Housing price prediction in King County: A Machine learning Approach

Team

Zihang Cai 11319479 Shihao Tong 11318294 Danni Weng 11304053

1. Introduction

1.0 Motivation

The King County dataset is very popular on Kaggle with more than 1100 notebooks trying to build machine learning models in order to predict local housing prices [1]. As can be seen, many of the users are using tremendously complicated and fancy models and techniques for this somewhat simple problem for example the artificial neural network (ANN) [2], deep neural network (DNN) [3] and also some other ensemble models. Concerning the variance bias trade-off, especially for the neural networks, the increasing number of parameters to fit can easily lead to overfitting. One of our purposes is to find a relatively simpler model with fewer necessary parameters to lift up the generalization power. One other thing we also notice is that everyone cleaned the data by removing the outlier before training models. However, the way people determine what points are outliers varies a lot. Most of them plot the distribution of the response variable, which is the housing price, and take those values surpassing a threshold as outliers. We doubt if this is legit. Thus our second purpose is to see how the outliers actually affect the performance of the models. We achieve this by running our model on the raw dataset and also the one with the outlier removed.

1.1 Dataset

The data come from the Kaggle site and contain the sales prices of 21,613 homes sold in Washington State between May 2014 and May 2015 in King County, which includes the city of Seattle, among other places. This dataset also contains 19 variables measuring different characteristics of homes. These correlation of variables are as below:

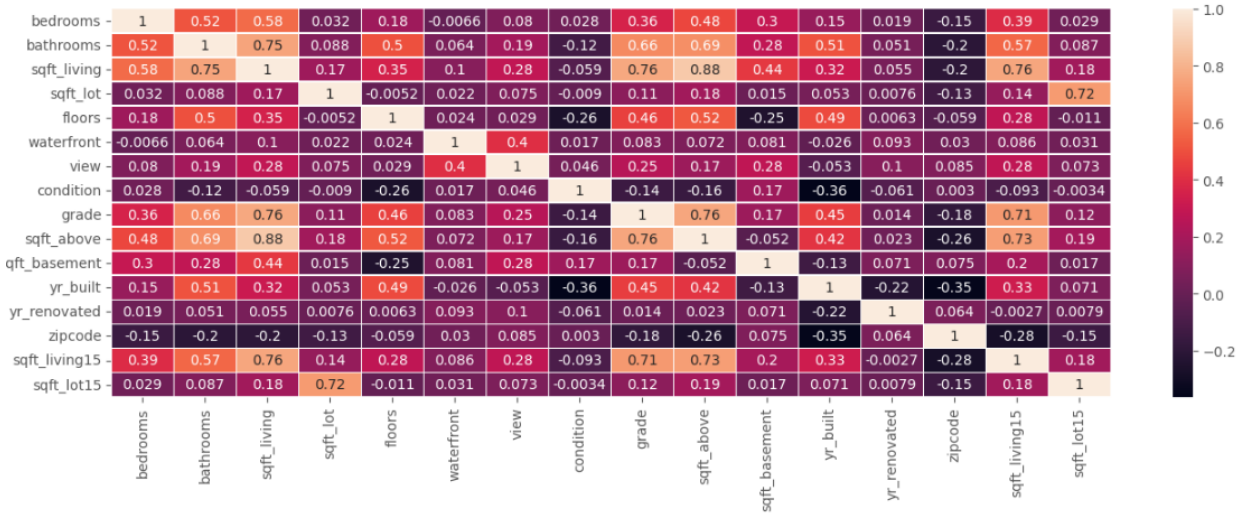

Figure 1: Correlation Matrix

As we expect, some variables are highly correlated, we don't want to include two highly correlated variables in our model, we select eight we believe most representative variables as below:

* `price`: selling price of the property (variable of interest)
* `bedrooms`: number of rooms
* `bathrooms`: number of bathrooms
* `yr_built`: year of construction
* `sqft_lot`: area in square feet (entire lot)
* `lat`: latitude
* `long`: longitude
* `grade`: Overall score for the quality of the renovations and building materials (score from 1 to 13 - a higher score indicates better quality materials used). This score is calculated according to the King's count scoring system.

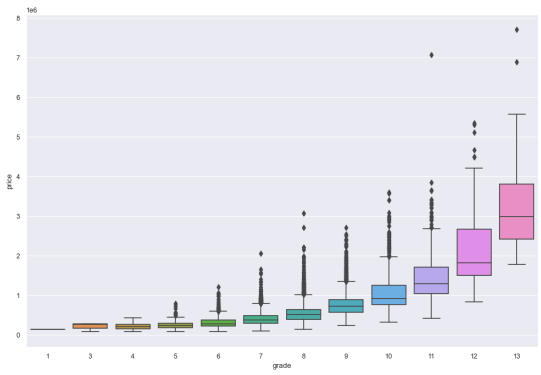

Figure 2: Housing Price v.s. Grade of House

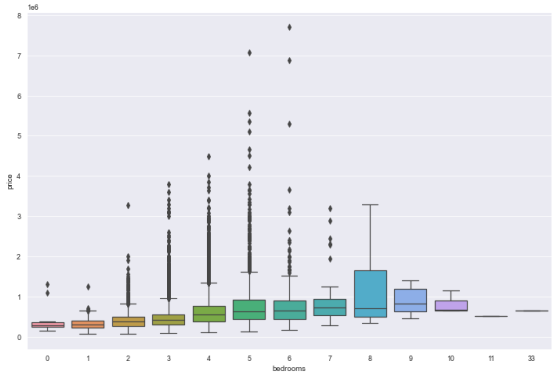

Figure 3: Bedroom distribution regarding price

Figure 2 implies that the price change is quite sensitive to the quality of the house(grade), as the higher the grade, the higher the quality of design and construction, increasing the

value. Especially when the grade is greater than 10. Meanwhile, we can see that there are outliers in our data, especially at grade 11 and 13.

Apparently figure 3 shows that there are outliers as well. We could easily notice that our dataset includes an extreme house with 33 bedrooms. Meanwhile, there are houses without any bedrooms. We are definitely going to pay some attention to these outliers in order to meet our second purpose.

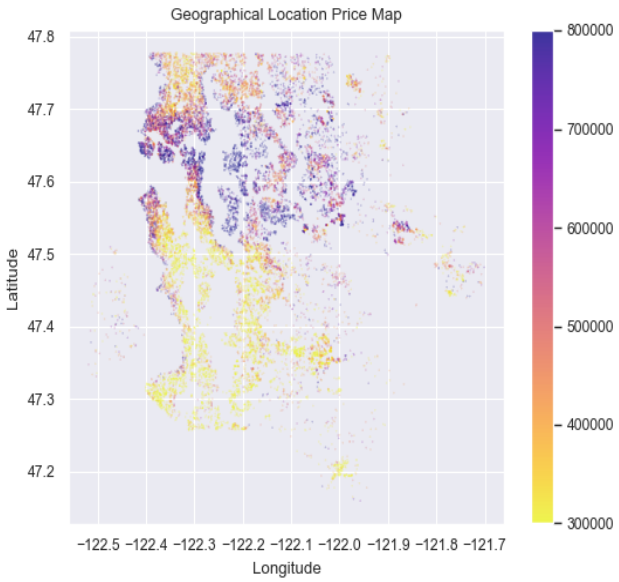

Figure 4: Geographical distribution of housing price in King County

The heatmap of figure 4 visualizes the house price by location, which indicates that the Lake Union area suffers from relatively higher house prices than the rest of Seattle. To reduce noise, we use a color scale to highlight the price range of 300k to 800k, thus we won't get the visualization distorted by neither cheap nor expensive houses.

1.2 Remove Outliers

As mentioned in the motivation, one of our targets is to compare the performance of models on the raw dataset and also the outlier removed dataset to see the effect of the outlier. It is crucial to define what an outlier is. Specifically, we define a data to be an outlier if it falls beyond the upper and lower fence of the set it belongs to which is

$$x \notin [Q_3 + 1.52 * IQR, Q_1 - 1.52 * IQR]\}$$

where Q1 is the 25th quantile, Q3 is the 75th quartile and IQR stands for the interquartile range which is

$$IQR = Q_3 - Q_1$$

Specifically in our case, we determine a *row,* which is a single recording, to be an outlier if one of its corresponding features is an outlier. This rule applies to price, bedroom and area in square feet. We only apply this to the numerical variables but not categorical ones.

2. Methods

In total, we ran 4 different models, excluding the simple linear regression which serves as a baseline for comparison, which are K-nearest neighbourhood, polynomial regression, random forest and random forest with XG Boost. Each of the above models goes through the training and testing phases on the raw dataset and outliers-omitted dataset once. Details of each model are discussed as follows.

2.0 Cross-Validation

In order to evaluate the quality of our model, we use cross-validation. We divide our dataset into a training set(80%) and a test subset(20%). We are going to use a training set to fit the model, and the test set to check the performance of the data.

2.1 Model Evaluation

We are going to use R^2 and Root Mean Squared Error(RMSE) to evaluate the performance of our model, and further as criteria to select the best model. Low R^2 and RMSE are preferred.

2.2 Linear Regression

We will construct a linear regression with variables 'price', 'grade', 'bedrooms', 'bathrooms', 'sqft_lot', 'yr_built', 'lat', and 'long'. We will use this model as the benchmark to compare with other more advanced models. The residuals of the model are normally distributed. The benchmark performance is :

R-square: 0.6522
RMSE: 217153.6288

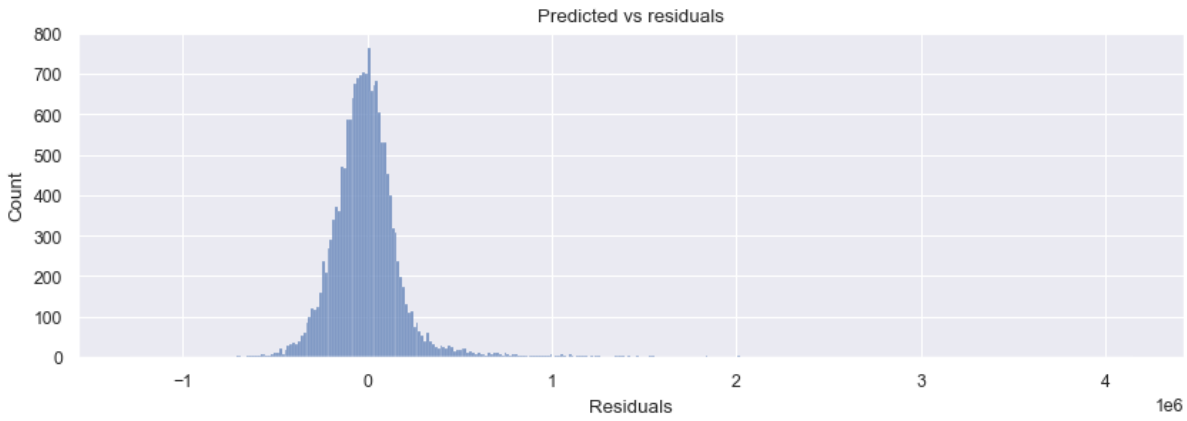

Figure 6: Histogram of Residuals

2.3 K Nearest Neighbour Regression

K-nearest neighbor (k-NN) is an algorithm that can be used for both regression and classification tasks. In the context of predicting housing prices, k-NN could be used as a regressor to predict the price of a house based on its relevant features. The KNN does not have a specific training phase since there is only hyper-parameter K to be fitted but no model parameters. Once the K is determined, we can use it to make predictions on new houses for which we only have the location and other attributes. The k-NN algorithm will find the "k" closest houses in the training set based on the distance metric and then use the mean of the k housing price as the prediction. KNN can be simple and useful but it is important to carefully evaluate its performance and compare it to other algorithms to ensure that it is the best choice for our specific problem.

The KNN model has three major components: the number of neighbors K, the distance metric and the variables used in the distance metric. The KNN has a very basic assumption: the closer the points, the more similar its character. The similarity is purely reflected by the distance metric and its associated features. In our model, instead of choosing K, the type of distance is also set as a hyperparameter to be determined. The type of distance metric we tried is the Minkowski Distance up to order 8 which is

$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{\frac{1}{p}}$$

When p = 1, Minkowski distance is Manhattan distance and when p = 2 we restore the common Euclidean distance. Tow other distances of interest are the Chebyshev distance which is

$$D_{\text{Chebyshev}}(x,y) := \max_i(|x_i - y_i|)$$

It is the asymptotic result as p goes to infinity in Minkowski distance, and the cos distance which is

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}.$$

Cosine distance logically makes sense since considering a line of condos in downtown with a high housing density. Those condos should have similar values regardless of the Euclidean distance since they are all located downtown. With all these metrics, we ran our analysis on two different sets of features. The first set of features only includes the geographical distance which are the longitude and latitude of the house. The second set of features are all the variables available as suggested in part 1.1.

2.4 Random Forest with XGBoost

Random forests are ensemble models, which means they are composed of multiple individual decision trees that work together to make predictions. The use of multiple decision trees can improve the performance of the model by reducing overfitting and improving the generalization of the predictions.

Decision trees are a type of machine learning algorithm that can be used for both regression and classification tasks. They work by creating a tree-like model of decisions, with each internal node representing a decision based on the value of an input feature, and each leaf node representing a prediction.

Random forests combine multiple decision trees by training each tree on a different subset of the training data. This is done by selecting a random sample of the data for each tree, and also a random sample of the input features for each split in the tree. This creates a diverse set of trees, which can then be combined to make a prediction. Additionally, random forests can handle large and high-dimensional datasets, which is often the case in housing price prediction, where there may be many different features that can affect the price of a property.

The random forests training algorithm uses a general technique called bootstrap aggregating, or bagging, for tree learners. Given a training set of data points with responses, bagging repeatedly selects a random sample with replacement from the training set and trains a tree on each sample. This process is repeated a set number of times, B. After training, unseen samples can be predicted by taking the average of the predictions from all the individual trees. The process is outlined as follows:
For $b = 1, ..., B$:

1. Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.

2. Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

$$\widehat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

Bagging involves repeatedly selecting random samples of the training data with replacement and training a tree on each sample. This process reduces the variance of the model, which means that it is less sensitive to noise in the training data. By averaging the predictions of many trees, the random forests algorithm is able to smooth out the effects of noise and improve the overall model's accuracy.

The estimate of the uncertainty of the prediction can be made as the standard deviation of the predictions from all the individual regression trees on x':

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x')-\widehat{f})^2}{B-1}}$$

B is the number of the trees (samples). The number of trees used in a random forests model typically ranges from a few hundred to several thousand, depending on the size and complexity of the training data.

XGBoost (eXtreme Gradient Boosting) is a type of gradient boosting algorithm, which means that it trains multiple "weak" learning models, such as decision trees, and combines them to create a single "strong" predictive model. XGBoost is known for its speed and performance, and it has several key features that make it an effective machine learning algorithm. These include its ability to handle missing values, its support for parallel processing, and its built-in regularization to prevent overfitting.

We will create a random forest model using the same variables as we did in the linear regression model. We will use the default settings and compare their performance to our benchmark model. Additionally, we are interested in using XGBoost to improve the accuracy of the model. The n_estimator is the number of trees in a random forest model. Increasing the number of trees typically improves the performance of the model, but it also makes the code slower to run. We will try tuning the hyperparameters like the n_estimator and try to achieve better model performance.

3. Results

	Before Data Cleaning		After Data Cleaning		RMSE Change% * **	
Models	**RMSE (training set)**	**RMSE (test set)**	**RMSE (training set)**	**RMSE (test set)**	**RMSE (training set)**	**RMSE (test set)**
Polynomial regression (2nd degree)	199171.19	208949.375	140084.71	142418.952	29.667%	31.84%
Random Forest with XGBoost	52797.60	133128.411	133004.893	137449.612	-151.915%	-3.246%
Random Forest	221772.838	306347.824	54248.568	144689.441	75.539%	52.78%
KNN(location)	211154.89	230262.632	93599.54	116734.039	55.673%	49.304%
KNN(all)	311097.015	338329.645	172962.19	193329.84	44.402%	42.858%

Table1：Model performance comparison (RMSE)

**Computed based on: (RMSE before data cleaning - RMSE after data cleaning)/RMSE before data cleaning.*
*** A positive percentage means decrease in RMSE and negative percentage means an increase in RMSE*

Table 1 summarizes the performance of each model with the best hyperparameter regards the RMSE. For the model obtained on the raw dataset without cleaning, the one with the lowest RMSE is random forest with XGBoost which has RMSE equal 133128.411 and the worst one is KNN using all possible features. After outliers are removed, KNN shows huge improvement for both with all features and location only and the one using only location in the distance metric function becomes the best model. However random forest with XGBoost has even an increase in RMSE.

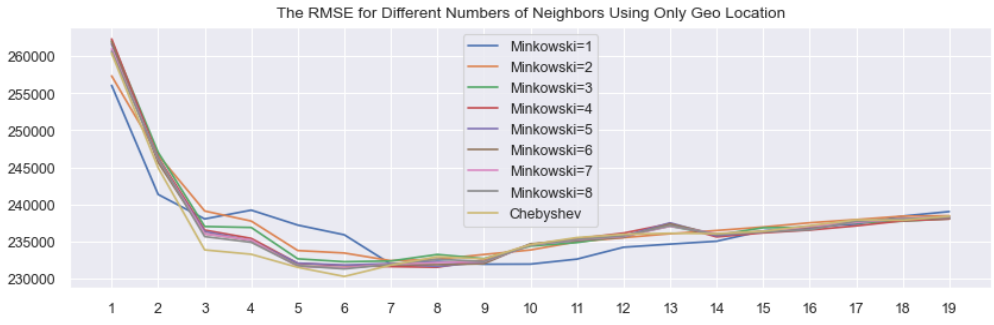

Figure 7: RMSE for KNN model using location with outliers

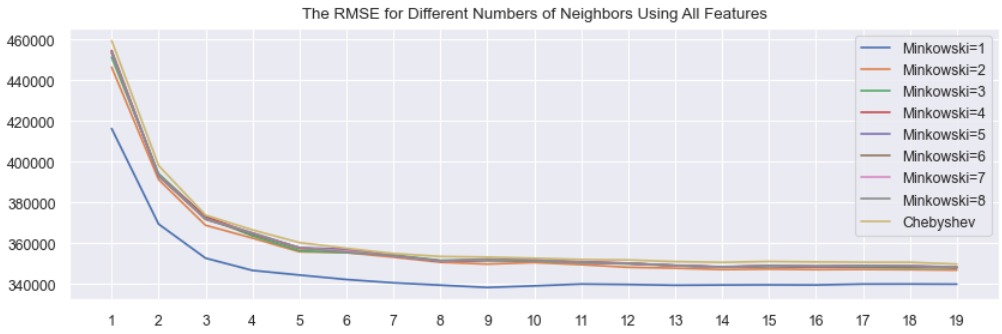

Figure 8: RMSE for KNN model using all features with outliers

Figure 7 and figure 8 show the performance of KNN model on a test set of the raw data using two sets of variables. Figure 7 shows the testing performance using just longitude and latitude while figure 8 shows the performance using all possible variables as mentioned in part 1.1. Based on the result, we found the optimal combination of K and distance metric is Chebyshev distance at K = 6 using just longitude and latitude with RMSE equals 230262.632, and Minkovski distance with order equals 1 (e.g. the Manhattan distance) at K = 9 using all possible variables with RMSE equals 338329.645. We observe that the performance using more variables in the distance function underperforms the one using only geographical location. The RMSE almost doubles at each K when switching to all features. Especially we observe that when using all possible variables, the order 1 Minkovski distances outperform all other distance metric at any K.

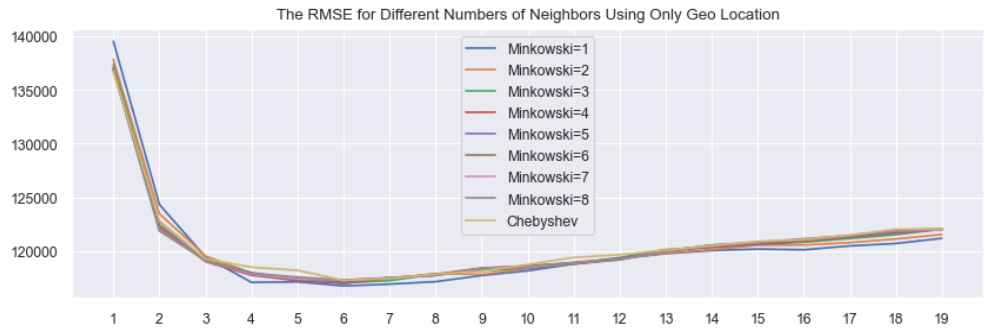

Figure 9: RMSE for KNN model using all features without outliers

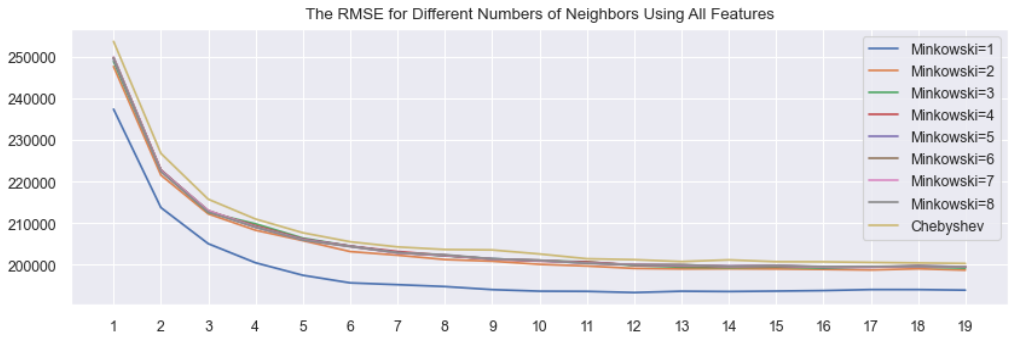

Figure 10: RMSE for KNN model using all features without outliers

Figure 9 and 10 shows the performance on dataset after removing the outliers. The optimal combination we observed for the one using only geographical location is Minkovski distance of order 1 at K = 10 with RMSE equals 116734.039, and Minkovski distance of order 1 at K = 4 with RMSE equals 193329.84 for the one using all possible features. It can be seen that both graphs have a similar pattern as its counterpart in figure 7 and 8. Also the performance of cosine distance is explored but of extremely bad performance. The performance graph is put in the appendix for information.

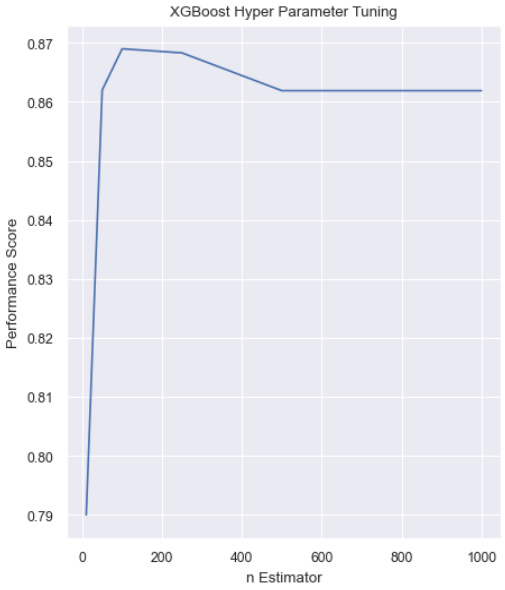

Figure 11: Random Forest Performance Score (Rr) by n_Estimator

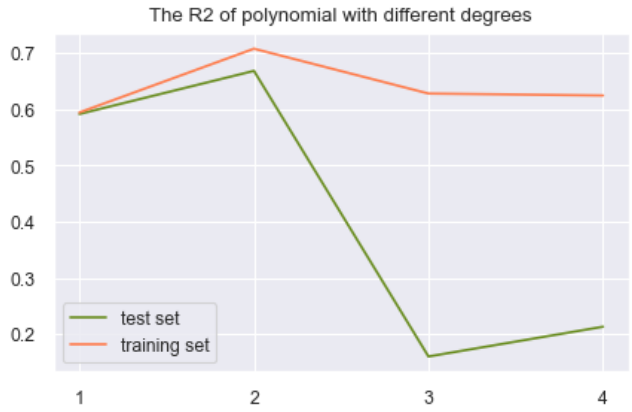

Figure 12: Polynomial Regression Model Performance

Figure 11 shows the result of the n_estimator tuning of the random forest model as shown above. In our case, we found that increasing the n_estimator beyond 500 did not further improve the performance of the model.

Figure 12 shows the polynomial regression model performance (at different degrees) on both test set and training set.

4. Discussion

Regarding the RMSE of KNN model, apparently the model that includes all variables we selected has relatively poor performance regarding its RMSE that is approximately 44% higher than the model only includes the location (longitude and latitude) information, which implies that fewer variables work better in our context. Logically, location is more natural to be considered as the relative similarity so this result make sense. This may also because of anomalies of some of our variables, for instance, as we mentioned the bedroom includes extreme outliers. Moreover, after we removed the outliers the difference of RMSE became smaller, although the difference in percentage increased to almost 80%. For figure 7 and 9 we see that as K surpass some threshold, the test RMSE start to increase. This is a very clear sign of overfitting. However, once we switch to using all features in the distance metric, this pattern fades away. The test RMSE keeps decreasing and converges to a certain level. As K goes beyond 10, there's no certain gain with incremental K while we could also anticipate a convergence of RMSE in the training set.

We have also tried to fit the polynomial regression and test with different degrees. The result is not satisfactory. We saw no improvement gain after the polynomial degree reach 2nd degree. However, the performance of polynomial regression with 2nd degree polynomials is not bad when compared with the KNN models, which is similar to the performance of the KNN model that includes the geographical variables only. While the polynomial model performance on the training set did not improve, its performance on the test set fell drastically which means the model suffers overfitting with higher polynomial degrees.

When comparing the performance of training and test sets, we initially found out that all the random forest model has a trend of overfitting, as the RMSE of the training set is over 50% lower than the test set, the XGBoost did boost the model accuracy but it did not help with the overfitting. However, in terms of the performance regarding the test set, it is definitely the best performanced model, the RMSE of which is around 35% lower than the best performanced polynomial model among the other three.

When it comes to the effect of outliers, as we expected, the performances of most models are improved. For the KNN models, the RMSE decreased by around 50% . The polynomial model has a relatively slight improvement with roughly 30% decrease of RMSE. Random Forest model seems to generate some overfitting after we cleaned the

outliers, and the RMSE decreased dramatically. However the Random Forest with XGBoost model has the opposite situation. Before the outliers are removed, the XGBoost has extremely overfitting. Its testing RMSE is almost double the training RSME. While after outliers are removed the overfitting seems to be eliminated. There is just 3% difference between the training and testing RMSE although the overall model performance almost has no change. This is an interesting phenomenon for further investigation.

5. Conclusion

Our study aims to find a simpler and more efficient model for housing price prediction and also shed light on quantifying the effect of outliers on model performance. In conclusion, our analysis shows that the KNN model with only the location variables performs better than the model with all variables. This may be due to anomalies in some of the variables, such as extreme outliers in the bedroom variable and negative prices in grade 13. Removing these outliers improves the performance of the KNN model. Polynomial regression with a 2nd degree polynomial also performs well in comparison to the test set and training set, but it also has a tendency to overfit the data. Outliers removal improves the performance of most of the models, with the largest improvement seen in the KNN models and changes our decision on the best model to pick. Overall, our analysis suggests that the location variables are important factors in predicting house prices. KNN is the best simple model trained using outlier removed dataset. At last, outliers are indeed crucial in model selection and it's a must to be done before a task starts.

References

[1] https://www.kaggle.com/datasets/harlfoxem/housesalesprediction/code
[2] https://www.kaggle.com/code/faressayah/predicting-houses-prices-using-anns
[3] https://www.kaggle.com/code/ayushikaushik/deep-neural-network

Appendix

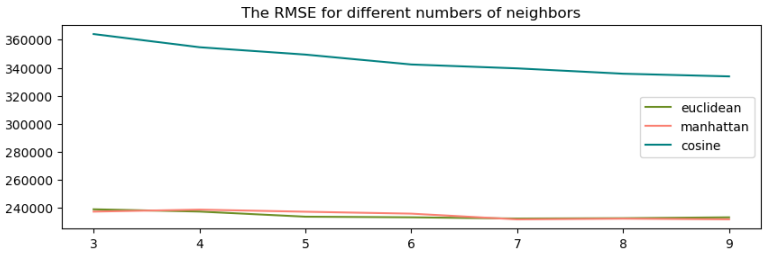

Figure 11: RMSE for KNN model with location

Code

See attached python code.