King County, Washington: analisi vendita immobiliare

MICHELE GAZZOLA[1], SARA NOCCO[1], GIACOMO STOFFA[1], AND MIRKO TRITELLA[1]

[1] *Università degli studi di Milano Bicocca, CdLM Data Science*

Compiled February 3, 2022

L'implementazione di modelli per la previsione del prezzo è un task molto popolare nell'ambito del Machine Learning. In questo report si analizza il problema di stima dei prezzi applicandolo ad un dataset di immobili venduti tra il 2014 ed il 2015 nella Contea di King, stato di Washington (Stati Uniti). Sono stati utilizzati diversi metodi di apprendimento supervisionato propriamente addestrati, tra cui modelli di regressione lineare, modelli di regressione ad albero e regressione tramite reti neurali artificiali, al fine di individuare quelli capaci di risolvere in maniera più appropriata il problema per questo dataset.

Keywords: supervised learning, linear regression, regression trees, multilayer perceptron, cluster analysis

1. INTRODUZIONE

Il nostro studio nasce dalla curiosità di analizzare le caratteristiche del mercato immobiliare negli Stati Uniti negli anni 2014 e 2015, anni considerati interessanti in quanto caratterizzati dalla fine della crisi immobiliare iniziata nel 2006. La domanda di ricerca poste su cui questo articolo si concentrerà è: Quali sono le caratteristiche che più influenzano il prezzo di una proprietà?

Per rispondere alla domanda di ricerca è stato utilizzato il dataset disponibile sulla piattaforma Kaggle [1]: esso contiene i prezzi delle case vendute tra il Maggio 2014 e il Maggio 2015 nella Contea di King, la più popolosa all'interno dello Stato di Washington (Stati Uniti). [3].

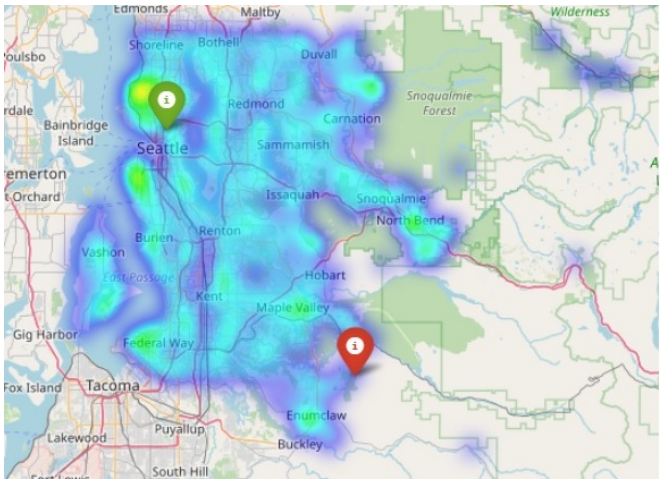

Fig. 1. Mappa di densità delle proprietà in vendita e proprietà estreme: la più cara (in verde) e la meno cara (in rosso)

Seattle è il capoluogo della Contea di King: con i suoi 750.000 abitanti, essa è la città più popolosa della Contea [2] e tra città più popolose negli Stati Uniti. Il suo sviluppo lo ha portato ad essere tra le prime cinque città per aumento della popolazione fino al 2015 con una crescita annuale del 2,1% Questa importanza è sicuramente data dalla sua posizione strategica: infatti, all'interno del territorio è presente il quarto porto più grande del Nord America considerato uno snodo fondamentale per il commercio asiatico. Inoltre, la città di Seattle offre un'altissima qualità della vita, possiede il numero più alto di laureati di tutti gli Stati Uniti (53,8%) ed è base di grandi aziende tecnologiche come Microsoft, Boeing, Amazon e Real Networks.

L'articolo è così costruito: inizialmente è stato introdotto il dataset, è stata fatta un'analisi preliminare sui dati e mostrata la fase di preprocessing; successivamente sono stati presentati i modelli utilizzati e, infine, sono state riportate le valutazioni ed i risultati ottenuti.

2. PRESENTAZIONE DEL DATASET

A. Descrizione delle variabili

Il dataset analizzato è composto da 21613 istanze e 21 attributi, ed è privo di valori mancanti. Di seguito l'elenco delle variabili originali, così come pervenute originariamente dalla fonte, con relativa descrizione:

- *id*: identificativo univoco dell'immobile venduto;
- *date*: data di vendita dell'immobile;
- *price*: prezzo dell'immobile venduto (in USD);
- *bedrooms*: numero di camere da letto;

- *bathrooms*: numero di bagni, dove 0.5 sta ad indicare un bagno con toilette ma senza doccia;
- *sqft_living*: metratura totale degli ambienti interni dell'immobile;
- *sqft_lot*: metratura dello spazio esterno dell'immobile;
- *floors*: numero di piani;
- *waterfront*: dummy variable per indicare se l'immobile è affacciato a sorgenti d'acqua;
- *view*: indice da 0 a 4 per indicare la qualità della vista dell'immobile;
- *condition*: indice da 1 a 5 segnalante la condizione dell'immobile;
- *grade*: indice da 1 a 13, dove 1-3 caratterizza immobili dalla costruzione e design insufficienti, 7 indica un livello di costruzione e design nella media e 11-13 denota un livello di costruzione e design alti;
- *sqft_above*: metratura dello spazio interno della casa che si trova sopra il livello del suolo;
- *sqft_basement*: metratura dello spazio interno della casa che si trova sotto il livello del suolo;
- *yr_built*: anno in cui l'immobile è stato costruito;
- *yr_renovated*: anno in cui l'immobile è stato restaurato l'ultima volta;
- *zipcode*: codice postale a cui l'immobile fa riferimento;
- *lat*: latitudine;
- *lon*: longitudine;
- *sqft_living15*: metratura degli interni dei 15 immobili più vicini;
- *sqft_lot15*: metratura dello spazio esterno dei 15 immobili più vicini;

B. Analisi Esplorativa dei dati

La prima attività svolte è stata quella di applicare il nodo "Table Manipulator" per poter osservare il nostro dataset. Prima di inserire il nostro file all'interno della piattaforma, abbiamo applicato alcuni cambiamenti direttamente sul file csv. Per aumentare l'informatività abbiamo deciso di sostituire i valori '0' dell'attributo *yr_renovated*, i quali stavano a rappresentare le case che non avevano subito ristrutturazioni, con il valore dell'anno dell'attributo *yr_built* eliminando in tal modo una colonna rendendo il dataset meno "pesante" in termine di computazione. Un'ulteriore modifica al dataset riguarda la creazione di una colonna *restoration_time*, la quale è stata generata dalla differenza tra l'attributo *yr_renovated* e *yr_built* per vedere meglio le case che hanno subito o meno una ristrutturazione e aumentare ancora di più l'informatività del nostro dataset. Abbiamo successivamente visualizzato informazioni ulteriori tramite analisi esplorativa in Python disponibile in questo notebook [6].

Applicando il nodo "Statistics" abbiamo potuto vedere se gli attributi del nostro dataset presentassero dei valori asimmetrici oltre a permetterci di controllare anche l'eventuale presenza di valori mancanti tra i nostri dati. Successivamente per studiare la variabile target 'price', abbiamo deciso di utilizzare un boxplot per osservare i suoi valori, come mediana, primo e secondo quartile e ovviamente il suo comportamento per quanto riguarda asimmetrie e outliers. Da ciò che appare si nota una forte asimmetria positiva con una coda molto lunga sulla parte superiore del boxplot.

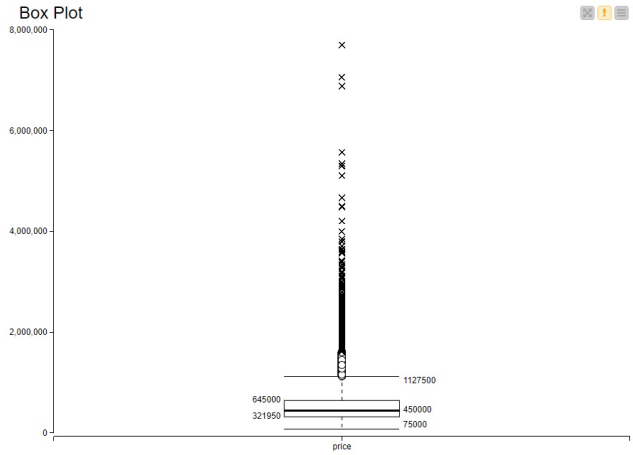

Fig. 2. Visualizzazione boxplot del prezzo.

Abbiamo poi utilizzato i boxplot anche per valutare le altre variabili nel nostro dataset:

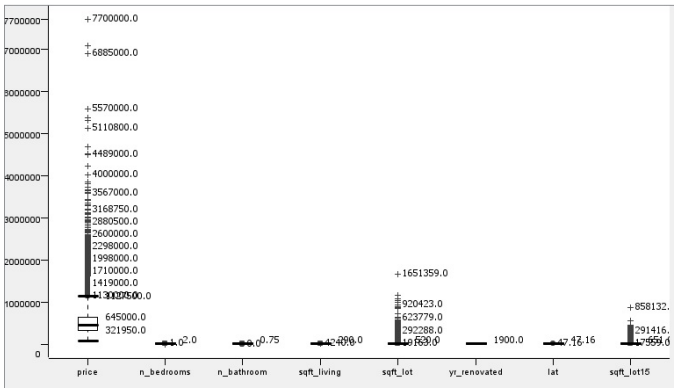

Fig. 3. Visualizzazione boxplot di tutte le variabili.

Si nota dal grafico che tre delle nostre variabili a disposizione presentano asimmetria positiva: price, *sqrt_lot* e *sqrt_lot15*. Valuteremo in seguito se sarà necessario applicare una trasformazione logaritmica a queste variabili al fine di normalizzare la loro distribuzione ed eliminare di conseguenza la forte asimmetria positiva. All'interno della sezione relativa al preproccessing spiegheremo come abbiamo trattato gli outliers nel dataset.

Inoltre, applicando il nodo "linear correlation" abbiamo valutato in linea generale, senza applicare correzioni, le correlazioni che esistono fra le nostre variabili, il che è molto importante per la previsione che dobbiamo fare, perché tenere all'interno del nostro dataset variabili eccessivamente correlate fra di loro comporterebbe l'inclusione del fenomeno della multicollinearità nei nostri modelli andando a distorcere la previsione.

Abbiamo inoltre deciso di controllare tramite un istogramma se ci fosse un aumento o una diminuzione di costruzioni di case

nel corso degli anni. Il nostro dataset presenta case costruite nel periodo compreso tra 1900 e 2015 e si nota chiaramente che nel corso di questo periodo c'è stato un forte aumento del numero di case, probabilmente legato all'aumento della popolazione, come mostra il grafico:

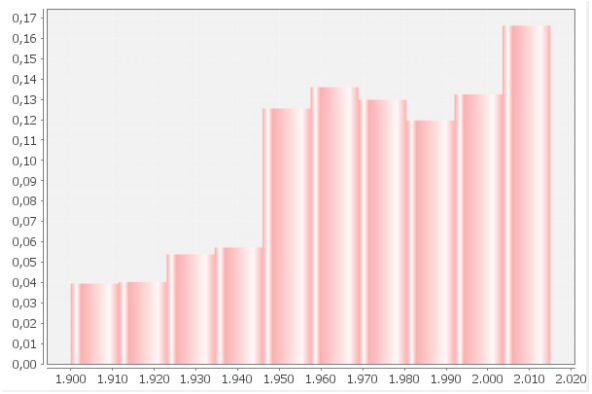

Fig. 4. Istogramma: numero di case costruite negli anni dal 1900 al 2015.

Inoltre, abbiamo voluto studiare le 50 case più costose in relazione alle 50 case meno costose sulle variabili *n_bedrooms*, *n_bathrooms*, *sqrt_living* e *sqrt_lot* per capire se ci fosse un pattern ricorrente che influenzasse il prezzo. Dai grafici questo è ciò che si può notare:

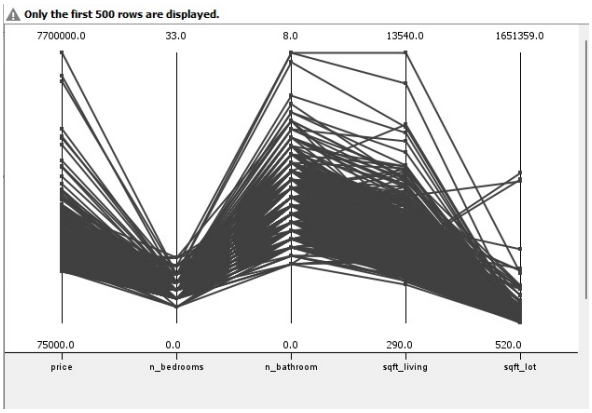

Fig. 5. 500 case più costose.

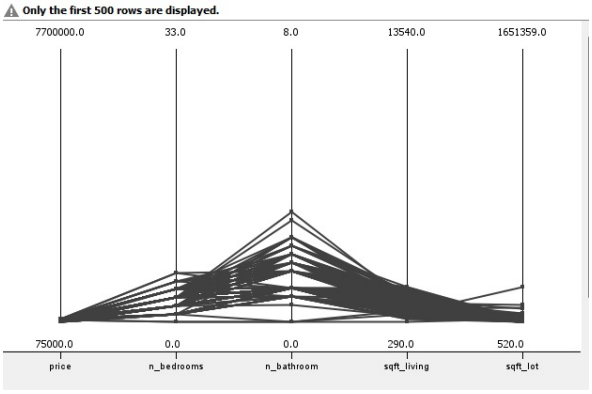

Fig. 6. 500 case meno costose.

Le case più costose oltre ad un prezzo maggiore presentano anche un numero di elementi maggiori al loro interno come: un maggior numero di stanze da letto, di bagni, una metratura totale degli ambienti interni dell'immobile e una metratura dello spazio esterno dell'immobile sempre maggiore rispetto a quelle meno costose.

3. PREPROCESSING

A. Eliminazione Outliers

Abbiamo inizialmente deciso di applicare un'eliminazione degli outliers filtrando le istanze che ricadessero all'infuori del range interquartile per ogni attributo. Questa procedura avrebbe comportato l'eliminazione di circa 4000 istanze, causando una perdita di informazioni rilevanti per la nostra analisi successiva. Abbiamo per questo motivo deciso di eliminare manualmente solo gli outlier più rilevanti. Più precisamente abbiamo agito in questo modo: il nodo "sorted" ci è risultato utile in più occasioni permettendoci di osservare il dataset in ordine ascendente o discendente per valutare le istanze che si distaccavano di più per quanto riguarda i valori. In particolare, abbiamo deciso di eliminare l'istanza con 33 bagni, quella con la metratura dello spazio esterno dell'immobile pari ad un valore di 1654000 e le tre case con il prezzo più alto con valori molto superiori al valore più alto non considerato outlier. Infine, abbiamo deciso di eliminare tutte le istanze con numero di bagni e di camere da letto pari a 0 perché questi valori creavano problemi durante l'applicazione della trasformazione logaritmica di queste variabili. Abbiamo scelto dunque di applicare questa eliminazione, per quanto riguarda le variabili bedrooms e bathrooms, per due motivi: da un lato abbiamo risolto il problema della forma logaritmica, il quale ci avrebbe restituito un errore, ottenendo appunto "infinito" come valore a seguito della trasformazione, dall'altro in quanto, non avendo bagni nè camere, abbiamo considerato le case come immobili non abitativi.

B. Trasformazione Attributi

Successivamente alla rimozione degli outliers abbiamo utilizzato un metanodo in cui, tramite il nodo "rule engine" abbiamo trasformato le features cambiando la tipologia delle variabili nella forma corretta per utilizzarle in modo appropriato nella nostra analisi successiva. In particolare, abbiamo trasformato le variabili *waterfront*, *center* (variabile creata a partire da *zipcode*, la feature che ci ha permesso di classificare e distinguere le case più in centro, le quali potrebbero essere quelle che presentano prezzi abbastanza sopra la media, rispetto a quelle più in periferia), *basement* e *restoration* in variabili "dummy" e mediante il nodo "column filter" abbiamo eliminato le stesse variabili che però non avevano ancora subito la trasformazione. Abbiamo poi aggiunto il nodo "Table to R" al fine di creare un raggruppamento della variabile *grade*, la quale inizialmente presentava valori compresi tra 1 e 13, in intervalli denominati: "very low", "low", "medium", "high". Si può fare un'ulteriore considerazione per quanto riguarda le variabili *sqft_basement* e *yr_renovated*. Queste ultime presentano rispettivamente 13126 e 20780 istanze con valore 0, questo sta ad indicare un'ingente presenza di immobili senza seminterrato o che non sono mai stati ristrutturati. Per sfruttare più efficacemente questa informazione, abbiamo deciso di trasformare questi due attributi in variabili binarie.

C. Correlazione

Utilizzando ora il nodo "Linear correlation" abbiamo deciso di risolvere il problema delle variabili che presentano una cor-

relazione troppo alta tra di loro. Abbiamo deciso di inserire all'interno del nodo le variabili numeriche e dummy e controllare quindi la correlazione presente tra di loro tramite una matrice di correlazione ed eliminare le variabili che presentano una correlazione superiore a 0.8, cercando di risolvere, in questo modo, il problema della multicollinearità fra le variabili. Oltre a questo, abbiamo considerato opportuno rimuovere anche le variabili *id*, *date* e *zipcode* perché inutili alla predizione. Qui di seguito è riportata la matrice di correlazione di Pearson:

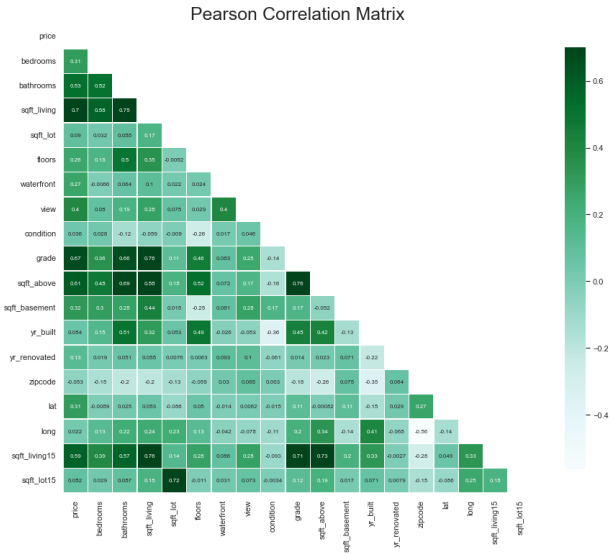

Fig. 7. Matrice di correlazione di Pearson.

4. MODELLI

Per ottenere le stime del prezzo di un immobile nell'area di interesse, sono stati implementati molteplici modelli di regressione, dalla complessità crescente, al fine di individuare quelli capaci di risolvere in maniera migliore il compito di previsione. Di seguito una lista dei modelli di apprendimento supervisionato e dei relativi nodi sfruttati:

- **Simple Linear Regression**: il nodo utilizzato, basato su Weka 3.6, accetta in input solo attributi numerici e per predire la variabile target seleziona come regressore l'attributo che permette al modello di raggiungere il minor mean squared error.

- **Multiple Linear Regression**: implementata usando un nodo base di Knime, riceve in input una trasformazione logaritmica della variabile target. La funzione logaritmica è stata applicata alla variabile price per ovviare alla violazione dell'ipotesi, alla base del modello di regressione lineare, di normalità dei residui [7]. Tale trasformazione è stata effettuata anche precedentemente all'esecuzione della regressione lineare semplice.

- **Polynomial Linear Regression**: inserita nel workflow tramite un nodo base di Knime, è un particolare tipo di regressione lineare attraverso il quale la relazione tra le variabili esplicative e la variabile target price è adattata mediante un'equazione curvilinea. L'implementazione di tale modello è da porre a confronto con quella della regressione lineare multipla, allo scopo di individuare quale dei due modelli è capace di derivare una relazione più adeguata tra variabili indipendenti e variabile dipendente. Considerando la multicollinearità introdotta nel modello a causa della presenza di termini polinomiali, abbiamo proceduto alla standardizzazione degli attributi esplicativi. Questa trasformazione è essenziale per prevenire l'offuscamento della significatività dei termini del modello, nonché la generazione di coefficienti imprecisi [8]. Dal momento che il nodo di apprendimento utilizzato non accetta variabili categoriche nella loro forma originale, è stato necessario convertire questi attributi in variabili binarie. Per conseguire ciò, abbiamo individuato la tecnica dell'one-hot encoding, implementata dal nodo Knime One To Many. A seguito di questa operazione, per ogni n livello assunto da un attributo categorico, n colonne sono state aggiunte al dataset di input per il nodo di regressione polinomiale. Infine, abbiamo deciso di fissare il parametro di grado massimo del polinomio a 2. La scelta è stata dettata dalla consapevolezza del rischio di overfitting di questo tipo di modello all'aumentare del grado del polinomio [9]. Per prevenire la particolare sensibilità di questo algoritmo agli outliers, questi ultimi sono stati opportunamente trattati nella fase di pre-processing, come illustrato in precedenza.

- **Simple Regression Tree**: questo primo modello ad albero è stato implementato tramite un nodo, contenuto nel KNIME Ensemble Learning Wrappers, che addestra un singolo albero di regressione applicando delle semplificazioni, tra le quali l'assenza di pruning. In ogni nodo foglia viene restituito, come valore stimato della variabile target, il valore medio di quest'ultima per gli attributi all'interno di quel nodo foglia. Per gli attributi nominali dati in input al modello è stata selezionata l'opzione di split binario.

- **Tree Ensemble**: i metodi di ensemble combinano molteplici alberi decisionali per produrre performance predittive migliori di quelle ottenute utilizzando un singolo albero decisionale, generando predizioni costituite dalla media degli output dei singoli alberi [10]. Raramente, infatti, un singolo albero decisionale è capace di generalizzare bene dati sui quali non è stato addestrato. Questo modello di ensemble generico è stato inserito nel workflow tramite i nodi Tree Ensemble Learner e Tree Ensemble Predictor. Nella configurazione del primo, abbiamo impostato la configurazione di ciascun albero su di un sottoinsieme di righe (campionate casualmente) e di attributi diversi per ogni albero, procedimento che previene il fenomeno di overfitting.

- **Random Forest Regression**: un particolare tipo di metodo di ensemble, si caratterizza per essere una tecnica di Bagging, tale per cui tutti i calcoli all'interno di un albero sono eseguiti in parallelo e non vi è interazione tra gli alberi decisionali durante il loro processo di costruzione [11]. Il nodo utilizzato per implementare l'addestramento dei dati con questo modello di regressione è il Random Forest Learner (Regression). Rispetto al Tree Ensemble Learner, è caratterizzato da un minor numero di opzioni di configurazione.

- **Gradient Boosted Trees**: tende ad essere la più potente tra le tecniche di ensemble, nonostante l'interpretazione dei suoi risultati si riveli di maggiore complessità. Si basa su un approccio sequenziale: viene costruito in maniera incrementale un ensemble, addestrando ciascuna nuova istanza del modello in modo tale da enfatizzare le istanze del training set mal classificate dai modelli precedenti. In questa

maniera, un modello viene addestrato sulla base degli errori di un modello più debole, dove il modello più debole, in questo caso, è un singolo albero decisionale [12]. Sono stati usati i nodi Gradient Boosted Trees Learner (Regression) e Gradient Boosted Trees Predictor (Regression).

- **Multilayer Perceptron**: è un modello di regressione basato su reti di neuroni artificiali, ovvero una collezione di nodi connessi, ognuno dei quali riceve in input un segnale e, dopo averlo processato, lo propaga ai neuroni ad esso connessi. Al nodo utilizzato, basato su Weka 3.6, sono stati passati in input i valori normalizzati delle variabili esplicative. Questo perché la normalizzazione dei dati, generalmente, permette di velocizzare e rendere più flessibile l'addestramento del training set [13].

Ognuno di questi algoritmi è stato addestrato e successivamente validato utilizzando sia il metodo Holdout che la tecnica di Cross-validation. Per quanto riguarda il primo metodo di partizione del dataset, abbiamo assegnato in maniera casuale l'80% dei record al training test ed il restante 20% al test set; nella Cross-Validation, considerando la grandezza del dataset, abbiamo ritenuto appropriato fissare il numero di sottoinsieme disgiunti a 5. Per garantire la riproduzione delle stesse partizioni randomiche su ogni dispositivo, abbiamo impostato un seed pari a 200 in ogni nodo Partitioning e X-Partitioner. Salvo eccezioni precedentemente illustrate ed argomentate, tutti gli attributi disponibili nel dataset dopo la fase di preprocessing sono stati utilizzati.

5. VALUTAZIONE MODELLI

Per valutare la performance dei modelli appena descritti, confrontiamo i valori di due metriche di qualità ottenute, per ognuno di questi modelli, sui dati di test. Esprimeremo, cioè, un giudizio in merito a quanto ciascun modello di regressione sia in grado di utilizzare i valori degli attributi esplicativi al fine di predire i valori della variabile target per tutte quelle istanze che non sono state usate per addestrare il modello stesso. Le metriche individuate sono il coefficiente di determinazione corretto R2adj e il Root Mean Squared Error (RMSE).

- R^2_{adj} : indica la frazione di varianza spiegata nei dati, cioè la proporzione della variabilità dell'attributo target Y spiegata dalla relazione tra Y e gli attributi esplicativi. Ha un range da 0 a 1 e maggiore è il suo valore, migliore è l'adeguatezza del modello ai dati. L'adjusted R2 è stato preferito al coefficiente di determinazione non corretto a fronte della costruzione di modelli con un numero diverso di variabili indipendenti. Esso è, appunto, aggiustato per il numero di predittori nel modello, facendo sì che il coefficiente aumenti di valore solo quando l'aggiunta di una nuova variabile è responsabile di un miglioramento del modello più di quanto ci si aspetti per solo caso.

$$R^2_{adj} = 1 - \frac{\sum_{i=1}^{n}(y_i - y_{mean})^2}{\sum_{i=1}^{n}(y_i - y_{i_{predicted}})}$$

dove:

  - y_{mean} indica la loro media;
  - $y_{i_{predicted}}$ indica il valore osservato per l'i-esima osservazione;

- **Root Mean Squared Error** : è la deviazione standard dei residui e indica quanto in media i valori previsti della variabile target sono distanti dai corrispondenti valori realmente osservati. Rispetto all'errore quadratico medio (Mean Squared Error), è misurato con la stessa unità della variabile target. Minore è il suo valore, maggiore è l'adattamento del modello ai dati.

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_{i_{predicted}} - y_i)^2}$$

dove:

  - $y_{i_{predicted}}$ indica il valore predetto per l'i-esima osservazione;
  - y_i indica il valore osservato per l'i-esima osservazione;
  - n rappresenta la grandezza del campione.

In generale, confrontando per ogni modello i valori dell'adjusted R^2 tra metodo Holdout e Cross Validation, è possibile osservare come questa metrica risulti in media sempre leggermente minore negli insiemi di test ottenuti tramite il secondo metodo. Il modello per cui la discrepanza tra i due valori di R2adj risulta essere maggiore è il Multilayer Perceptron, dove, a seguito del partizionamento tramite metodo Holdout, si può affermare che l'83% della variabilità del prezzo può essere spiegata dalla sua relazione con gli attributi esplicativi selezionati, a fronte di un 72% ottenuto mediante una Cross Validation di 5 folds. In termini assoluti, a seguito della nostra analisi, il modello capace di spiegare la più grande proporzione di variabilità nella variabile price è il Gradient Boosted Trees con metodo Holdout, contraddistinto da un R2adj pari a 0.858. D'altra parte, il modello con R2adj minore è quello di regressione lineare semplice, avente per regressore la metratura degli spazi interni dell'immobile (*sqft_living*). È proprio nei modelli di regressione lineare che l'R2adj registra i suoi valori più bassi, discostandosi da quelli ottenuti nel Multilayer Perceptron Regressor e nei modelli di regressione ad albero. Inoltre, è interessante notare come, proprio in questi ultimi, vi sia un incremento del coefficiente di determinazione all'aumentare della complessità del modello. È bene sottolineare, tuttavia, che valori alti di R2adj non sono sempre un bene, in quanto potrebbero celare una moltitudine di problemi che risiedono nell'implementazione del modello. Allo stesso modo, anche le misure di RMSE risultano essere migliori nel test set risultante dalla partizione tramite Holdout. Questo particolare pattern nei risultati potrebbe trovare le sue motivazioni nel fatto che i modelli sono sensibili nei confronti dei dati utilizzati per addestrarli e per poi testarli: mentre nella Cross Validation i modelli vengono, in un certo senso, testati su tutti i dati, motivo per cui questa tecnica viene riconosciuta come delle più robuste e che ci fornisce metriche più vicine a quelle reali, nel metodo Holdout le istanze sulle quali le performance vengono validate sono un costante 20% dell'intero dataset. È possibile, pertanto, che queste osservazioni corrispondano ad istanze i cui valori, per caso, si adeguano meglio ad i modelli di quanto facciano i dati di training.

Row ID	R^2	mean absolute error	mean squared error	root mean sq...	mean signed ...	mean absolut...	adjusted R^2
Simple Linear Regression Holdout	0.49	0.131	0.026	0.162	-0.003	0.023	0.489
Simple Linear Regression Corss validation	0.482	0.133	0.027	0.164	-0	0.024	0.481
Multiple Linear Regression Holdout R	0.732	0.09	0.014	0.117	-0.001	0.016	0.732
Multiple Linear Regression Crossvalidation R	0.723	0.091	0.014	0.12	0	0.016	0.723
Polynomial Regression Holdout	0.656	139,360.134	42,615,549,926.816	206,435.341	-82,435.22	0.277	0.653
Polynomial Regression Cross validation	0.614	150,742.317	49,696,643,394.335	222,927.44	-53,910.147	0.324	0.613
Simple Regression Tree Holdout	0.749	100,733.754	31,037,726,332.11	176,175.272	1,731.209	0.183	0.749
Simple Regression Tree Cross validation	0.727	103,321.932	35,102,932,289.185	187,357.765	623.733	0.19	0.727
Tree Ensemble Holdout	0.828	86,629.923	21,338,180,139.244	146,075.94	-351.81	0.169	0.827
Tree Ensemble Cross validation	0.81	89,458.421	24,415,058,408.824	156,253.187	379.672	0.173	0.81
Random Forest Holdout	0.837	83,233.537	20,215,122,300.978	142,179.894	370.803	0.16	0.836
Random Forest Cross validation	0.817	86,732.935	23,556,185,681.658	153,480.245	604.758	0.167	0.817
Gradient Boosted Trees Holdout	0.858	78,635.36	17,560,772,031.229	132,517.063	-7,624.337	0.145	0.858
Gradient Noosted Trees Cross validation	0.855	79,456.372	18,601,777,286.557	136,388.333	-6,947.064	0.148	0.855
Multilayer Perceptron Holdout	0.833	0.017	0.001	0.026	-0.006	NaN	0.832
Multilayer Perceptron Cross validation	0.738	0.018	0.001	0.033	-0.001	NaN	0.737

Fig. 8. Risultati dei diversi modelli di regressione.

6. CLUSTERING

Per quanto riguarda i modelli di clustering, sono state implementate diverse metodologie, utilizzando algoritmi di K-means e K-medoids e andando a valutarne l'efficacia con il coefficiente di silhouette e lo scorer di entropia. Gli algoritmi di clustering raggruppano gli elementi sulla base della loro distanza reciproca, quindi l'appartenenza o meno ad un insieme dipende da quanto l'elemento preso in esame è distante dall'insieme stesso. Quello di K-means, come K-medoids, è un algoritmo di analisi dei gruppi partizionale che permette di suddividere un insieme di istanze in k gruppi sulla base dei loro attributi, avendo in input n oggetti. Entrambi gli algoritmi cercano di minimizzare l'errore quadratico medio, la distanza tra punti di un cluster e il punto designato per esserne il centro. Nel nostro caso le variabili utilizzate sono state le seguenti: *price*, *n-bedrooms*, *n-bathrooms*, *sqft_living*, *sqft_lot*, *n_floors*, *waterfront*, *basement*, *restoration*, *centre*,*lat* e *long*.

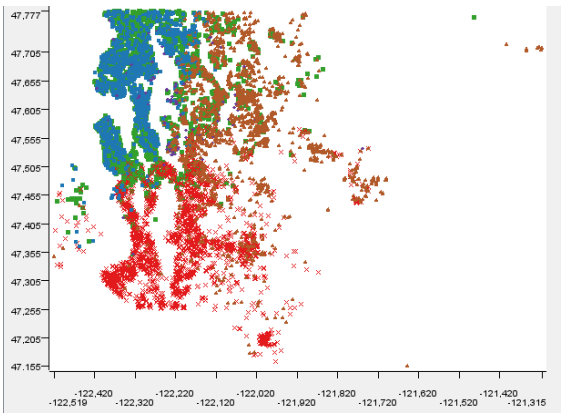

Fig. 9. K-means 1: cluster vs price (prime 10.000 righe).

Per quanto riguarda il K-means sono stati eseguiti 2 differenti procedimenti: Per il primo metodo (Figura 9) è stato deciso di dividere il dataset in 2 partizioni, rispettivamente 60% e 40%, utilizzando la prima divisione per creare e salvare un modello k-means con 5 cluster, standardizzando inizialmente i dati per il modello. La seconda divisione del dataset, invece, è stata utilizzata per applicare e quindi testare il modello di cluster creato su nuovi dati, valutandone le performance mediante l'utilizzo del coefficiente di silhouette e del valore di entropia.

Nel secondo test (Figura 10, in cui sono stati creati 9 cluster, è stato usato lo scorer di entropia per migliorare in corso l'assegnazione dei cluster, assegnando come centroidi di inizializzazione le prime k righe e con un numero di iterazioni massimo pari a 99.

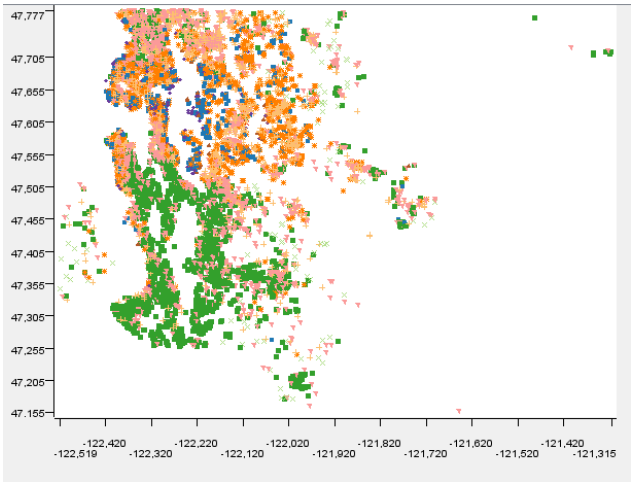

Fig. 10. K-means 2: sqft_living vs price (prime 10.000 righe).

Infine, l'algoritmo di k-medoids (Figura 11 viene applicato

alla tabella di input che, partendo da un inizializzazione casuale dei medoidi, esegue iterativamente una ricerca esauriente sui dati di input, determinando il costo per lo scambio di qualsiasi medoide con qualsiasi riga di dati di input e sostituendo il medoide con la riga di dati che riduce maggiormente il costo. Nel nostro caso i costi sono stati determinati utilizzando una matrice di distanza precalcolata (porta 0 del nodo) e una misura di distanza connessa (porta 1). Si può notare come l'ultimo clustering raggruppi maggiormente le case con un prezzo superiore alla media e situate prevalentemente nella zona più vicina alla città di Seattle.

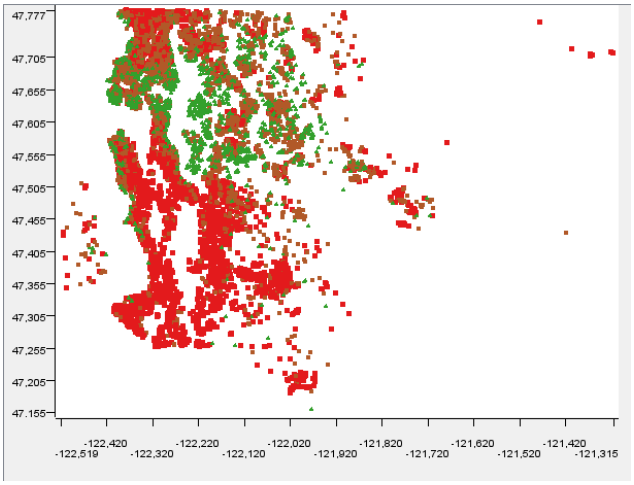

Fig. 11. K-Medoids 1: rappresentazione su mappa (prime 10.000 righe).

7. CONCLUSIONI

Come argomentato nell'introduzione, il compito di prevedere il prezzo di un'immobile a partire da alcune delle sue caratteristiche quantitative misurabili, categoriche e binarie, sembra sia stato portato a termine in modo soddisfacente. Le modifiche apportate nella fase di preprocessing si sono rivelate cruciali nel conseguimento dei risultati ottenuti. A seguito dell'applicazione di vari modelli di regressione, ognuno di essi con due diversi metodi di partizione, è emerso che le migliori performance di previsione per il nostro dataset sono raggiunte dai modelli di regressione ad albero e di regressione del tipo Multilayer Perceptron, tramite una partizione Holdout in cui l'80% delle istanze sono state assegnate al Training Set ed il restante 20% al Test Set. Per migliorare ulteriormente le predizioni, una futura analisi potrebbe concentrarsi sulla selezione degli attributi più rilevanti per il nostro scopo, cosa che gioverebbe anche verso una riduzione della dimensionalità del dataset. Infine, si potrebbe curare e approfondire maggiormente la scelta dei parametri durante la configurazione di alcuni dei modelli implementati. Potenzialmente, infatti, esistono numerose altre vie da intraprendere per estrapolare informazioni interessanti in questo dataset, sia prendendo in considerazione una combinazione di variabili differenti, sia ponendosi delle domande di ricerca alternative.

REFERENCES

1. https://www.kaggle.com/harlfoxem/housesalesprediction
2. https://en.wikipedia.org/wiki/Seattle
3. "Seattle no longer America's fastest-growing big city", https://www.seattletimes.com/seattle-news/data/seattle-no-longer-americas-fastest-growing-big-city/
4. https://www.knime.com/
5. https://www.r-project.org/
6. https://github.com/5526-michelegit/machine-learning/blob/main/KCHOUSE_analyses-exploration_preprocessing.ipynb
7. https://medium.com/swlh/log-transformations-in-linear-regression-the-basics-95bc79c1ad35
8. "When Do You Need to Standardize the Variables in a Regression Model?", Jim Frost, https://statisticsbyjim.com/regression/standardize-variables-regression/
9. https://www.upgrad.com/blog/polynomial-regression/
10. "Making Tree Ensembles Interpretable: A Bayesian Model Selection Approach" Hara S., Hayashi K. March 1 2017
11. https://cnvrg.io/random-forest-regression/
12. https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
13. https://towardsdatascience.com/why-data-should-be-normalized-before-training-a-neural-network-c626b7f66c7d