Previsione mercato immobiliare nell'area di New Taipei
Report per l'Esame di Fondameti di Machine Learning

Giulio Barabino

146829
Corso di Laurea in Ingegneria Informatica - Sede di Mantova
260553@studenti.unimore.it

Abstract

Progetto per la previsione dell'andamento del mercato immobiliare nella zona Sindian di New Taipei, Taiwan. Verrà effettuata un'esplorazione dati iniziale e una prima evaluation su alcuni modelli di regressione; si sceglierà il migliore tra essi e ne saranno modificati gli iperparametri così da ottenere le performance migliori. Verrà, infine, fatto un confronto con lo stesso modello senza il tuning degli iperparametri.

1 Introduzione

L'andamento del mercato immobiliare è da sempre preso come riferimento per task di regressione. Diversi sono i dataset disponibili di questo tipo. Il caso preso a riferimento è relativo all'anno 2012-2013 e riguarda i prezzi (per unit/area) dell'area Xindian della città di New Taipei, Taiwan. Il dataset consta di 414 samples e 7 feature:

Feature	**Dtype**
transaction date	float64
house age	float64
distance to the nearest MRT station	float64
number of convenience stores	int64
latitude	float64
longitude	float64
house price of unit area	float64

Risulta, quindi, essere un dataset ridotto, con nessuna feature categorica e tutte feature numeriche. La house price per unit area è espressa in dollari taiwanesi su area quadrata.

Le feature presentano una denominazione contrassegnata in questo modo: *X1 transaction date*, motivo per il quale ho apportato immediatamente una modifica alle stesse rimuovendo il simbolo *"X"* e il numero che lo segue per una migliore comprensione.

2 Analisi dei dati

Le feature indicano: la data di transazione della compravendita dell'immobile (tutte datate tra 2012 e 2013), l'età della casa (minore è meglio), la distanza dalla più vicina stazione ferroviaria di trasporto rapido presente nel paese (minore è meglio), il numero di market nelle vicinanze (maggiore è meglio), latitudine e longitudine e la feature da predire, cioè il prezzo per unit/area.

L'EDA ha inizio con il controllo delle presenza di valori NaN all'interno del dataset. Fortunatamente non ne risultano presenti. Una prima descrizione del dataset, però, ci dà un'indicazione sui valori delle feature quali media, deviazione standard e valore massimo della stessa e si può

notare subito come il valore massimo della feature da predire sia eccessivamente alto, infatti:

Mean	**STD**	**MAX**
37.980193	13.606488	117.500000

Dimostrato anche dalla distribuzione dei valori della feature:

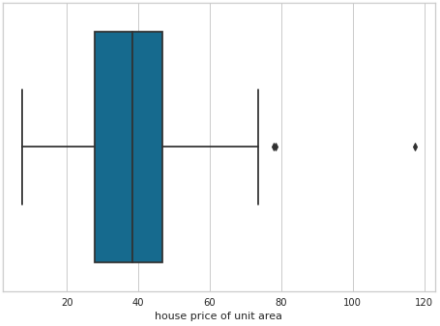

Figure 1: Target distribution

Decido così di proseguire con l'analisi di questa feature per appurare che tale sample non sia un outlier. Per fare ciò verifico che i valori delle altre feature per quel sample siano anch'essi al di sopra della media. Se così è per la vicinanza alla stazione MRT, lo stesso non si può dire per il numero di convenience store, pari a 1 (ben al di sotto delle media).

Proseguo quindi con il plot delle feature *latitude* e *longitude* rispetto alla variabile attesa, così da avere una rappresentazione visuale delle aree col maggior costo.

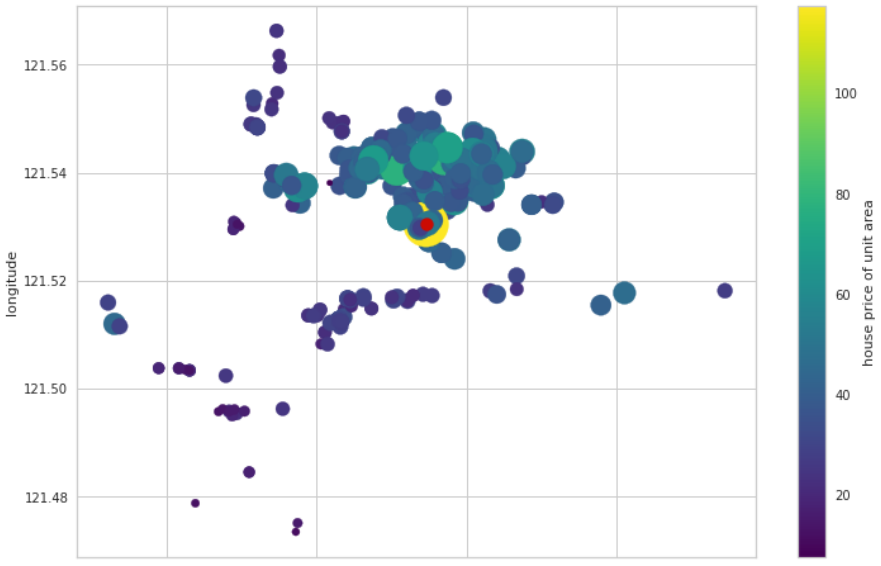

Figure 2: Latitude and longitude against y

Risulta chiaro dall'immagine come il sample non sia un outlier perchè si trova nella zona geografica con il costo degli immobili maggiore di tutta l'area considerata. Non opero quindi nessuna eliminazione dal dataset e assumo che si tratti di un errore di battitura o di qualche sfortunato acquirente.

2.1 Feature Selection

Definisco la matrice di correlazione delle feature e le feature meno correlate risultano essere *house age* e *transaction date*. Contrariamente a quanto è possibile pensare l'età della casa non influenza il prezzo finale.

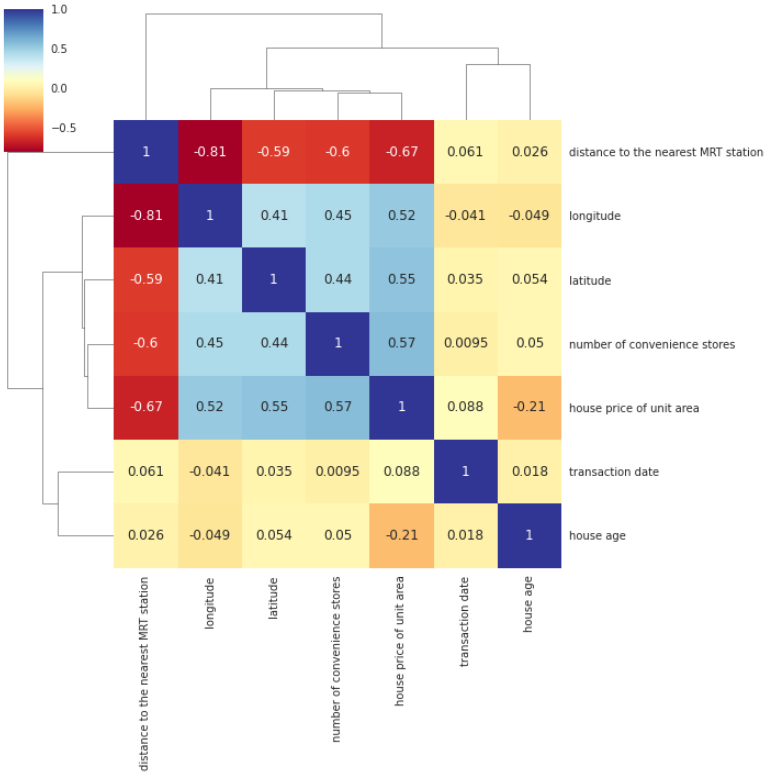

Figure 3: Correlation Matrix

Non utilizzo *Wrapper methods* per la selezione delle feature, ma opero un filtraggio sulle feature e tengo solo quelle con uno score di correlazione > 0.5, eliminando le altre.

2.2 Pre-Processing

Utilizzo una funzione di *shuffle* del dataset per assicurarmi che i dati siano, per quanto possibile, indipendenti e identicamente distribuiti. Suddivido, infine, il dataset in training e test set secondo il paradigma 80-20. Successivamente applico anche uno *scaling* alle feature, del training e del test set, sottraendo la media, del training set, e dividendo per la varianza, sempre del training set.

3 Model Selection

La fase di scelta del modello parte da una semplice analisi dei residui tramite il modello di regressione lineare a cui diamo in pasto i subset di training, prima, e di test, dopo. Già da questa prima analisi capiamo che c'è linearità nei dati.

Definisco, quindi, una lista di modelli che utilizzeremo per calcolare un primo score così da poterli confrontare e poter scegliere quello che performa meglio. I modelli utilizzati sono i seguenti:

Modelli

- Linear Regression

- K-Neighbors Regressor
- Decision Tree Regressor
- SVR

Scelgo di valutare i modelli in base a due metriche: *NRMSE* e *R2*. Il primo grazie al quadrato considera allo stesso modo errori positivi e negativi e perchè minimizza gli errori degli outlier. Il secondo fornisce un'indicazione delle performance in scala [0-1] così da avere un riscontro immediato. Utilizzo il metodo

```
cross_val_score
```

fornito dalle librerie per ML di Python. Tale funzione effettua una *cross-validation*, su un numero di fold deciso dall'utente, e restituisce un array di score secondo la metrica decisa. Utilizzo tale metodo sul training set e ottengo i seguenti risultati:

Model	**R2**	**NRMSE**
Linear Regression	0.53771	-8.36062
K-Neighbors Regressor	0.66364	-7.10076
Decision Tree Regressor	0.58006	-7.88130
SVR	0.54346	-8.38052

Come si può vedere dalla tabella il modello con le performance migliori risulta essere *KNN*. Gli score in tabella sono la media dei valori restituiti. Nonostante nel codice sia presente anche lo score migliore ottenuto, esso non viene considerato ai fini della valutazione perchè la cross-validation opera, a partire da un dataset già di suo piccolo, una validazione su una trentina di elementi, statisticamente poco rilevanti.

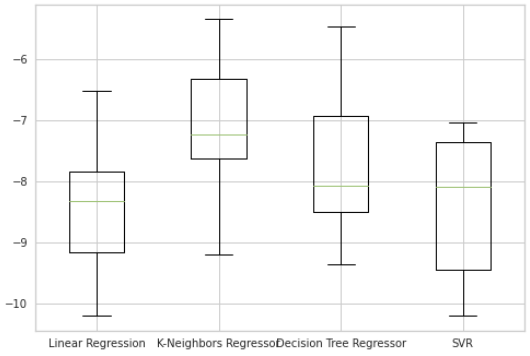

Figure 4: NRMSE estimators comparison

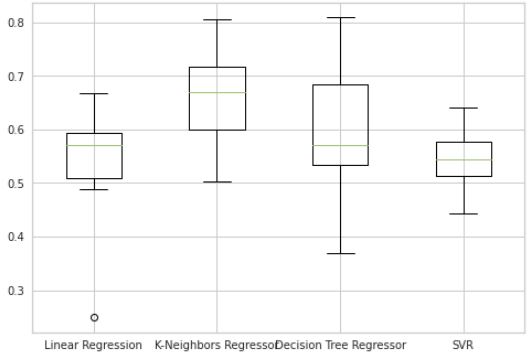

Figure 5: R2 estimators comparison

4 Hyperparameter Tuning

Scelto come modello il *K-Neighbors Regressor* procedo con il tuning degli iperparametri al fine di ottenere le prestazioni migliori. Per fare ciò mi servo del metodo

```
GridSearchCV
```

il quale effettua una cross validation, sempre su 10 folds, per ogni combinazione di iperparametro presente all'interno della griglia che ho definito. Gli iperparametri definiscono:

- L'algoritmo usato per computare i vicini
- Il numero di vicini da considerare

- Il peso da assegnare ai vicini

La grid incorpora tutte le possibilità così da provare tutte le combinazioni possibili e trovare la migliore, la quale risulta essere:

```
{'algorithm': 'brute', 'n_neighbors': 13, 'weights': 'distance'}
```

Con un NRMSE di -6.82177, con un delta di 0.279 rispetto al modello a parametri predefiniti.

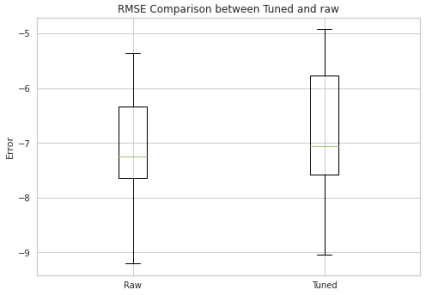

Figure 6: Comparison

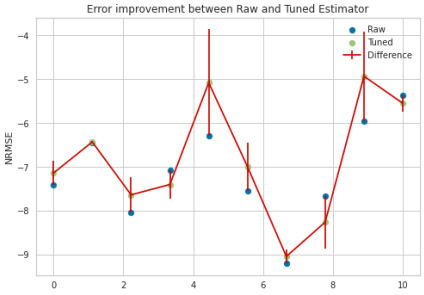

Figure 7: Improvement

Di seguito riporto anche le curve di training e di validation ottenute con le diverse combinazioni di iperparametri.

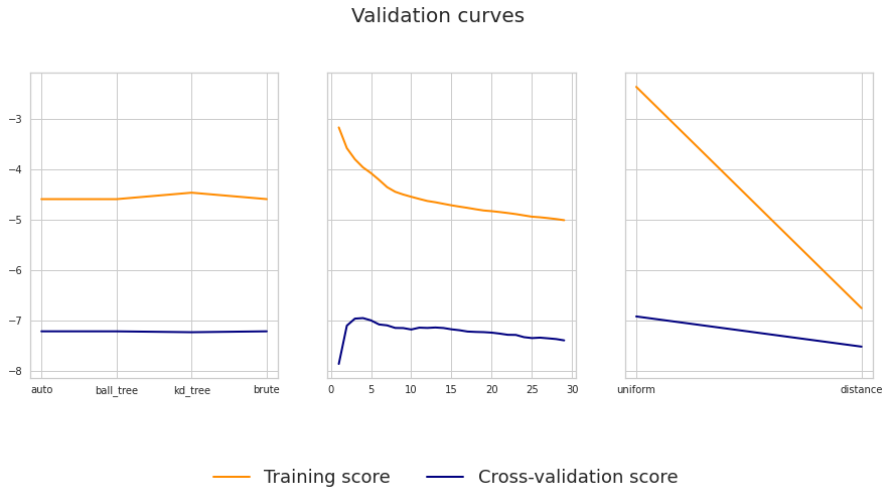

Figure 8: Validation curves

Dalla figura capisco che l'algoritmo, contrariamente a quanto è possibile pensare, non influenza molto l'errore. Diversamente, il parametro k ha un andamento decrescente nella fase di training e un, almeno iniziale, andamento crescente nella fase di validation. Ciò significa che all'aumentare di k il modello è in grado di generalizzare meglio. Risulta anche evidente l'overfitting presente in *k=1*. Lo sweet spot come si può notare anche graficamente si presenta con *k=13*, punto in cui si raggiunge il trade-off migliore tra errore in traning e errore in validation.

5 Testing

Concluso il tuning degli iperparametri procedo con la fase di testing e la predizione della variabile attesa in funzione dei dati del subset *X_test* utilizzando la combinazione di iperparametri appena trovata. Effettuo la predizione anche con un algoritmo di K-Nearest-Neighbors predefinito per

Model	NRMSE	MAE	R2
Tuned	-10.8398	5.7563	0.5797
Untuned	-11.4076	6.3072	0.5345

valutare gli effetti delle modifiche degli iperparametri e la differenza di errori tra i due estimator. Ottengo i seguenti risultati:

Il miglioramento non è sostanziale ma è presente, come è indicato da tutte le metriche. Il modello è sicuramente più aderente al dataset e consente di ottenere previsioni migliori del 4% circa. Di seguito alcune immagini per graficare la differenza di errori e di predizioni tra i due modelli:

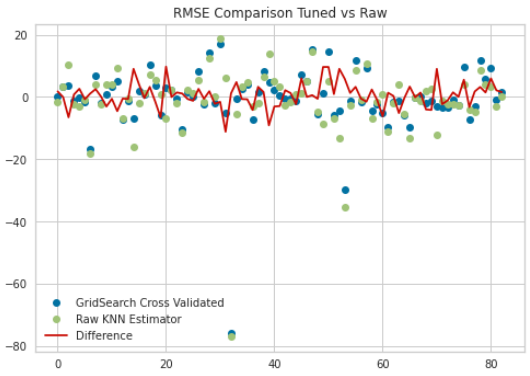

Figure 9: NRMSE Comparison

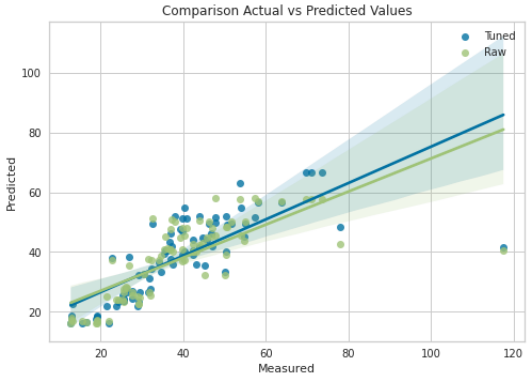

Figure 10: Actual vs Predicted

E il confronto tra le varie metriche di errore:

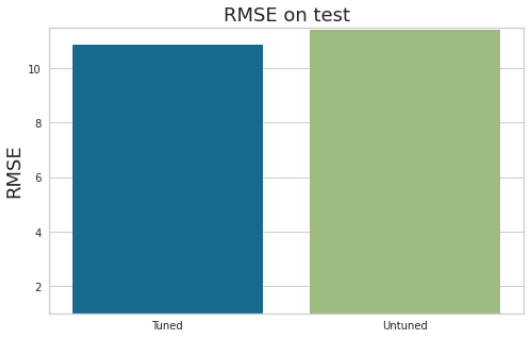

Figure 11: RMSE on Test

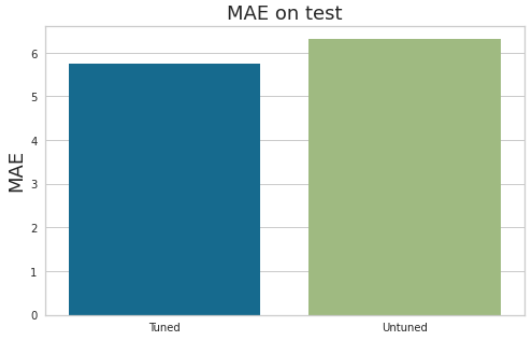

Figure 12: MAE on Test

5.1 Considerazioni finali

Il modello più semplice è risultato essere anche il migliore in fase di comparazione. Anche effettuando il tuning degli iperparametri della SVR non si sono ottenuti risultati migliori rispetto a KNN. Il codice relativo a tale gridsearch è stato omesso dalla versione finale.

Il miglioramento ottenuto non è netto, come già detto, e su un valore medio di 37.980193 unit/area, uno scarto di 5.7563 sicuramente non è trascurabile. La dimensione del dataset, però, probabilmente non consente altro margine di miglioramento utilizzando i modelli scelti.

Eliminando l'outlier di cui ho discusso nella sezione 2 i modelli risultavano avere performance più alte, ma per una questione di aderenza alla realtà e per i motivi già discussi ho ritenuto opportuno non rimuoverlo.

Ulteriori miglioramenti si sarebbero potuti ottenere con una grid randomica, ma lo spazio di manovra dato dagli iperparametri da me definiti credo sia sufficiente.

References

[1] Yeh, I. C., & Hsu, T. K. (2018). *Building real estate valuation models with comparative approach through case-based reasoning.* Applied Soft Computing, 65, 260-271.

[2] Line Ton That, *Real estate valuation in Xindian district, New Taipei, Taiwan*, Github.