

**Reporting to:**

**DR. MOHD HALIM MOHD NOOR**

# PROJECT REPORT

A COMPARISON OF  
REGRESSION MODELS'  
PERFORMANCE IN  
PREDICTING HOUSE PRICES

**Reported by::**

**MUHAMMAD RAMDHAN HIDAYAT**

(P-COM0078/20)

)

**XU YANZHAO**

(P-COM0214/20)



**USM**

## Table of Contents

1.0	Project Background .....	2
1.1	Background of the problem domain .....	2
1.2	Issue and Problem Statement.....	2
2.0	Literature Review .....	3
2.1	Hedonic Pricing.....	3
2.2	Previous studies of ML Implementation on Hedonic Pricing Model.....	3
2.3	Proposed Methodology.....	4
3.0	Methodology.....	4
3.1	Project Framework .....	4
3.2	Dataset and Preparation.....	5
3.2	Dataset Description.....	5
3.2.1	Preparation – exploratory analysis .....	5
3.2.2	Preparation – cleaning and preprocessing .....	7
3.3	Machine Learning Algorithms .....	11
3.3.1	Parametric: Ridge Regression and Lasso .....	11
3.3.3	Non-Parametric: K-Nearest Neighbors (KNN) .....	12
3.3.4	Non-Parametric : Decision Tree Regression .....	13
3.4	Feature Selection .....	13
4.0	Experiments and Analysis .....	15
4.1	Experimental Setup .....	15
4.2	Result Analysis .....	16
4.2.1	OLS, Ridge, and Lasso Regression .....	16
4.2.2	KNN Regression .....	18
4.2.3	Decision Tree Regression .....	20
5.0	Summary and Future Work .....	21
6.0	References .....	23

## 1.0 Project Background

### 1.1 Background of the problem domain

Real estate is a key sector in the economy of each country, and the housing market is an essential part of it. The rental market of housing has been booming as urbanization occurs and has been very complex in metropolises. The price of accommodational need concerns both tenants and owners. Many tenants are young working staff and they have a strong need to find places with suitable conditions and fair prices. As for owners, setting a suitable price will help them maximize their profit.

A better understanding of mechanisms determining the rental price for all market participants is also good for forming a rental market with rational and benign development and thus reduces the risk of bubbles, which has happened a number of times in both mature and developing economies.

The problem domain of this project is the analysis of price formation of housing rentals using quantitative and automatic tools like machine learning algorithms, with an applicable model as the final product. CRISP-DM methodology will be applied.

### 1.2 Issue and Problem Statement

The prices of housing are formed by many factors, and this is reflected in a large number of attributes in the dataset discussed in 3.2. The attributes may not be independent but twined or convolved with others. Some attributes that are considered as simple may represent many associated sub-attributes. There may exist hierarchies among certain attributes. The data is obtained and generated into numerous entries, and errors and missing values are common and unavoidable. Although the dataset can be from a single source, it actually represents different scenarios and types of rental rather than one type. This results in the situation that the values of attributes have different varieties.

The major problems involved in this project can be summarized as:

- 1) How to explore and preprocess the data properly to make it suitable for training and testing?
- 2) How to choose suitable and useful features (attributes)? This requires us to find the most influencing attributes and to study the interrelationships among attributes.
- 3) Choice of suitable machine learning algorithms based on characters of attributes and data which have suitable time complexity and accuracy. This requires us to test different methods and choose the most suitable measurement of model quality.
- 4) Machine learning model explainability. We hope to get a good understanding of attributes, the data, and the model imposed on them can be all reached, which is useful for bridging the gap between IT niche knowledge and business practice without the direct involvement of such technical backgrounds.

### 1.3 Objective

The objective of the project is to build a machine learning model to predict the rental price based on the dataset discussed in section 3.2 with solving issues and problems raised in section 1.2.

## 2.0 Literature Review

Many known Machine Learning (ML) techniques have been applied to the house price forecasting problem. Some methods returned a better prediction performance than others, depending on the characteristics of the data being used. In this section, the theoretical foundation of house pricing (Hedonic House Pricing) will be briefly explained and then followed by a comparison of previous studies on the implementation of ML algorithms on it.

### 2.1 Hedonic Pricing

The Hedonic Pricing (HP) theory is one of the most popular and classical theory on the house price forecasting problem. Studies on the valuation of properties often refer to this theory as their theoretical basis (Chau & Chin, 2002). The HP model proposes that residential properties are differentiated commodities; hence their value is influenced by their heterogeneous characteristics such as structural and location profiles.

The formalization of HP theory on a market equilibrium condition is as follows:

$$V(H) = f(S, L) + \epsilon$$

*Equation 1: Hedonic Pricing*

Where  $V(H)$ ,  $S$ , and  $L$  are the price of a residential property, structural and locational features respectively. The  $\epsilon$  refers to the reducible error produced by the model being applied.

### 2.2 Previous studies of ML Implementation on Hedonic Pricing Model

The HP theory has been around for some time since its first application in 1996 by Lancaster (Montero & Fernández-Avilés, 2014). Several studies tried to employ machine learning techniques based on HP theory with varying results.

A study of several cities in the US (Dorr, 2016) challenged HDM's viability by employing Linear Regression (LR) model to test the predictive performance of a property's square footage on its price. However, while the finding was significant at the  $\alpha = 0.05$  level, the coefficient of determination was not as robust as anticipated.

On a study of real estate in China's metropolises, (Xiao, 2016) faced methodological obstacles related to the inherent spatial autocorrelation effect that usually comes with hedonic analysis. This effect violates the assumption made by regression methods that variables are independent. (Xiao, 2016) observed that the impact of nearby amenities on the price of residential properties are somewhat mixed. The study summarized that while accessibility to nearby amenities like supermarket appears to be negatively correlated with the price, other amenities such as a parking space showed positive correlations.

In the classifiers category, the Naïve Bayes model was shown to give a better performance on classification tasks than the Random Forest model when implemented to predict the house price while considering the interest of both sellers and buyers (Valle, 2017).

In another study, Ridge Regression (or L2 Regularization) was found to improve the valuation performance of real estate properties in Szczecin (Poland) by lowering the average absolute percentage errors (APE) (Gnat, 2020).

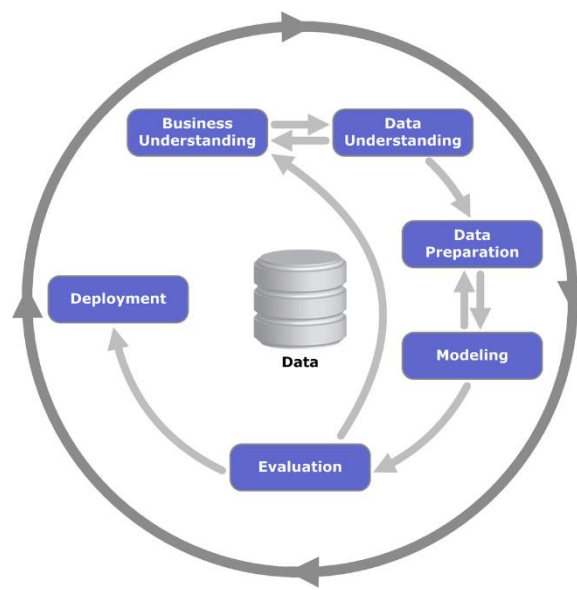
For KNN, (Baldominos et al, 2018) have shown that KNN offered better performance than LR, Random Forest (RF), and Support Vector Machines (SVM) in implementing the HP model. However, (Wezel and Potharst, 2005) found that Principal Component Regression (PCR) offered slightly better performance than KNN while (Ma et al, 2018) concluded that the KNN model could drive significant variation in the performance, based on the dataset provided.

### 2.3 Proposed Methodology

In accordance with the literature review, while also considering the characteristic of our dataset, we explored the use of parametric regression models such as Linear Regression, Ridge Regression, and Lasso Regression. The use of Ridge and Lasso model were considered due to its strength in dealing with multicollinear data (Maxwell et al., 2019), while KNN seems promising to give better performance than other options (Baldominos et al., 2018). The inclusion of Decision Tree model is for having another non-parametric model to compare with KNN.

## 3.0 Methodology

### 3.1 Project Framework



*Figure 1: CRISP-DM Methodology*

Figure 1 shows a data science workflow based on the CRISP-DM methodology.

The application of CRISP-DM methodology is a recursive process consisting of business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The business and data understanding phase are critical before working on other stages. This is because having a deep understanding on the characteristics of the data and how they are useful in a business context will lead to effective decision making in the later stage when sometimes some features have to be filtered out. Then, on the next stage, the data preparation and modeling task can be based on each other reciprocally, and the evaluation of the model has a direct improvement on business understanding as well, which may promote other steps in turn.

## 3.2 Dataset and Preparation

### 3.2 Dataset Description

The dataset is provided by AI Futurelab 2019, a notable Chinese AI contest. According to the official description, the dataset is provided by Cric China (<http://www.cricchina.com/>), a large real estate consulting company, and Data-as-a-service (Daas) and Platform-as-a-service(PaaS).

#### 3.2.1 Preparation – exploratory analysis

The dataset is in the form of a table. There are 50 input attributes plus the target in this dataset. There are

The attributes are the following grouped into 4 categories:

##### A. Section- Basic information

- 1 ID: ID of the house
- 2 area: the area of the house
- 3 rentType: wholly rent/share with others/unknown
- 4 houseType: e.g. 3 bedrooms+1 livingroom+1 bathroom
- 5 houseFloor: high/middle/low
- 6 totalFloor: total floors of the building
- 7 houseToward: orientation of the (main windows) of the house – important in China because of lack of sunshine during winter
- 8 houseDecoration: fine decoration/simple/undecorated/others

##### B. Section- Information of the community

- 9 CommunityName: name of the residential unit (condominium, apartment-like units)
- 10 city: SH means Shanghai
- 11 region: the district in the city
- 12 plate: the subdistrict
- 13 buildYear
- 14 saleSecHouseNum: the number of second-handed houses supplied in the plate/subdistrict

C. Section- Related facilities (of the plate; in China residential communities usually are lacking of exclusive facilities like them in Malaysia)

- 15 subwayStationNum: number of subway stations
- 16 busStationNum: number of bus stations
- 17 interSchoolNum: number of international school
- 18 schoolNum: number of public schools
- 19 privateSchoolNum: number of private schools
- 20 hospitalNum: number of (general) hospitals
- 21 DrugStoreNum: number of pharmacies
- 22 gymNum: number of gyms
- 23 bankNum: number of banks
- 24 shopNum: number of shops
- 25 parkNum: number of parks
- 26 mallNum: number of shopping malls
- 27 superMarketNum: number of supermarkets

#### D. Section- Other information (all counted per month for the plate)

28 totalTradeMoney: total trade amount of second-handed properties in the plate  
29 totalTradeArea  
30 tradeMeanPrice: mean price of sales of second-handed houses in the plate  
31 tradeSecNum: number of second-handed houses successfully dealt  
32 totalNewTradeMoney: total trade amount of new houses  
33 totalNewTradeArea  
34 totalNewMeanPrice  
35 tradeNewNum  
36 remainNewNum: number of new untraded houses  
37 supplyNewNum: number of newly supplied houses  
39 supplyLandNum: number of sections of land supplied  
39 supplyLandArea  
40 tradeLandNum  
41 tradeLandArea  
42 landTotalPrice  
43 landMeanPrice  
44 totalWorkers: number of working staff in the plate  
45 newWorkers  
46 residentPopulation  
47 pv: number of views on the webpages containing information about this plate  
48 uv: number of people who viewed the related webpages about this plate  
49 lookNum: number of offline viewing  
50 tradeTime

#### Target

51 tradeMoney: that is the rent

#### General analysis of attributes:

##### 1). Attributes are hierarchical:

Attributes grouped into Basic information and community name are independent as they are not affected or determined by other attributes. All other attributes are dependent on some other attributes, that is, attributes grouped in Related Facilities and Other Information, except tradeTime, plus saleSecHouseNum are determined by the plate of this house. The plate is further determined by the community name; the plate determines the region and city.

There are differences among entries with the plate, mostly in Other Information as they are counted by month and subject to fluctuations. This fact should be considered thoroughly for feature selections in order to select most useful features with no significant dependence among others.

##### 2). Attributes are from two perspectives:

Typically, participants involved in the real estate market are tenants, house owners, platform owners, or regulators. The basic information like area and decoration is available for all people, but attributes like supplyNewNum and totalWorkers may be only known by authorities. This provides more internal information for developing the model, but the availability of these attributes in future data can be a problem. So a prudent study of attributes and feature selection is crucial.

### 3). Imbalanced Data

There are two attributes with large numbers of missing values. They are rentType with about 74% (30758/41439) values missing as "Unknown" and houseDecoration with 70% (29039) of the entries missing as "other". There is only one value for 'city', so this attribute will be dropped.

After the removal of outliers and extreme values, a large number of numerical attributes show skewed or multimodal distribution.

The unbalance of the distribution of attributes is further seen when they are explored after cleaning by plates (see Python codes attached). In terms of built year, the majority of houses with known built year in plates like BK0007 and BK0030 are in 1980s and 1990s, while in plates like BK0045, BK0047 and BK0062 most houses are built after 2000.

#### 3.2.2 Preparation – cleaning and preprocessing

Firstly, instances with any of the essential attributes such like ID and plate missing, and attributes with only one value are dropped. Attributes with small number (less than 100) of missing values are processed by having respective rows dropped.

#### Missing Values:

The overall treatment of missing values is shown in Table 1.

*Table 1: Missing Values Treatment*

Attribute	Missing Number	Process Method	Reason
rentType	5	Fill the missing value with "unknown"	There are too many, more than half "Unknown" values in "rentType"
houseToward	962	Fill the missing value with the next non missing value in the dataset.	There are no special value distribution in "houseToward" group by community or plate, so we just simply fill the missing value with a python method fillna()
buildYear	2808	fill the missing value with the mode of 'buildYear' group by "plate"	the build year distribution is largely different among different plates
pv uv	18	fill the missing value in "pv" and "uv" simply by their mean values	the number of missing values is too few



## Outliers:

There are only one numerical continuous attributes (area) in basic information , and target attribute "tradeMoney" is also a numerical continuous attribute. We can only clean outliers in these two attributes, because other numerical continuous attributes are based on plate or month, if we delete some outliers from them ,much related data will also be deleted. The steps taken to remove outliers are as follows:

### 1.0 Area :

First we plot a boxplot of area as below:

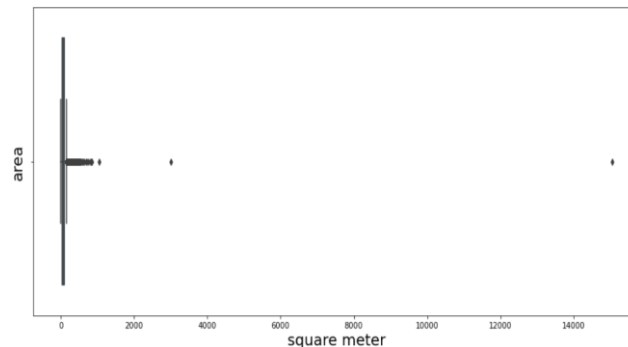


Figure 2: Boxplot of 'area'

There are some extreme values,so we decided to use IQR rule to delete the outliers, and limit the area to  $(-28.5, 161.1)$ , and since there are no negative values of area in real world, we limit the area to  $(10, 161.1)$ , the new area distribution is as below:

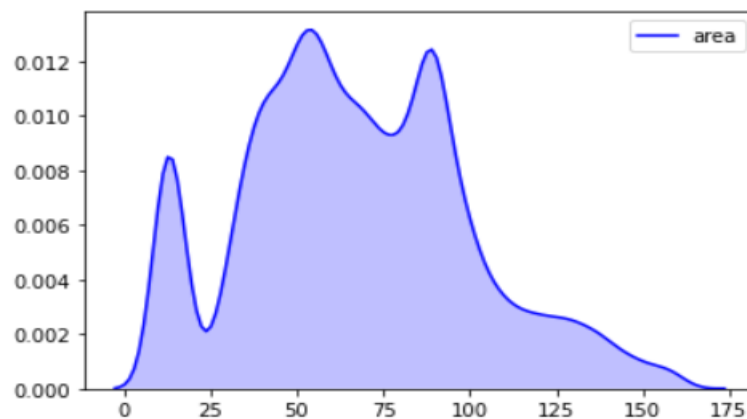


Figure 3: Distribution of Area

From the distribution of area we can see the area less than 20 has a peak, so we explore the data with area less than 20 further more .

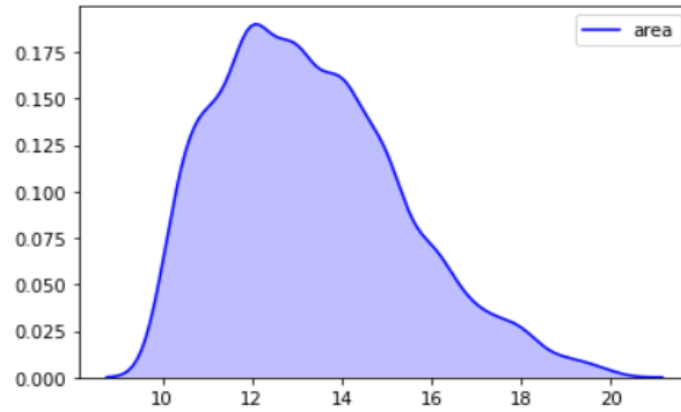


Figure 4: Distribution of 'area'<20

After exploration, we found that the areas less than 20 conform norm distribution and most of examples whose area is less than 20 were built during 1995-2010.

## 2.0 tradeMoney:

A boxplot of tradeMoney is as below:

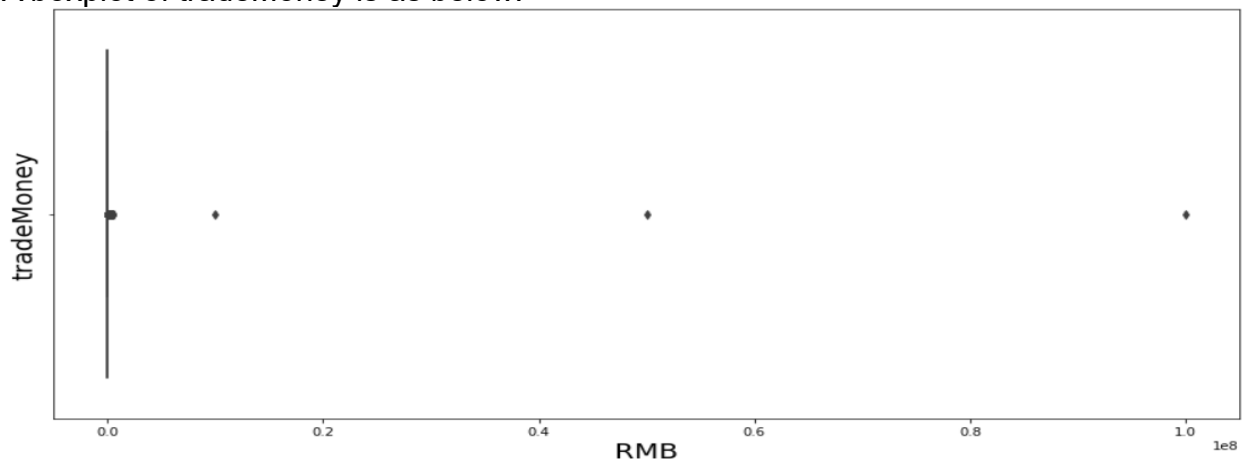


Figure 5: Distribution of 'tradeMoney'

We can see from the boxplot that there are some very extreme values in trademoney, so we limit the tradeMoney to (0,35000) to remove those extreme values.

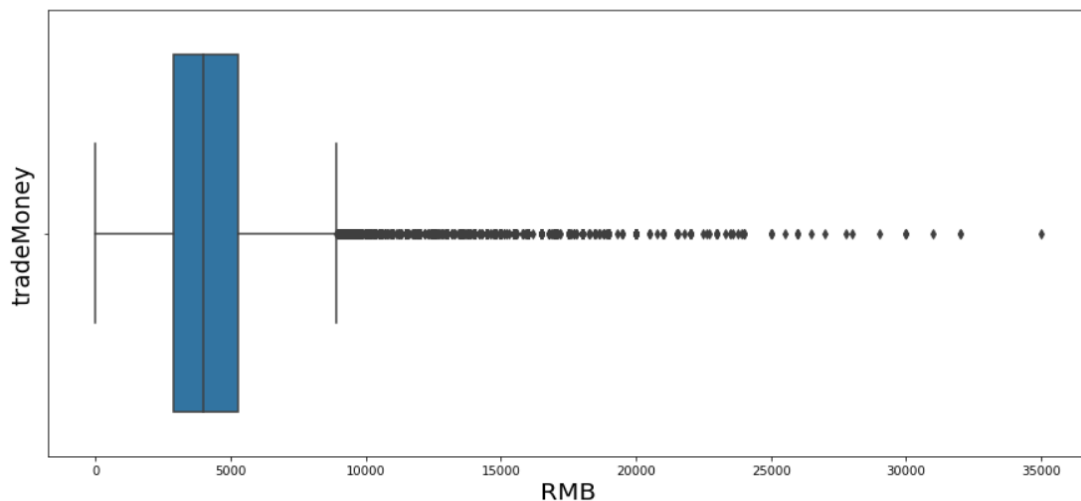


Figure 6: Distribution of 'tradeMoney' excluding outliers

There are still some outliers in tradeMoney ,but we don't limit the target value too narrow in order to avoid overfitting.

### data validation:

According to the official data description,there are some relationships between attributes,so we did a data validation process to make sure the data set is correct.

```
# check whether each value of "communityName" corresponds only one value of "plate"
df_check = df[['communityName', 'plate']]
df_check = df_check.drop_duplicates()
if df_check.shape[0] == len(list(df['communityName'].unique())):
    print('each value of "communityName" corresponds only one value of "plate"')
else:
    print('each value of "communityName" does not correspond only one value of "plate"')
```

each value of "communityName" corresponds only one value of "plate"

Figure 7: Feature Validation Code

We confirmed that the plate is further determined by the community name; the plate determines the region and city by Python code similar to the above picture. In addition , we found each value of "plate" corresponds only one series of value in attributes which describe neighbor facility number and added a "tradeMonth" attribute by extracting month in "tradeTime" attribute to simplify this attribute.

We found almost all attributes in other information are determined by month and plate except for 'pv','uv' and 'lookNum'. So we deleted those three attributes because they don't conform the official data description and we think these attributes may be erroneous.

### data transformation:

The attribute houseType is split into into three attributes: livingroom, bedroom, and bathroom.

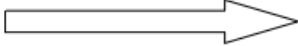
houseType					
0	2Br1Lv1B		bedroom	livingroom	bathroom
1	3Br2Lv2B		0	2	1
2	3Br2Lv2B		1	3	2
3	1Br1Lv1B		2	3	2
4	3Br2Lv3B		3	1	1
			4	3	2
					3

Figure 8: Data Transformation Process on 'houseType'

Ordinal categorical attributes such as houseDecoration and houseFloor are transformed into separate numerical values describing each of the item they are representing.

### 3.3 Machine Learning Algorithms

#### 3.3.1 Parametric: Ridge Regression and Lasso

Ridge Regression (RR) can be considered as an upgraded Linear Regression (LR) to deal with multicollinearity in a dataset (independent variables are highly correlated). Multicollinearity means that the variance of variables is high, which causes high deviations of observed values from their actual value. The RR model deals with this problem by adding a degree of bias to the regression estimates. The difference between LR and RR in estimating the cost function can be illustrated by the below formula:

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left( y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2$$

Equation 1: The Cost Function of Linear Regression

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left( y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p w_j^2$$

Equation 2: The Cost Function of Ridge Regression

Where:

- M: number of instances
- p: number of features
- x: input
- y: output
- w: slope

As shown by the two equations above, the main difference between the two models is that in RR, the cost function is altered by adding a penalty term  $\left( \lambda \sum_{j=0}^p w_j^2 \right)$ . This term is often called

the *Euclidean Norm* (or *L2 Norm*). The penalty term will penalize the cost function when its coefficients take large values. Thus, the whole process can be seen as a method for shrinking the coefficient ( $w$ ). Methods that limit a model's complexity are called Regularization, which leads to another name for RR: *L2 Regularization*. RR is also often called *Ridge Classifier* when it is being used to perform classification tasks. RR minimizes prediction error in training phase by minimizing the sum of the squared magnitude of the coefficients.

Lasso Regression (LSR) is another regularization technique like RR. The term LASSO is the short term for *Least Absolute Shrinkage and Selection Operator*.

$$\sum_{i=1}^M (y_i - \hat{y}_i)^2 = \sum_{i=1}^M \left( y_i - \sum_{j=0}^p w_j \times x_{ij} \right)^2 + \lambda \sum_{j=0}^p |w_j|$$

*Equation 3: The Cost Function of Lasso Regression*

When the lambda value equals zero, the Sum of Squared Residual becomes the Ordinary Least Squared (OLS) Method. This is basically the 'usual' Linear Regression model, which is also known as OLS model. As we increase the value of the lambda, the parameters decrease to a large degree and often also decrease to zero. In the training phase, Lasso model minimizes the prediction error by minimizing the sum of the absolute value of the coefficients, which is also known as the L1 norm of the vector of coefficients. When the coefficient of a variable is large in magnitude, LSR penalizes the coefficient by driving one or more of them to zero.

### 3.3.3 Non-Parametric: K-Nearest Neighbors (KNN)

As a classifier, the KNN model assumes that data points that are close together share similar labels. While this assumption remains true for low dimensional space, it tends to breakdown as the dimension of data increases (Aggarwal et al., 2002).

The quality of KNN performance relies heavily on its distance metric. While there are several options for the metric, the most commonly used is the *Minkowski Distance*, which is formalized by

$$\text{dist}(\mathbf{x}, \mathbf{z}) = \left( \sum_{r=1}^d |x_r - z_r|^p \right)^{1/p}$$

*Equation 4: The Minkowski Distance*

Where:

- p: distance parameter (any non-negative value)
- d: dimensionality
- x: feature instance
- z: class instance

The distance parameter  $p$  can take any non-negative value. When  $p = 1$  the metric is known as the *Manhattan Distance*, whereas when  $p = 2$  it is called the *Euclidean Distance*. For low-dimensional data, The *Euclidean Distance* works better. Then, as the data

dimension increases, the *Manhattan Distance* becomes the preferred choice (Aggarwal et al., 2002).

### 3.3.4 Non-Parametric : Decision Tree Regression

Decision tree is a supervised learning algorithm, which learning process is done by asking a series of questions to the series of information (data) it encountered. The whole learning process represents a tree as its name suggests. The tree is constructed through recursive partitioning, which starts from the root node as the first parent node and then continuously splits and produce child nodes until the pre-defined *maximum depth* threshold is achieved. Maximum depth simply means the longest path from the root of the tree to the leaf (the last child nodes).

Decision Trees are divided into Classification and Regression Trees. Regression trees are needed when the response variable is numeric or continuous. In classification tasks, the common impurity criterion used is the Entropy. However, for regression tasks the impurity measurement is done using the weighted mean squared error (MSE) of the children nodes since it can handle continuous variables.

$$\text{MSE}(t) = \frac{1}{N_t} \sum_{i \in D_t} (y^{(i)} - \hat{y}_t)^2$$

*Equation 5: MSE in Decision Tree Regression*

Where  $\hat{y}$  is the actual value of target variable,  $D_t$  represents the training subset at the node  $t$ , and  $N_t$  denotes the quantity of training samples at node  $t$ ,  $D_t$ .

### 3.4 Feature Selection

The basic concept behind feature selection is that given a set of features  $S=\{s_1, s_2, \dots, s_n\}$ , which of its subsets maximize a learner's ability to identify patterns. The mechanisms of how a feature selection method takes place can be divided into two categories: filter and wrapper—the filter category filter out variables by using statistical techniques, whereas the wrapper method applies models to subsets of features then select features that result in better performance based on a predefined scoring metrics. In this project, the filter method was chosen and performed using Principal Component Analysis (PCA) and R-Square Test.

The general concept when using PCA as a method for selecting features is to pick the variables according to the magnitude (from the largest to the smallest in absolute values) of their coefficients (loadings). The more lower-level concept is that PCA attempts to replace  $p$  (more or less correlated) variables with  $k < p$  uncorrelated linear combination of the initial variables (projected variables).

In this project, PCA was applied on features describing locational profiles of each house. Those features are tabulated in Table 2 below.

Table 2: Dimensional Reduction Using PCA on Location Features

Combinations of Location Features	Features
Combination 1	'subwayStationNum', 'busStationNum'
Combination 2	'interSchoolNum','schoolNum', 'privateSchoolNum'
Combination 3	'hospitalNum', 'drugStoreNum', 'gymNum', 'bankNum','shopNum','parkNum', 'mallNum', 'superMarketNum'

In addition to PCA, correlation analysis was also employed and it's result was used as the basis for feature selection. The correlation score is as visualized in Figure 9 below.

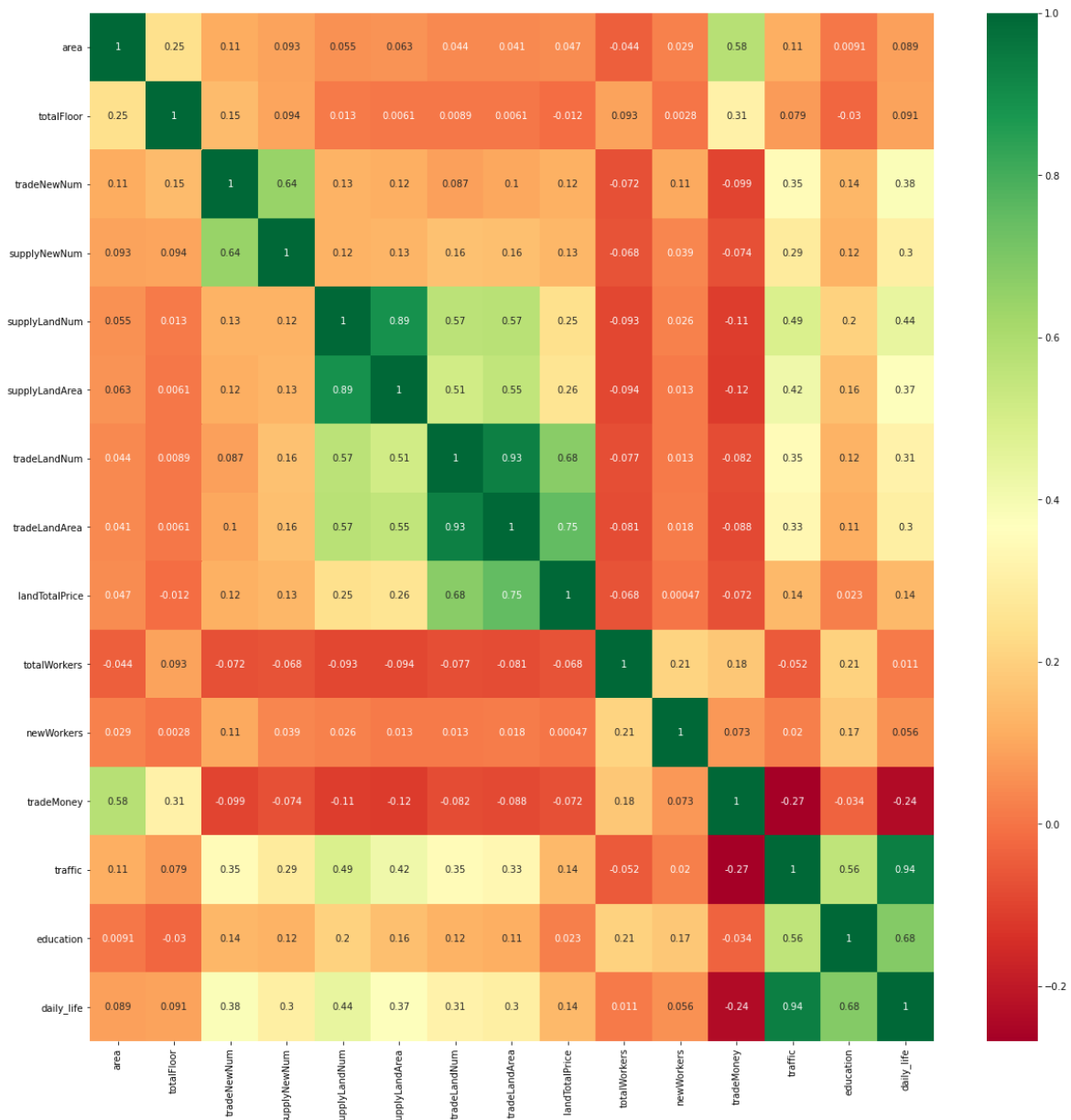


Figure 9: Correlation Scores using Heat Map

The correlation score visualized in colours ranging from the lowest (represented by red) to the highest (represented by green). There are several features having very high correlations (shown by dark green color): 'supplyLandArea', 'tradeLandArea', 'daily\_life', and 'traffic'. We chose to drop the first two and then merge 'daily\_life' and 'traffic' into one feature named 'convenience' using PCA. Although there are some features having what can be considered to have high correlation (above 0.7), we chose to keep them since we will also use Ridge and Lasso models which are designed to address high correlation issue. The remaining data after feature and dimensional reduction phase have 38707 observations (rows) and 20 features (columns).

## 4.0 Experiments and Analysis

### 4.1 Experimental Setup

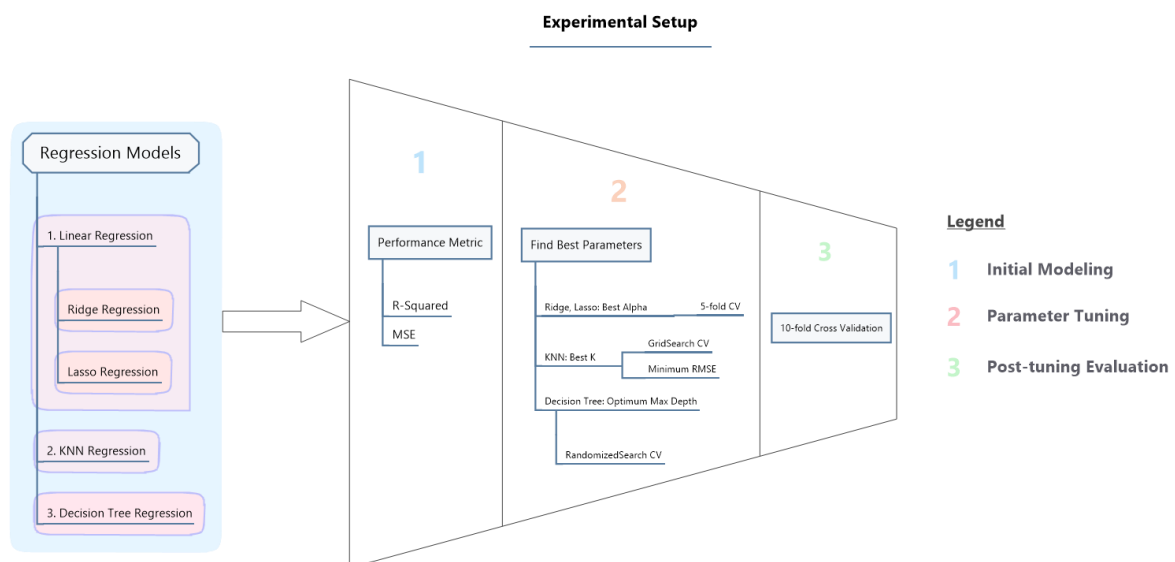


Figure 10: Experimental Setup

Figure 10 shows the diagram of experimental setup applied for this project. Regression models were processed through three layers of process. The first process was the initial modelling in which performance of models were evaluated without any parameter tuning. Then, we proceed to the second step in which several methods such as 5-fold Cross Validation (CV), Elbow Method using RMSE, Grid Search using GridSearchCV, and Randomized Search using RandomizedSearchCV were applied on regression models. Then, in the final stage, we evaluate the performance of each model using the respective best parameters and then did another Cross Validation using 10-fold setting in order to grasp the variability of the performance.



## 4.2 Result Analysis

### 4.2.1 OLS, Ridge, and Lasso Regression

In the first phase of experiment, we trained the non-parametric regression model (OLS) without any Regularization of parameters. The result is as shown in Figure 11. The result obtained from `LinearRegression()` give the value of `R_squared` to be 0.611 after rounding. This indicates that the model can explain around 61% of the data. Figure 11 also provides visualization of how the predicted 'tradeMoney' (Price) values are different in terms of their distribution.

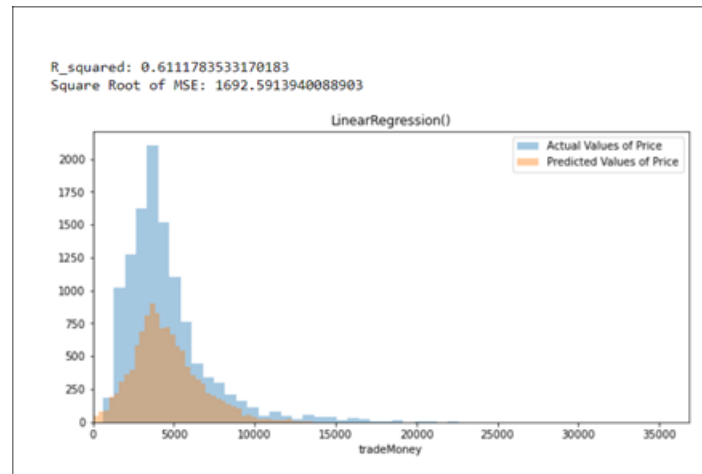


Figure 11: Linear (OLS) Regression Training Output

Next, we tried to find the best value of lambda that would give the best performance. Lambda was selected through generalized cross validation. Ridge Regression cross validation was done using 'RidgeCV' across the range of alpha values that we originally test using 5-fold cross validation. In addition, we also found that increasing the number of fold to 10-fold doesn't change the best alpha recommended by CV process. Then, we select optimal alpha value as shown. The optimal alpha value for our analysis ends up being 2.78 after rounding. We could get a more accurate value for our alpha by doing a more RidgeCV calculations at alphas near 2.78.

After that, the same process is applied on Lasso regression using LassoCV. For Lasso regression, the recommended best alpha is 0.359 after rounding and the `R_squared` is 0.611. This is not too different from what we get in OLS and Ridge model. In addition the distribution of predicted versus actual value is also similar to the other two parametric regression as shown in Figure 13.

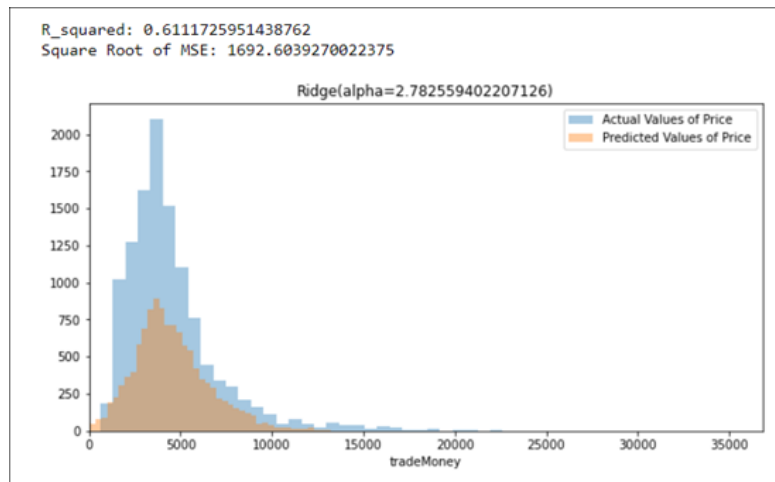


Figure 12: Ridge Regression with Best Alpha

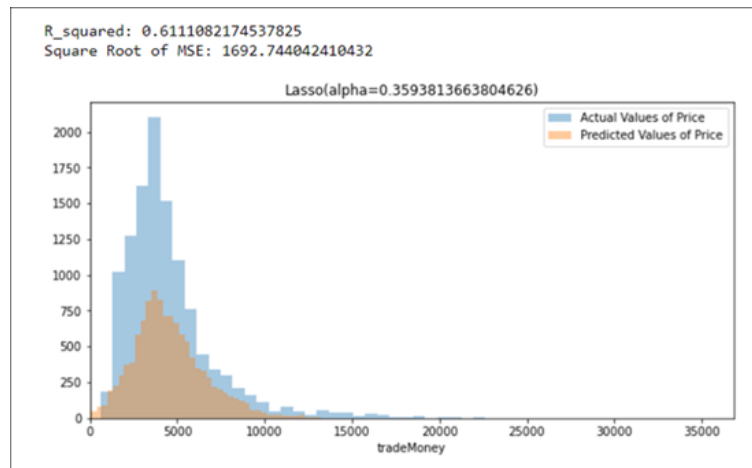


Figure 13: Lasso Regression with Best Alpha

From the results we have seen above, it turns out that OLS, Ridge, and Lasso doesn't have significant difference in the performance. This can be expected when the optimum parameter lambda for both Ridge and Lasso models is not too far from zero. As mentioned earlier, when parameters of Ridge and Lasso regression approaches zero, it would be a regular Linear Regression model or OLS.

The finding can also suggest that the treatment of dropping correlated value in the feature selection phase was the right one and have positive effect. This is because Ridge and Lasso model's strength is in dealing with data having high correlation between predictive features. The fact that these two models does not significantly improve the OLS model suggest this.

Next, we analyse the coefficients we have obtained from OLS, Ridge, and Lasso model.

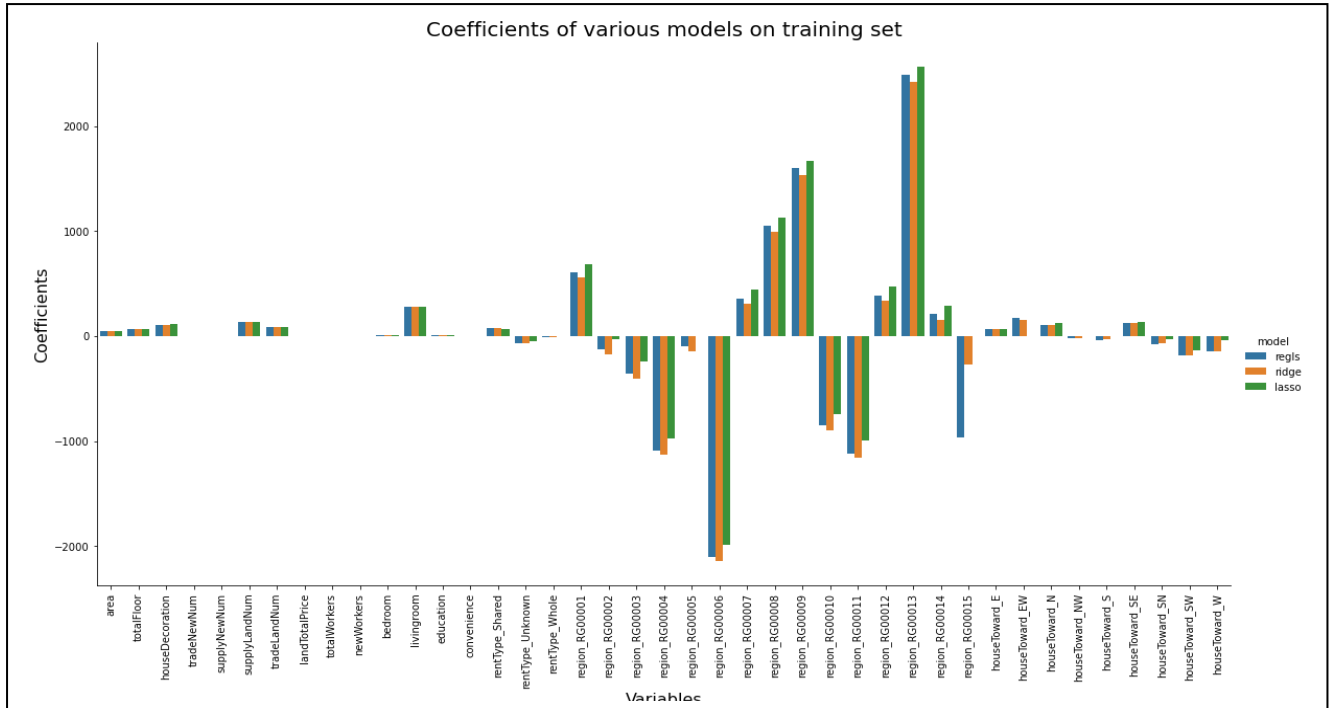


Figure 14: Comparison of Coefficients from OLS, Ridge, and Lasso Models

Figure 14 shows coefficients of property price's predictive features collected by OLS (blue), Ridge (orange), and Lasso (green) Regressions. The magnitude of coefficients is represented by the y-axis while its feature is noted in the x-axis. The highlight from the figure is that there are features having significantly larger coefficient's magnitude than others.

Overly large coefficient magnitudes can indicate that there are some issues in the model, especially related to collinearity. In this case, the large coefficients are belonging to regional location feature of the houses, which is represented by region\_00001 until region\_00014. This is agreed with the finding by (Xiao, 2016) covered in the literature review mentioning about auto-spatial correlation effect, which can induce collinearity among locational features. We can deduce from this finding that collinearity among locational features of real estate property such as houses is indeed an inherent characteristic of housing data.

#### 4.2.2 KNN Regression

The hyperparameter to be optimized for KNN is the number of the nearest neighbors  $k$ , which is expressed as `n_neighbors` in scikit.learn package. First, we employ the default setting of  $k$  (`n_neighbors=5`) then we evaluate the performance metric MSE across different values of  $k$ . We plot the results in Figure 15 and pick the best  $k$  using the elbow method. The figure suggests that the best  $k$  can be either  $k=5$  or  $k=6$ . Therefore, we explore this further using another method, which is GridSearch CV to get a more specific recommendation.

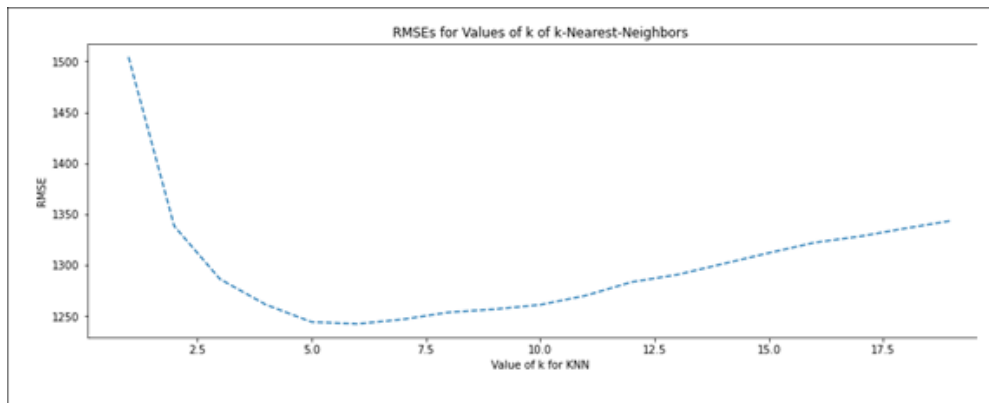


Figure 15: Determining best  $k$  using the elbow method

Setting the parameters to be checked ranging from  $k=2$  to  $k=9$ , the GridSearchCV recommend the best  $k$  to be 5 as shown in Figure 16.

```
#using GridSearch to get best k for comparison
from sklearn.model_selection import GridSearchCV
params = {'n_neighbors': [2,3,4,5,6,7,8,9]}

knr = neighbors.KNeighborsRegressor()

model = GridSearchCV(knr, params, cv=5)
model.fit(X_train,y_train)
model.best_params_

{'n_neighbors': 5}
```

Figure 16: GridSearch CV for determining best  $k$

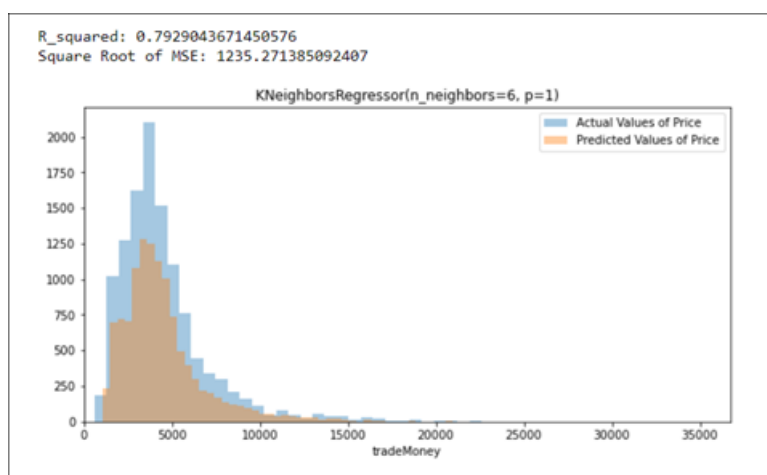


Figure 17: KNN Regression using Best  $k$  and Manhattan Distance

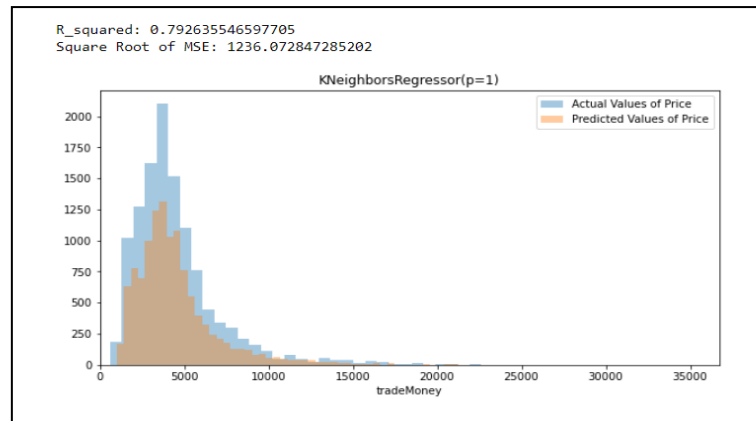


Figure 18: KNN Regression using  $k=5$

We further check by training the KNN regression model on both  $k=5$  and  $k=6$  with distance metric is set using the Manhattan Distance. The output of these two is shown in Figure 17 and Figure 18. Interestingly, we found that  $k=6$  give a slightly better performance as shown in the Figure 17. Although the difference is very small, it is an interesting finding that recommendation from GridSearchCV sometimes is not actually the best.

#### 4.2.3 Decision Tree Regression

In the decision tree model, there can be several hyper parameters to be considered. In our analysis, only the `max_depth` is the option for the hyper parameter. The range of the `max_depth` to be checked is the integers from 2 to 8. For this model, we use RandomizedSearchCV as we found using GridSearchCV on decision tree require more computing power and time. The result indicates that the optimal value for the `max_depth` is 8. The  $R^2$  is 0.757401 and the RMSE is 1336.968 under `max_depth=8`.

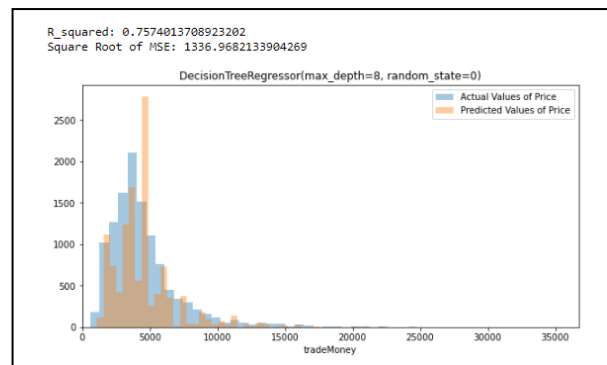


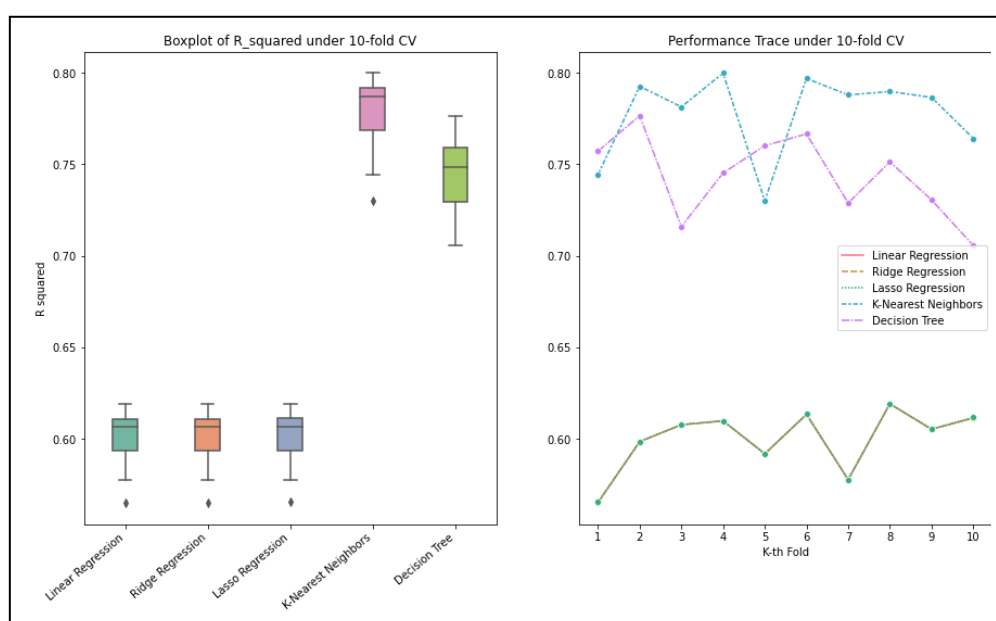
Figure 19: Decision Tree Regression using maximum depth = 8

Next, after best parameters were obtained for all regression models, we proceed by conducting a 10-fold cross validation to evaluate and compare their best performances. The result of this is tabulated in Table 1 below.

**Table 3: Post-Tuning Performance Evaluation Using 10-fold CV**

	Linear Regression	Ridge Regression	Lasso Regression	KNN Regression	DT Regression
1	0.5652452	0.5652899	0.5655088	0.7443622	0.7571130
2	0.5985176	0.5985068	0.5985230	0.7925644	0.7765678
3	0.6077311	0.6077738	0.6077131	0.7813201	0.7159529
4	0.6099265	0.6098821	0.6097655	0.7997649	0.7452404
5	0.5919693	0.5918811	0.5916859	0.7299761	0.7601150
6	0.6132821	0.6133926	0.6136244	0.7968352	0.7665724
7	0.5777565	0.5777815	0.5778752	0.7878885	0.7288255
8	0.6191526	0.6191457	0.6192140	0.7897356	0.7513057
9	0.6052822	0.6052969	0.6053582	0.7864888	0.7305396
10	0.6113751	0.6114532	0.6117293	0.7641475	0.7056543
Mean	0.6000238	0.6000404	0.6000998	0.7773083	0.7437886

According to the result in the table, the best regression model in our project is the KNN Regression since the mean of the scores for each round is the highest for KNN. Decision Tree came close with only 0.3% difference below KNN while the performance of Linear, Ridge, and Lasso are not differed too much from each other.



**Figure 20: Post-Tuning Performance Comparison using 10-fold CV**

The box and line plots above show the distributions and the variation of scores for each model under ten iteration of cross validation. The decision tree model as well as KNN shows the best performance in our analysis. However, the decision tree model produced a wider range of variability as shown in its boxplot and line plot. Linear and ridge regressions do not show a significant difference in their performance.

## 5.0 Summary and Future Work

## 5.1 Summary

- For this project, KNN regression is the best model in predicting house prices ('tradeMoney') using its structural and locational features. The Decision Tree model gave similar average performance, but it has a high variance issue as shown by its boxplot and.
- Linear, Ridge, and Lasso models suggests that the locational feature such as related to the region have multicollinear issue as shown that the large magnitude of their coefficients.

## 5.2 Future Work

- Ridge and Lasso Regression is valuable for multicollinear data but it may be wise to drop a predictor that causes the multicollinearity so that we do not impart bias into our sample.
- In this project, the collinearity detection was done using The Pearson's R correlation scoring method. Another detection tool such as Variance Inflation Factor can be used to give a more robust collinearity detection.
- Decision Tree Regression works well on range of values that it has seen in the training data, but could struggle when the range of values of the new data are outside the seen range. Training this model on a wider range of value for prices would improve its ability in predicting prices on new data.

## 6.0 References

1. Chau, K. W., & Chin, T. L. (2002). A Critical Review of Literature on the Hedonic Price Model (SSRN Scholarly Paper ID 2073594). Social Science Research Network. <https://papers.ssrn.com/abstract=2073594>
2. Montero, J.-M., & Fernández-Avilés, G. (2014). Hedonic Price Model. In A. C. Michalos (Ed.), *Encyclopedia of Quality of Life and Well-Being Research* (pp. 2834–2837). Springer Netherlands. [https://doi.org/10.1007/978-94-007-0753-5\\_1279](https://doi.org/10.1007/978-94-007-0753-5_1279)
3. Dorr, T. (2016). Real Estate Home Pricing: Hedonic model variables and Community Amenities Roles. *Journal of Management and Innovation*, 2(2), Article 2. <https://doi.org/10.18059/jmi.v2i2.27>
4. Xiao, Y., Chen, X., Li, Q., Yu, X., Chen, J., & Guo, J. (2017). Exploring Determinants of Housing Prices in Beijing: An Enhanced Hedonic Regression with Open Access POI Data. *ISPRS International Journal of Geo-Information*, 6. <https://doi.org/10.3390/ijgi6110358>
5. Baldominos, A., Blanco, I., Moreno, A. J., Iturrarte, R., Bernárdez, Ó., & Afonso, C. (2018). Identifying Real Estate Opportunities using Machine Learning. *Applied Sciences*, 8(11), 2321. <https://doi.org/10.3390/app8112321>
6. Gnat, S. (2020). Impact of the Regularization of regression models on the results of the mass valuation of real estate. *Folia Oeconomica Stetinensia*, 20(1), 163-176. <http://dx.doi.org/10.2478/fofi-2020-0009>
7. Masías, V. H., Valle, M., Crespo, F., Crespo, R., Vargas Schüller, A., & Laengle, S. (2016, January 20). Property Valuation using Machine Learning Algorithms: A Study in a Metropolitan-Area of Chile.
8. I van Wieringen, W. N. (2020). Lecture notes on ridge regression. ArXiv:1509.09169 [Stat]. <http://arxiv.org/abs/1509.09169>
9. Maxwell, O., Amaeze Osuji, G., Precious Onyedikachi, I., Obi-Okpala, C. I., Udoka Chinedu, I., & Ikenna Frank, O. (2019). Handling Critical Multicollinearity Using Parametric Approach. *Academic Journal of Applied Mathematical Sciences*, 511, 150–163. <https://doi.org/10.32861/ajams.511.150.163>
10. Zhang, S., Deng, Z., Cheng, D., Zong, M., & Zhu, X. (2016). Efficient kNN Classification Algorithm for Big Data. *Neurocomputing*, 195. <https://doi.org/10.1016/j.neucom.2015.08.112>
11. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory*. <https://doi.org/10.1109/TIT.1967.1053964>
12. Aggarwal, C., Hinneburg, A., & Keim, D. (2002). On the Surprising Behavior of Distance Metric in High-Dimensional Space. First Publ. in: *Database Theory, ICDT 200*, 8th International Conference, London, UK, January 4 - 6, 2001 / Jan Van Den Bussche ... (Eds.). Berlin: Springer, 2001, Pp. 420-434 (Lecture Notes in Computer Science ; 1973).