analizzare i risultati ottenuti da essi.

Il progetto è diviso in cinque parti principali:

1. Visualizzazione e presentazione dei dati;
2. Pulizia del dataframe e individuazione di outliers;
3. Valutazione e analisi di modelli presenti nella libreria sklearn;
4. Valutazione e analisi dei modelli presenti nelle librerie XGBoost e CatBoost;
5. Valutazione e analisi delle predizioni e delle feature.

2 Notebook1

In questo Notebook si presenta il dataframe nel suo complesso, visualizzando i dati contenuti in esso: si visualizza la dimensione del dataset fornito che risulta essere (2930, 82), (reperibile al seguente link), le informazioni associate alle sue feature, il loro nome e i possibili valori che queste possono assumere.

Una prima osservazione è che le feature non sono solo numeriche ma anche categoriali. Per questo motivo si è scelto di adottare la tecnica di 'One-Hot Encoding' per poter gestire tali feature.

Al termine del Notebook si effettua lo split in training-set e test-set in modo che il primo occupi l'80% del dataset e il secondo il rimanente 20%.

3 Notebook2

Il Notebook2 tratta l'individuazione degli outliers e la pulizia e l'aggiunta di feature.

3.1 Outliers

Il metodo con cui si è deciso di individuare gli outliers (sia sul dataset originale, che sul dataset pulito) è dato dal confronto di due metodi: LocalOutlierFactor e IsolationForest; entrambi questi metodi sono contenuti nella libreria sklearn.

Questi due metodi sono eseguiti in due funzioni differenti ed entrambe restituiscono:

1. Il numero di istanze definite come outliers;
2. Il numero di istanze non outliers;
3. Il risultato individuato dal modello in esame: Se il valore riferito a un'istanza è maggiore di 0, allora questa viene identificata come outlier; al contrario, se il valore riferito a un'istanza è minore o uguale a 0, allora questa non viene considerata come un outlier.

Si è ritenuto di considerare outliers solamente le istanze che sono definite outliers da entrambi i modelli.

3.2 Feature

Inizialmente viene presentata una *HeatMap* per mostrare se ci sono eventuali relazioni tra le feature nel dataset fornito: alcune feature sono correlate fra loro perchè riferite a uno stesso ambito. Ad esempio, si può notare che *GarageCars* e *GarageArea* hanno una forte relazione. Ad ogni modo, nel corso di questo paragrafo verranno elencate e spiegate le scelte effettuate.

Innanzitutto, visualizzando i valori distinti che le feature assumono, si riscontra che sono presenti delle scale di qualità. Si è deciso di gestirle ordinando dal più basso al più alto valore possibile assunto dalla singola scala e di associare ad ogni valore un numero. In questo modo è più facile per i modelli che si analizzaranno in seguito, gestire tali feature. Ad esempio, nell'albero di decisione (uno dei modelli adottati) si può essere interessati a individuare istanze la cui qualità generale dell'abitazione sia inferiore o superiore a un determinato parametro. Così facendo, è più semplice per l'albero individuare lo split migliore.

Si sono aggiunte le seguenti feature:

1. Il numero totale di bagni nel seminterrato. Si è deciso di aggiungerla in modo da tenere solamente una feature riguardante i bagni del seminterrato, di fatti questa è il risulato della somma tra il numero dei bagni completi e dei mezzi bagni presenti in cantina.

2. Il numero totale di bagni al piano superiore. Il motivo è analogo a quello per il numero totale di bagni nel seminterrato. *Specifica*: Non si è scelto di creare un'unica sola feature per il numero totale dei bagni presenti nell'abitazione perchè si è ritenuto importante sapere se il seminterrato avesse o meno dei bagni come fattore di predizione del prezzo.

3. Il valore della superficie totale dei piani superiori. Tale feature è data dalla somma della superficie del primo e del secondo piano. A questo punto del Notebook2 si ripresenta la *HeatMap*. Da questa si può osservare che ora esiste una relazione particolarmente forte tra la feature appena creata e *GrLivArea*. Di conseguenza, viene mostrato un grafico relativo alla loro relazione e si può osservare che questa risulta essere particolarmente lineare. Se ne può dedurre che la maggior parte delle abitazioni abbiano solo il primo piano.

4. La presenza di piscine. Nel dataset fornito la feature *PoolQC* è una scala di qualità rispetto alla piscina presente nell'abitazione, se questa vi è. Si può notare dai valori assunti dalla feature ora modificata, che quasi tutte le abitazioni non hanno una piscina. Perciò piuttosto di tenere una feature che riguarda la qualità di essa si è preferito tenere una feature che riguarda la sua presenza. (0 - se non vi è la piscina; 1 - se vi è la piscina).

5. La presenza e l'area del porticato. Nel dataset fornito sono presenti quattro feature riguardanti le diverse tipologie che il porticato di un'abitazione può assumere, in particolar modo viene specificata l'area del porticato. La scelta fatta per queste feature è quella di riassumere la loro informazione in due sole feature: una contenente la presenza di eventuali porticati e l'altra la somma delle aree che questi hanno.

6. Il valore del prezzo medio per quartiere. Tale scelta di aggiunta è stata fatta perchè in genere, nel mercato immobiliare, il prezzo medio per quartiere influisce sul prezzo di un'abitazione. Ad esempio, quartieri malfamati avranno un prezzo medio inferiore rispetto a quartieri benestanti.

Per ogni feature aggiunta è stato creato un grafico in cui nelle ascisse è presente la nuova feature e nelle ordinate è presente la feature target, ovvero *SalePrice*, in modo da valutare la relazione tra queste due feature.

3.3 Altre trasformazioni

Oltre alle trasformazioni sulle scale di qualità, si è ritenuto opportuno modificare anche le feature che contengono due soli valori o in cui un valore predomina sugli altri. Si sono ottenute in questo modo delle feature binarie più facili da gestire per i modelli.

3.4 Drop di feature

Nel corso del notebook si possono notare diverse eliminazioni di feature perchè ritenute o ripetitive o non utili ai fini della predizione o ambigue.

Nel primo caso si sono contemplate le feature riguardanti

1. Il numero di mezzi bagni e di bagni completi sia del seminterrato che dei piani superiori perchè si è scelto di mantenere il numero complessivo di essi;

2. La feature *GrLivArea* perchè assume per la maggior parte delle istanze il medesimo valore della feature pari alla superficie totale dei piani superiori (relazione fortemente lineare);

3. La feature *PoolQC*, in modo da mantenere solo l'informazione sulla presenza. Oltre a questa feature, si è eliminata *PoolArea* perché, dato che poche abitazioni hanno inclusa una piscina, si è ritenuta più rilevante l'informazione data dalla presenza di essa che della sua area;

4. Le feature contenenti l'area dei diversi tipi di porticato che l'abitazione può avere.

Nel secondo caso si è contemplata la feature *id* che contiene semplicemente un valore numerico compreso tra 1 e 2930 assegnato a ogni abitazione.

Infine nel terzo caso, si sono contemplate le feature *Fence* e *Electrical*. Queste due feature comprendono dei valori appartenenti a due ambiti diversi:

1. La feature *Fence* contiene valori riguardanti la privacy che una recinzione può garantire e il materiale di cui è costituita;

2. La feature *Electrical* contiene valori riguardanti la tipologia di essa e una relativa scala di qualità.

Si è deciso di eliminarle perchè si è ritenuto non opportuno tenere delle informazioni di tipo diverso all'interno di una stessa colonna. Inoltre, non si riuscirebbe nemmeno a individuare dei valori sufficientemente corretti per le istanze in cui è presente un valore riguardante un ambito piuttosto che un altro. Ad esempio, non si può essere certi che se il materiale della recinzione è un buon legno allora questa per certo garantirà una buona privacy e non è detto che in base a una determinata tipologia di elettricità sia corrisposto un determinato valore nella scala di qualità.

4 Notebook3

Il Notebook3 è dato da un insieme di cinque notebook contenenti ciascuno l'analisi di un modello diverso. Ogni modello viene valutato e analizzato sui dataset ottenuti dal Notebook2.

4.1 Regressione Lineare

Il primo notebook analizza la regressione lineare sul dataset fornito e sul dataset fornito senza outliers. Si è deciso di allenare tale modello solo sul dataset fornito in modo da poter ricavare l'errore baseline, dato che alla base del modello di regressione lineare vi è un ragionamento semplice, ossia il voler stimare i valori tramite una retta. Nei notebook successivi si cercherà di trovare un modello che possa essere migliore rispetto al modello base.

4.2 Random Forest

Nel secondo notebook è presente la Random Forest. Inizialmente, vi è un modello base, applicato e valutato su tutti e 4 i dataset: dataset fornito, dataset fornito senza outliers, dataset pulito, dataset pulito senza outliers. Dalle valutazioni risulta che il modello in cui la predizione è migliore è il dataset pulito senza outliers. Nel resto del notebook si fa tuning sui parametri e per questo si è deciso di lavorare sul dataset pulito completo (con gli outliers) in modo da analizzare tutte le possibili istanze. La motivazione di aver posto la Random Forest come secondo modello non è casuale: la Random Forest adotta delle tecniche per individuare le feature principali, o meglio, le feature con la più alta significatività per il modello. In questo notebook viene adottata la tecnica denominata RFECV, acronimo di Recursive Feature Elimination Cross-Validation, ossia vengono selezionate le feature migliori secondo la tecnica del Cross-Validation eliminando ricorsivamente quelle risultanti meno rilevanti. Prima di individuare le migliori feature, si è applicata la già citata tecnica del Cross-Validation direttamente sul dataset per ricercare il numero migliore di alberi che costituiscono la foresta in modo da ottenere la predizione migliore possibile. Perciò riassumendo, sul dataset pulito con outliers si è applicata la tecnica del Cross-Validation per individuare il numero migliore di alberi per la foresta e, con tale dato, si sono cercate le feature migliori. Successivamente vengono modificate le dimensioni dei dataset puliti e viene allenato il modello il cui valore del numero di alberi è pari al valore individuato dalla tecnica di tuning. Infine tutti i dataframe puliti (dataframe pulito con outliers, dataframe pulito senza outliers, dataframe test) vengono convertiti in un nuovo file csv che verrà utilizzato dai prossimi notebook per eseguire i confronti.

In questo modo i dataset puliti cambiano: da training-set pulito a training-set pulito su cui si è applicata RFCEV e da training-set pulito senza outliers a training-set pulito su cui si è applicata RFECV senza outliers e da test-set pulito a test-set pulito su cui si è applicata RFCEV.

4.3 Albero di decisione

In questo notebook si valutano le prestazioni dell'albero di decisione sui dati forniti. Il principale problema dell'applicare al meglio questo modello è quello di trovare il numero migliore di foglie da far sviluppare all'algoritmo. Tale numero è stato trovato grazie all'applicazione della tecnica di Cross-Validation su tutti e 4 i dataset in modo che ognuno individuasse il proprio migliore numero di alberi. Si possono notare delle differenze tra dataset e dataset, infatti non tutti i modelli hanno individuato lo stesso numero di foglie, ma nonostante ciò per tutti e 4 i dataset il valore ideale è compreso tra le 40 e le 50 foglie. Infine, vi sono delle visualizzazioni grafiche di diverse misure derivanti dal modello per i test-set (test-set del dataset fornito e test-set del dataset pulito). Le misure prese in considerazione sono il $Bias^2$, la varianza e l'errore. Successivamente si sono applicate le tecniche di Bagging e Boosting per poter valutare eventuali miglioramenti in tali misure e si sono messe a confronto. Dalle valutazioni finali deriva che il boosting abbia avuto un impatto maggiore sul miglioramento complessivo delle misure.

4.4 k-NN

Nel quartultimo notebook si analizza il comportamento del modello k-NN, ovvero K-Nearest Neighbor. Tale modello prende in considerazione i vicini dell'istanza per poterne predire il valore richiesto. Come per l'albero di decisione, anche qui vi è un problema legato al trovare il miglior numero di vicini per la migliore predizione. Inizialmente, vi è il modello base (con numero di vicini pari a 3) e poi si attua Cross-Validation sui dataset per trovare il valore migliore di k (numero di vicini). In realtà questa tecnica non porta a un miglioramento dello score e dell'errore rispetto al modello base, perciò si prova a cercare il valore di k senza Cross-Validation, quindi basando la ricerca solamente sul training-set. Sfortunatamente, per ciascun modello, il k migliore risulta essere 1: significa che il modello prende in considerazione la singola istanza per predirne il valore target. Di conseguenza, le considerazioni finali vengono basate solamente sui modelli base. (Se non si considera un numero di vicini superiore in senso stretto a 1, allora il modello k-NN sarebbe superfluo).

4.5 SVR

Nel quinto e ultimo notebook si adotta il modello SVR, acronimo di Support Vector Regressor. Tale modello cerca di risolvere un problema multidimensionale in modo lineare. Anche in questo modello come nei precedenti è presente un parametro su cui poter fare tuning: il parametro di regolarizzazione. Si è creato un modello base, in cui tale parametro è pari a '1.0' e lo score raggiunto dai modelli non risulta essere per nulla soddisfacente. Di conseguenza, si applica nuovamente la tecnica del Cross-Validation su tale parametro per individuare il valore migliore. Il parametro di regolarizzazione, altrimenti indicato con 'c', serve come grado di importanza che viene attribuito alle predizioni errate: più è alto tale valore, meno saranno le predizioni errate e minore sarà l'errore del modello; contrariamente più tale valore è basso e più il numero delle predizioni errate aumenta e con esso l'errore. Dopo tali analisi, tutti e 4 i dataset restituiscono il valore di c più alto possibile; scelta prevedibile dato il significato appena illustrato di tale parametro.

5 Notebook4

Questo notebook, come il notebook precedente, è suddiviso in due sezioni. La prima contiene le analisi sul modello XGBoost, la seconda sul modello CatBoost.

5.1 XGBoost

XGBoost è una libreria open source che sfrutta il gradient boosting per essere altamente efficiente e veloce. I suoi modelli usano dei gradient boosting tree (GBDT). Dopo aver creato gli alberi, il modello li controlla uno per uno il prediction score e li somma tra di loro per avere lo score finale, la differenza con la Random Forest è nella fase di allenamento del modello. Essa infatti prevede l'ottimizzazione di una funzione, nel caso dell'XGBoost è: $obj(\theta) = \sum_i^n l(y_i, \hat{y_i}^t) + \sum_{i=1}^t w(f_i)$, dove $w(f_i)$ è la complessità dell'albero e f_i è la funzione che bisogna imparare dagli alberi, contenenti la loro struttura e lo score delle foglie. Il valore $\hat{y_i}^t$ è il valore predetto allo step t ed è uguale a: $\hat{y_i}^t = \sum_{k=1}^t f_k(x_i)$ A seconda della loss function la formula può variare, andando ad adattarsi ad essa. XGBoost permette anche di creare le proprie funzioni da implementare. Infatti, la libreria permette di andare a modificare quello che è l'obiettivo del training, permettendo all'utente di andarsi a scrivere le proprie funzioni e a passarle come parametro al modello. Vi sono anche funzioni già implementate dalla libreria, queste sono: 'squarederror', 'squaredlogerror', 'logistic',

'absoluteerror', più molte altre. Dalla versione 1.5 XGBoost prova a implementare una funzione per la gestione automatica delle feature categoriche, tuttavia essa è ancora in fase sperimentale. Per questo essa non verrà usata nel progetto. Esattamente come nella Random Forest e nel CatBoost anche XGBoost permette di calcolare l'importanza delle feature.

5.2 CatBoost

CatBoost è una libreria open source che si pone l'obiettivo di migliorare il gradiente negli alberi decisionali. In tale libreria vi sono modelli target-based, ovvero dei modelli che concentrano il loro focus sul come i valori delle singole feature impattano il valore da predire. I modelli accettano in input diverse tipologie di feature tra cui quelle di testo e quelle categoriali senza la necessità di doverle elaborare precedentemente; l'importante è che vengano specificate quali siano in input alla costruzione del modello. Le prestazioni dei modelli della libreria CatBoost sono ottime anche con i valori base e di default imposti dalla libreria stessa e per questo viene definita una libreria ad alte prestazioni. Quindi dei vantaggi non banali che questa porta sono la riduzione di tempo nell'elaborare feature non numeriche e nell'ottimizzazione dei parametri del modello e inoltre riduce la possibilità di fare overfitting. Tale libreria consente infine di poter individuare le feature che sono state ritenute importanti dal modello, come per XGBoost e per la Random Forest con RFECV.

6 Notebook5

Questo è l'ultimo notebook che compone il progetto e contiene delle valutazioni sulle predizioni e sulle migliori feature individuate da XGBoost e da CatBoost. All'inizio del notebook vengono presentati nuovamente dei grafici in cui si mettono a confronto le predizioni dei modelli migliori di ogni notebook precedente e i valori reali della colonna target delle istanze. Da questi confronti risulta che i modelli migliori siano la Random Forest, XGBoost e CatBoost. Di questi vengono analizzate le prime 50 istanze migliori (in cui l'errore medio assoluto risulta essere il più basso) e si mostrano dei boxplot per visualizzare in quali istanze questi modelli predicono in modo più accurato. I boxplot si basano su *SalePrice*, ovvero la colonna da predire, e su *TotalSFFloors*, ovvero la colonna che somma la superficie del primo e secondo piano. Si è scelta tale feature perchè (indubbiamente) la superficie totale di un'abitazione incide sul prezzo. La valutazione viene effettuata anche sulle peggiori 50 predizioni.

Infine vi sono dei grafici per le feature ritenute importanti da XGBoost e CatBoost. Tra queste una sola è in comune a entrambi gli algoritmi; un'ipotesi potrebbe essere che dato che i due modelli si basano su apprendimenti differenti cerchino e individuino feature diverse a seconda delle loro esigenze.

7 Misura di qualità adottata

All'inizio del progetto si sono riscontrate delle difficoltà nell'individuare una misura di qualità dei modelli che potesse essere effettivamente valida per selezionare il modello migliore. Nella prima parte, è stata adottata come misura di riferimento l'MSE, ovvero il mean squared error, ma ciò comportava che errori negativi e positivi si influenzassero tra loro ottenendo come valori delle cifre molto alte e poco significative dell'errore in sè. Successivamente si è deciso di adottare il MAE, ossia il mean absolute error, così da non far condizionare tra loro errori negativi e positivi.

8 Conclusioni finali

Nei modelli in cui lo score si avvicina molto a 1, il dataset utilizzato è il dataset pulito e ciò va a rinforzare l'idea che la pulizia sia stata un fattore determinante nell'intero processo del progetto. Nonostante questo, non tutti i modelli analizzati riescono a migliorare lo score e l'errore individuato dal modello base, ovvero la regressione lineare. Il modello migliore in assoluto si conferma essere CatBoost, al secondo posto vi è la Random Forest e subito dopo XGBoost.