CSI-5387 Data Mining and Concept Learning
Project Report: Real Estate Analysis

Group 6:
Peng Ao 300145650 apeng055@uottawa.ca
Zhang Lingfeng 300134245 lzhan278@uottawa.ca
Wang Ning 300142185 nwang091@uottawa.ca
Huang Zhihe 300084133 zhuan039@uottawa.ca

GitHub Link: https://github.com/AaPaul/CSI-5387_Data_Mining

Abstract: In this project, the main task is to do a regression task for predicting the house price. We tried 17 kinds of regression models based on Scikit-learn and Keras frameworks with Python and got the relatively best performance in the multi-layer perceptron model in the majority of 11 kinds of evaluation metrics. Before regression, we also did dataset analysis and pre-processing, including data visualization, outlier detection, data normalization, etc. Apart from that, bins-based classification is used to transfer regression problems to multi-class classification problems. In addition, we generated 4 more datasets based on the original dataset by feature selection and dataset resampling, applied 4 kinds of models and used Friedman and Nemenyi test to find the best classification model with a significant difference.

Keywords: regression, classification, data visualization, statistical experiments, neural network

1 Introduction

There are kinds of factors having effects on real estate so it is difficult for people to predict the fluctuation of the price of the house. In this project, we tried to use machine learning methods to build a regression model to predict the unit area price of a house. Besides, we also have done the classification task by binning prices into several classes according to price range. In addition, we also analyzed the dataset and outliers and then pre-processed them. Visualization is another important tool in data science, so we have done several kinds of visualization methods to find some interesting patterns.

In the regression part, we compared 17 regression models and explored the 11 evaluation metrics for finding the best model, which is the multi-layer perceptron. In the classification part, we tried different methods to augment the dataset as the lack of enough samples has some negative effects on the whole processing. Then we built 4 models and tried 4 basic different metrics to evaluate our models and compare the performance. Finally, we got the best performance with the Decision Tree model.

The main tasks of this project can be summarized below:

- Dataset analysis and outlier detection
- Data preprocessing for regression
- Regression model construction
- Regression model selection and comparison
- Bin classes, feature selection and re-sampling
- Classification model construction
- Model evaluation in classification

2 Data Pre-processing

2.1 Dataset Introduction

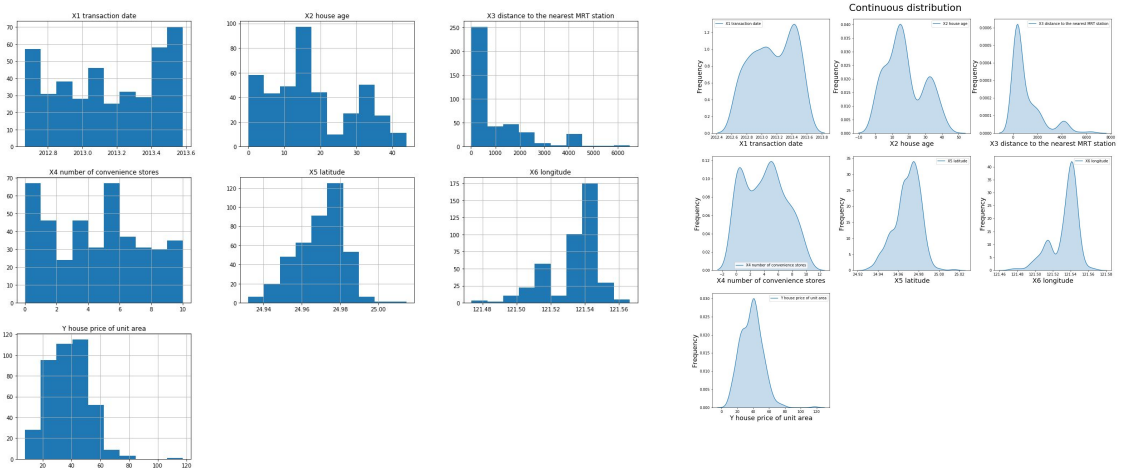
The dataset named *Real estate valuation data set* includes 7 attributes and 1 output, which is from the UCI machine learning repository. It contains 414 instances and there is no missing value in this dataset.

2.2 Data Exploration and Preprocessing

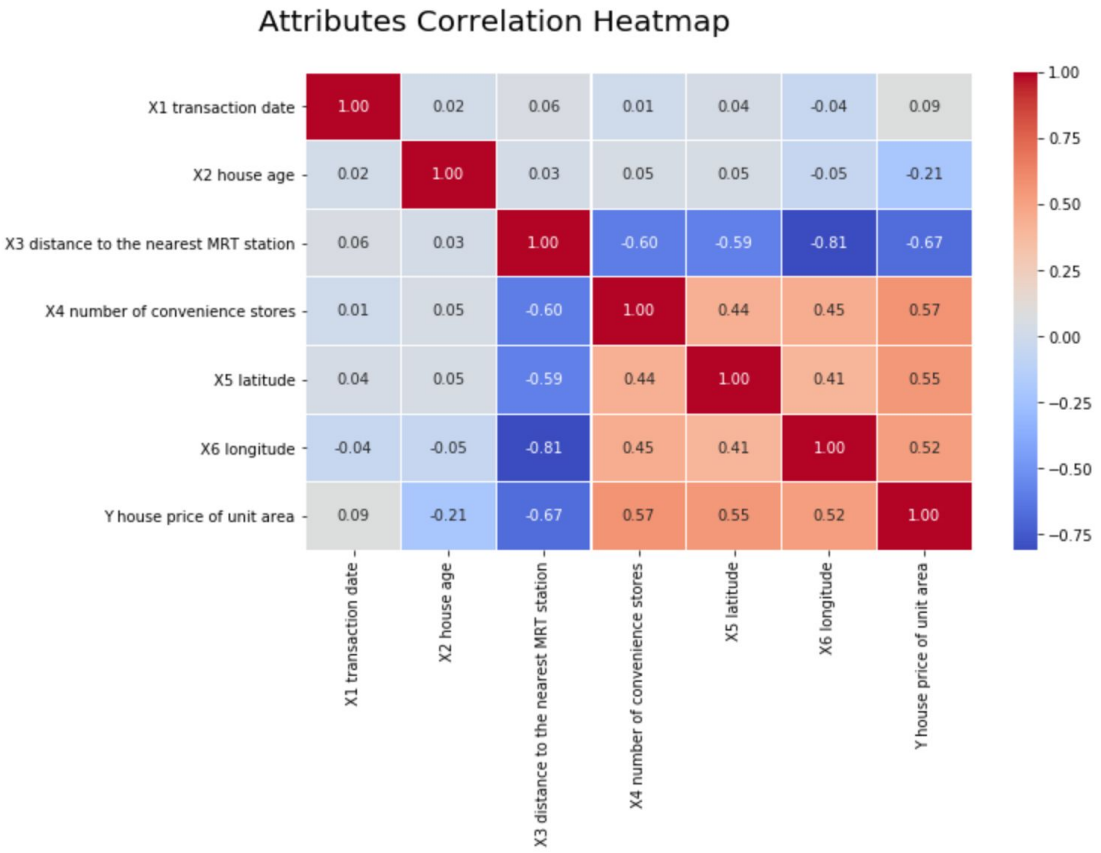
This part is composed of data preprocessing and the description of the dataset based on the former part. We mainly used kinds of plots to demonstrate this dataset and the potential correlation among the attributes.

Data Preprocessing is indispensable for data analysis. It helps to create clean and reliable data points for follow-up processing, including feature selection. Also, standardizing the dataset is contained in this part in order to simplify the processing during model construction and prevent the problem named the dominance of some features with large values.

The first step is to check if there are missing values and in fact, our data set is complete. After that, we empirically deleted one attribute named Number because this feature is useless for our task. Then we used the histogram and density plot to show a brief understanding of this dataset and its distribution for every attribute.



In order to get further information on the potential relationships among 6 attributes and the output, we chose the 2-dimensional heatmap.



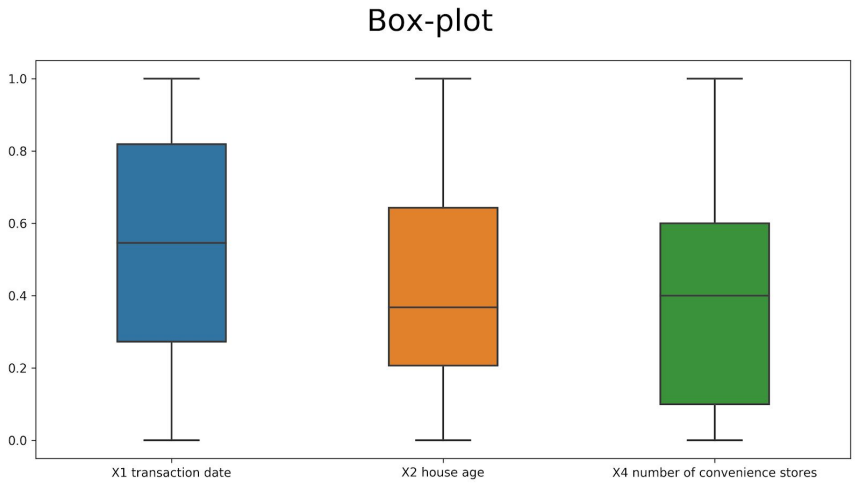
The next part is standardization and we used the *Min_Max Method* for re-scaling the dataset. Fixing the data distribution in the range from 0 to 1 is conducive to speeding up processing and improving the performance of the model, and reducing the weight of those attributes with larger values is beneficial for selecting features.

3 Outlier Analysis

Outlier detection is to deal with these kinds of data points that are distant from other observations. In our project, we tried 3 different methods to detect the outliers and got insights on how the outliers will affect our model based on the outlier analysis.

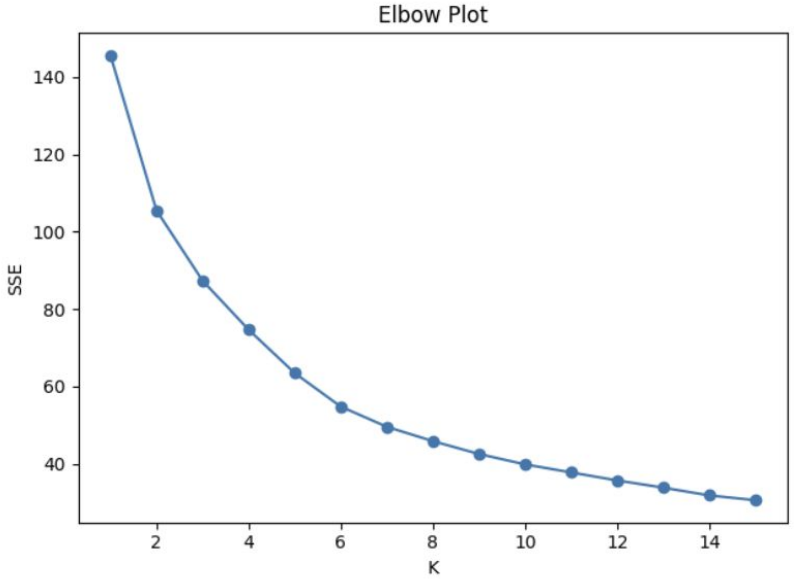
3.1 Box plot

The first method is the box plot from which people can get an intuitive feeling about whether there is an outlier or not. The limitation of this tool is that it can only work with the nominal data. Therefore, we choose 3 nominal attributes after standardization and draw the box plot showing below. From the figure, we can find there is no obvious outlier in our dataset. One reason may be because we just choose a part of attributes in the dataset and the other reason maybe because of the weakness of the dataset itself, lacking enough samples.

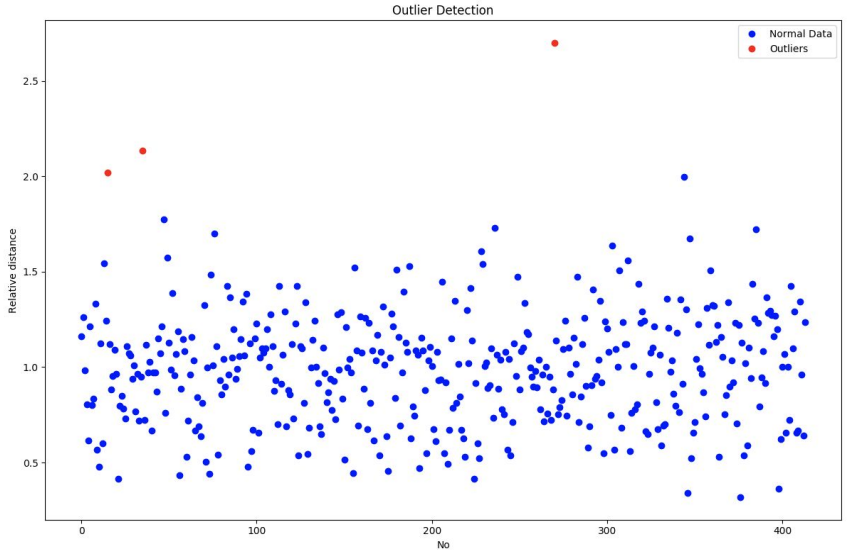


3.2 K-Means

Another method is K-Means, which is a common cluster algorithm. Cluster methods are often used for detecting outliers. We first used the elbow method to find a proper K value, which is 8. The reason why we do not choose too large K is that too many clusters in this small dataset maybe not beneficial for outlier detection.

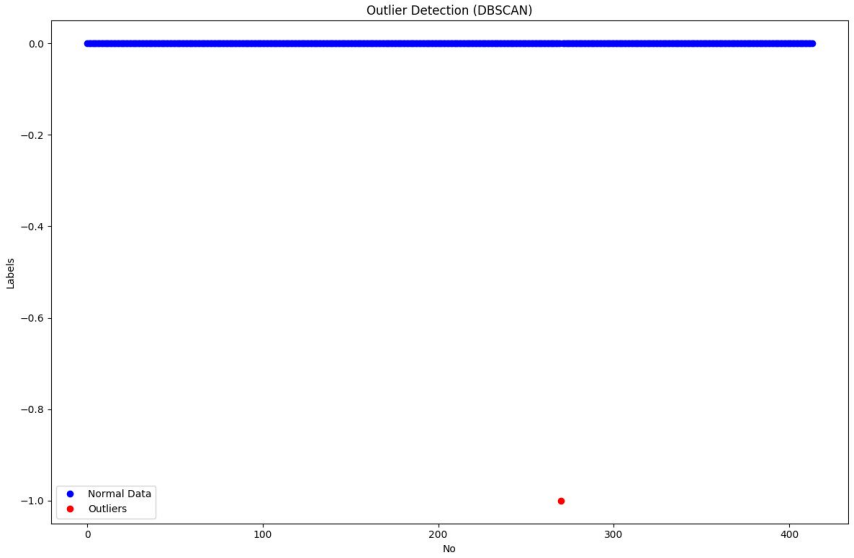


Then we created a K-Means model by using the Scikit-learn function named *K-Means* with K and 100 iterations to fit data. We simply divided the whole dataset into 2 sets. The one is normal data drawn by blue colour and another is the outlier with red colour. The method of recognizing the outlier is based on the absolute distance between every point and the center of each cluster and the threshold is set as 2. The figure named Outlier Detection shows the result that there are 3 possible outliers in our dataset.



3.3 DBSCAN

DBSCAN is another clustering method we use for outlier detection. We want to do a comparison between it and K-Means. The implementation is based on Scikit-learn as well. The model was built with epsilon 0.5 and 5 minimum samples, and these hyper-parameter settings can get the best performance, as the first parameter named epsilon controls the local neighbourhoods of the points and another parameter minimum samples determine how tolerant the model is. If these hyper-parameters are small, most data will be seen as the outlier and be labelled as -1. The figure below shows the result that there is only one outlier in the dataset and this outlier is the same as the one detected by K-Means.



As the first method cannot respond well to the entire dataset, we chose the second and the third methods as the approach to detect outliers. The common outlier detected by these 2 ways is No. 270 whose output is the highest (117.5) and the mean value of outputs is 37.98. Typically, it should be removed and we have also constructed this function. However,

because of the lack of samples in this dataset, we decided to keep it and we also wanted to check the robustness of our regression models by these outliers.

4 Regression

4.1 Regression models construction and analysis

4.1.1 Linear Regression Model

In this project, we use a linear regression model with a degree range from 1 to 5 separately because the degree is a hyper-parameter we should tune and we do not know which kind of polynomial regression model can fit the data properly before training models. Whether adding bias is another

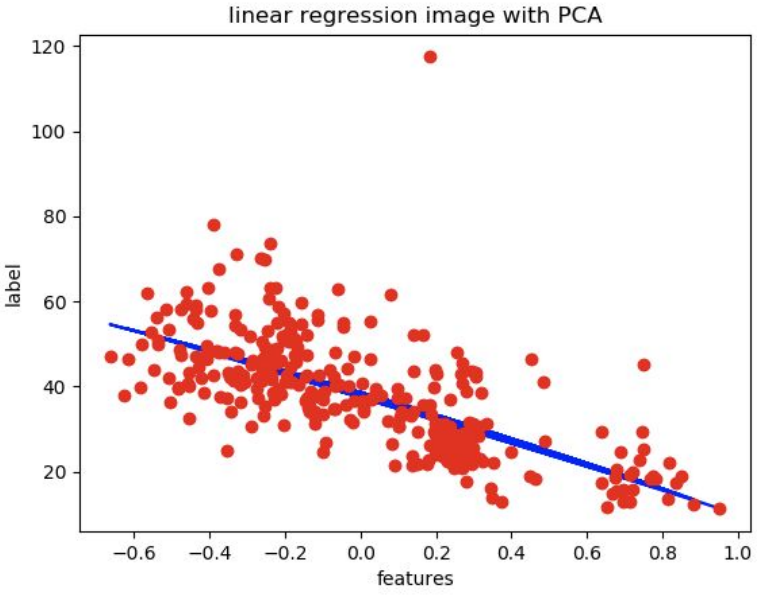
In addition, we also use linear regression with feature interaction because it can fit the data better. Intersection means one feature can do multiplication calculation with other features in the range of pre-setting degree, instead of only add a degree on a certain feature. For example, $y = ax_1^2 + bx_2^2 + cx_1x_2 + d$ is better than $y = ax_1^2 + bx_2^2 + c$ in practice.

The linear regression model optimization function is shown below:

$$min_w \, ||Xw - y||_2^2$$

The graph below shows the learning curve of linear regression with degree 2, and it seems to converge well.

The graph below shows the original data(red points) and the regression curve(linear regression with degree 2). Since the graph has to be drawn in two dimensions, the original data should be processed by dimensionality reduction, one dimension for X and another for Y. So we did a principal component analysis(PCA) to do feature selection, the most significant feature is selected by this method.



4.1.2 Ridge Regression Model

Linear regression is highly sensitive to random errors, producing a large variance if features are dependent. This is the problem of multicollinearity. Ridge regression addresses the problem of multicollinearity by imposing a penalty on the size of coefficients.

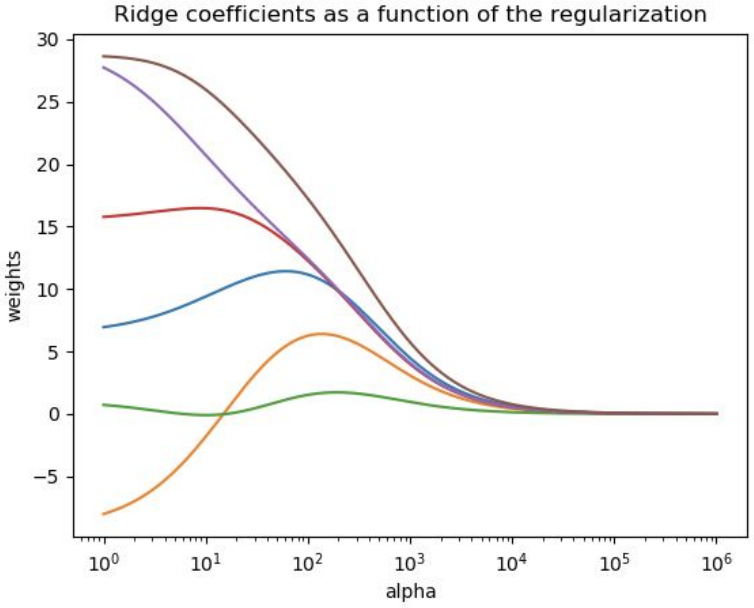
The ridge regression model optimization function is shown below:

$$min_w \ ||Xw - y||_2^{\ 2} + å||w||_2^{\ 2}$$

, where the greater å, coefficients become more robust to collinearity. This function uses L2-norm.

Scikit-learn package also supplies the 10-fold cross-validation ridge regression model, we can choose a list of alpha as hyper-parameters and a 10-fold cross-validation mechanism can help us choose which hyper-parameter is the best one and return the best model.

The graph below shows the ridge path, and this path represents the relationship between hyper-parameter alpha(penalty term) and weights. If the alpha value is too large, all weights tend to be the same and close to zero, meaning penalty too much. On the other hand, if the alpha value is too small, all weights are sparse and no penalty, which is the same as linear regression.



4.1.3 Lasso Regression Model

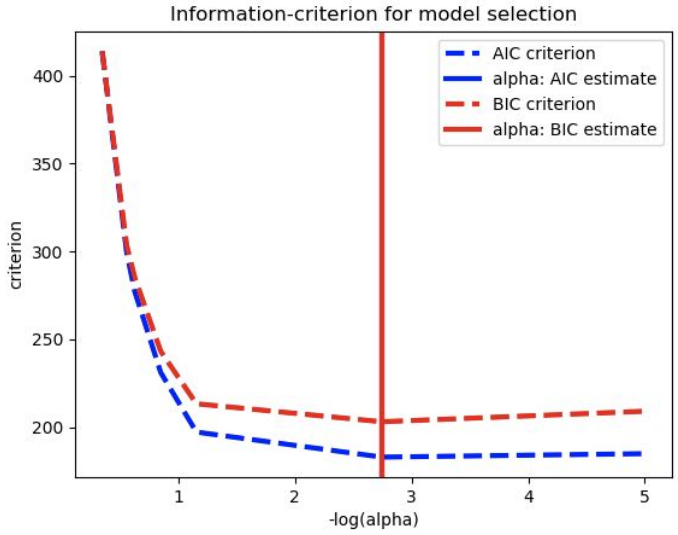
Lasso uses coordinate descent as the algorithm to fit the coefficients and the optimization function of lasso regression is shown below:

$$min_w \ (1/2n_{samples})||Xw - y||_2^{\ 2} + å||w||_1$$

This function uses L1-norm. The alpha parameter controls the degree of sparsity of the estimated coefficients. As the lasso regression yields sparse models, it can thus be used to perform feature selection. Scikit-learn also exposes objects that set the lasso alpha parameter by cross-validation.

Lasso can also use different criteria to do model selection, such as AIC and BIC. AIC means the Akaike information criterion and BIC means the Bayes information criterion. Such a criterion is useful to select the value of the regularization parameter by making a trade-off between the goodness of fit and the complexity of the model.

The graph below shows how AIC and BIC do model selection. Both two criteria get the same result. The best alpha can be used in lasso regression to get the best prediction.



4.1.4 Elastic Net Model

Elastic Net trade-off Lasso(sparse model, L1 norm) and Ridge(regularization, L2 norm). ElasticNetCV can be used to set the parameters alpha(å) and L1-ratio(rho) by cross-validation. The optimization function is shown below:

$$min_w\ (1/2n_{samples})||Xw - y||_2{}^2 + åp||w||_1 + (å(1-p)/2)||w||_2{}^2$$

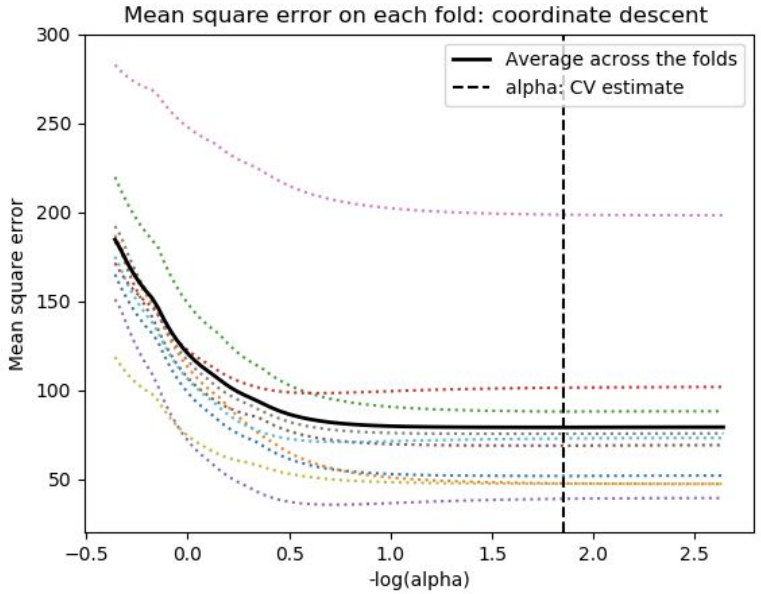
Initially, some elastic nets do not meet the convergence because default tolerance for the optimization is too small although the iteration number is large enough. So we change the maximum iteration and tolerance value to make the model converge.

4.1.5 Least Angle Regression Model

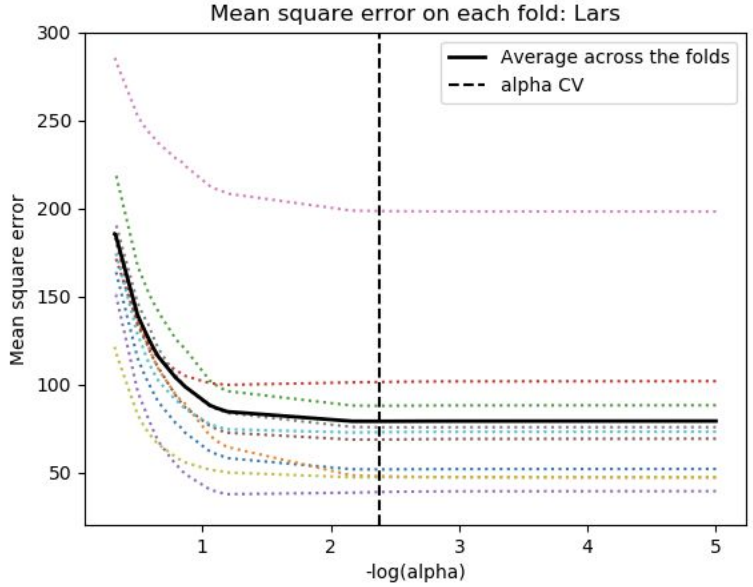
Least angle regression(LARS) is a regression algorithm for high-dimensional data. LARS is similar to forward stepwise regression. At each step, it finds the feature most correlated with the target. It is numerically efficient in contexts where the number of features is significantly

greater than the number of samples. Since LARS is based upon an iterative refitting of the residuals, it would appear to be especially sensitive to the effects of noise. LassoLars yields the exact solution, which is a piecewise linear function of the norm of its coefficients. Alpha is a hyper-parameter in this model, which means the penalty term.

The graph below shows the lasso path, meaning the best hyper-parameter in the lasso regression model can be selected by 10-fold cross-validation.



The graph below is the Lars path, same as the lasso path.



4.1.6 Bayesian Ridge Regression Model

Bayesian ridge regression is more robust to ill-posed problems(noise).

There is an assumption in the Bayesian ridge regression model, which is that weights are small. In addition, the prior coefficient w is given by spherical Gaussian:

$$p(w;\ \lambda) \sim N\ (\ 0,\ \lambda^{-1} I\ )$$

The task is given $\alpha,\ \lambda$ (these two priors are chosen from gamma distribution), find:

$$min_w\ \alpha||Xw - y||^2\ + \lambda||w||^2$$

4.1.7 Automatic Relevance Determination Regression Model

The automatic relevance determination regression model is similar to the bayesian ridge regression model, leading to sparser coefficients w. This regression model drops the assumption of the Gaussian being spherical and considers that each weight has an individual variance, so that:

$$p(w \mid \lambda) \sim N\ (\ 0,\ A^{-1}\ )$$

where $A = diag(\lambda_1, \lambda_2, ..., \lambda_H), \lambda_h \in R_+$

The optimization function is:

$$min_w\ \alpha||Xw - y||^2\ \ w^t A w$$

If the precision λ_h of the feature h is high, its weight w_h is very likely to be close to zero and is therefore pruned. The automatic relevance determination(ARD) regression model is also called the sparse Bayesian learning regression model.

4.1.8 Stochastic Gradient Descent Regression Model

Default loss of stochastic gradient descent regression model has squared loss, which is the same as ordinary least squares. If the loss is Huber, it is the same as Huber loss for robust regression. In addition, if the loss is epsilon insensitive, the model is the same to support vector regression model.

4.1.9 Passive-Aggressive Regression Model

The passive-aggressive algorithm in large-scale learning, which is similar to the perception that they do not require a learning rate. This is an online algorithm, which learns from massive streams of data. The whole procedure is getting an example first, then updating the classifier and throwing away examples iteratively.

4.1.10 Robust Regression Model

Robust regression models include Random Sample Consensus(RANSAC), Theil Sen and Huber in this project.

Robust regression aims to fit a regression model in the presence of corrupt data, either outlier in the model. In general, robust fitting in a high-dimensional setting(large n-features) is very

hard. Both Theil Sen and RANSAC are unlikely to be as robust as HuberRegressor for the default parameters. RANSAC will cope better with large outliers in the Y direction(a most common situation). Theil Sen will cope better with medium-size outliers in the X direction, but this property will disappear in high-dimensional settings.

RANSAC is good for strong outliers in the y-direction. Theil Sen is good for small outliers, both in direction X and Y, but has a breakpoint which performs worse than ordinary least squares. The scores of HuberRegressor may not be compared directly to both TheilSen and RANSAC because it does not attempt to completely filter the outliers but lessen their effect.
ThielSen Regression estimator uses a generalization of the median in multiple dimensions. It is thus robust to multivariate outliers but performs no better than ordinary least squares in high dimension. Theil-Sen is a median-based estimator.
The Huber regression model does not ignore the effect of the outliers but gives a lesser weight to them, and should be more efficient to use on data with a small number of samples.

4.1.11 Kernel Ridge Regression Model

Kernel ridge regression combines ridge regression and classification, which is identical to support vector regression(SVR) and uses squared error loss.

4.1.12 Support Vector Regression Model

The kernel is an important hyper-parameter in support vector regression, including RBF, linear, poly and sigmoid. LinearSVR is the same as the support vector regression with a linear kernel.

4.1.13 K Nearest Neighbor Regression Model

Neighbours-based regression can be used in cases where the data labels are continuous rather than discrete variables.
There are several hyper-parameters in the KNN regression model. The significant one is K, determining how many neighbours should be considered during prediction. In addition, these neighbours can be weighted according to the distance. In another way, prediction can be done according to the radius. KNN regression model is a lazy learner, meaning this is unnecessary to be trained before prediction.

4.1.14 Gaussian Process Regression Model

Gaussian process regression model implements Gaussian Process for regression purposes.

4.1.15 Decision Tree Regression Model

Decision tree regression is a non-parametric supervised learning method and the goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

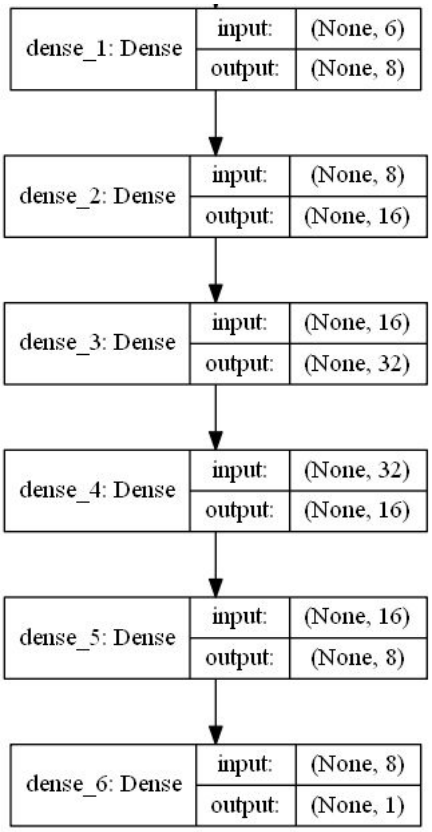
4.1.16 Voting Regression Model

The voting regression model combines several models together and returns the average prediction of all models. In these projects, we have chosen ARDRegression, Huber Regression, RadiusNeighborsRegressor, LassoLars, LinearRegression(degree 2, non-bias, false interaction-only), LassoLarsIC with AIC criterion. These models are relatively better than other models, so it is reasonable to combine them together.

4.2 Regression Model Evaluation

4.2.1 Neural Network: Multi-Layer Perceptron

In order to find a regression model that was capable of accurately predicting the price of real estate, we also employed a multi-layer perceptron(MLP) and compared its performance with that of other 16 regression models to determine which model was more appropriate for this regression task. With the help of the Deep learning framework of Tensorflow and Keras package, We have designed and optimized the structure of the network. As shown in Figure below, the network consisted of 6 fully connected layers, including one input layer, one output layer and 4 hidden layers. For the input layer, it was used to receive original data. In this first layer, it includes 8 neurons and uses Relu as the activation function. For 4 hidden layers, the number of neurons in each layer is 16, 32, 16 and 8 separately, and Relu is used as the activation function. For the output layer, since our task was a regression problem, there is only one neuron in the last layer using “linear” as the activation function.



4.2.2 Regression Models Evaluation Metrics

1. Mean Squared Error(MSE)

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

2. Mean Absolute Error(MAE)

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i|.$$

   If the dataset has outliers and we can make sure they are outliers, we can use MAE as the evaluation metric because it is more robust to outliers than MSE.

3. Explained Variance Score

$$explained\_variance(y, \hat{y}) = 1 - \frac{Var\{y - \hat{y}\}}{Var\{y\}}$$

   (Var means Variance)
   The best possible score is 1.0, and the lower the values, the worse the model.

4. Max Error

$$\text{Max Error}(y, \hat{y}) = max(|y_i - \hat{y}_i|)$$

   Max Error captures the worst-case error.

5. Median Absolute Error

$$\text{MedAE}(y, \hat{y}) = \text{median}(|\ y_1 - \hat{y}_1\ |, \ldots, |\ y_n - \hat{y}_n\ |).$$

   Median absolute error is robust to outliers because the loss is calculated by taking the median of all absolute differences between the target and the prediction.

6. Root Mean Squared Error
   The square root of mean squared error.

7. R2 Score

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

   R2 score can be negative because the model can be arbitrarily worse and the best score is 1.0.

8. Mean Absolute Percentage Error

$$MAPE = \frac{100\%}{N} \sum_{i=0}^{N} \frac{|y_i - \hat{y}_i|}{\hat{y}_i}$$

This evaluation metric considers large values that can range wider than small values. For example, the error of the target value is 10 and the prediction is 9 is the same as the target value is 100 and the prediction is 90.

9. Mean Squared Percentage Error

$$MSPE = \frac{100\%}{N} \sum_{i=0}^{N} \frac{(y_i - \hat{y}_i)^2}{\hat{y}_i}$$

This evaluation metric shows large values range relatively less compared with the function of MAPE.

10. Mean Squared Logarithmic Error(MSLE)

$$\text{MSLE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (\log_e(1 + y_i) - \log_e(1 + \hat{y}_i))^2.$$

There is a condition during using MSLE, which is that all values(both target and prediction) should be positive. In addition, this metric penalizes an under-predicted estimate greater than an over-predicted estimate. In our view, this metric works similarly to MSPE. In real estate prediction tasks, if the house price is higher, the bias between prediction and the exact price would have more tolerance than low house prices. This metric is probably the most reasonable for the real estate regression problem, so we decided to use this metric as our final metric to compare with various regression models.

11. Root Mean Squared Logarithmic Error(RMSLE)
    The same condition as MSLE and this is the square root of MSLE.

4.2.3 Regression Models Comparison

For experiments, in order to compare the performance of MLP and other regression models, we also used 75% of the dataset for training and the rest for testing. Besides, we employed backpropagation to update the weights in the neural network. For backpropagation, We select mean square error(MSE) as the loss function, because after comparing the results of different loss functions, such as mean absolute error(MAE), mean square error(MSE) and so on, the result of MSE was better than that of the others. For optimizer, we chose Nadam, which was similar to Adam, but performed better in this task. And other hyper-parameters were also fine-tuned, for example, our final epochs were 300 and the batch size was 68, and we also used bias in hidden layers. For evaluation, there were several evaluating methods for regression tasks, such as MAE, MSE, MSLE, R Square and so on. In order to compare the

performance of MLP with that of other regression models, we decided to use MSLE which was used to evaluate regression models in this section.

After the experiment, we compared the performance of different models. MLSE was employed as the main evaluating method to decide which model was the best. As shown in the first table below, we listed the result of each model, and we could find that the result of the MLP was 0.052, which was the lowest in the table. Therefore, we could consider MLP as the best model for this regression task. In addition, we also explored the performance of the same model with different parameters. Taking K nearest neighbour(KNN) as an example, as shown in the second table below, KNN was sensitive to the initial k, when k was greater than 3, the value of MLSE started to increase and the performance of the model gradually declined. Weights were another key factor for KNN. In the Scikit-learn package, the value of weights was mainly divided into two categories, including "uniform" and "distance". "Uniform" means all nearest neighbour samples having the same weight, and as for "distance", the farther the distance from the sample was, the smaller the weight was. Finally, we found that when the value of k was small, such as 1 and 3, "uniform" and "distance" had the same effect on the model. However, with the increment of k, the performance of the model with "uniform" was decreasing significantly. Therefore, KNN with "uniform" was more sensitive to the value of k.

Models	**MSLE**
ARD Regression (Automatic Relevance Determination)	0.070
Bayesian Ridge Regression	0.069
Gaussian Process Regression	2.243
10-fold cross validation Ridge Regression	0.068
SGD Regression	0.069
Theil Sen Robustness Regression	0.197
Decision Tree Regression	0.065
10-fold cross validation Elastic Net	0.068
Huber Robustness Regression	0.064
KNN Regression (k = 3, uniform weight)	0.054
Kernel Ridge Regression	0.075
Lasso regression (Akaike information criterion)	0.070
Lasso regression (Bayes information criterion)	0.070
10-fold cross validation Lasso Regression	0.068
Lasso Regression (alpha=0.05)	0.068
Least Angle Regression (alpha=0.05)	0.068
Linear Regression (degree =3, bias=False. interception=False)	**0.061**
Passive Aggressive Regression	0.063
Ridge Regression (alpha=5)	0.068
Random Sample Consensus Robustness Regression	0.067
Support Vector Regression(kernel=linear)	0.066
Voting Regression	**0.061**
Multilayer Perceptron	**0.052**

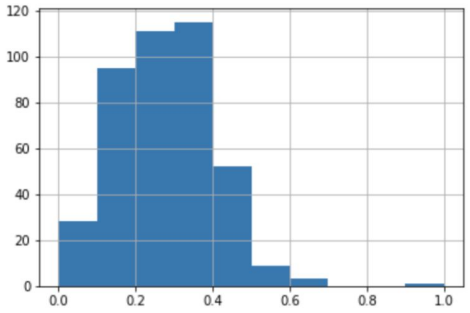
	MSLE
KNN Regression (k = 1, uniform)	0.075
KNN Regression (k = 1, distance)	0.075
KNN Regression (k = 3, uniform)	0.054
KNN Regression (k = 3, distance)	0.055
KNN Regression (k = 5, uniform)	0.059
KNN Regression (k = 5, distance)	0.056
KNN Regression (k = 7, uniform)	0.060
KNN Regression (k = 7, distance)	0.057
KNN Regression (k = 9, uniform)	0.061
KNN Regression (k = 9, distance)	0.057
KNN Regression (k = 11, uniform)	0.065
KNN Regression (k = 11, distance)	0.058

5 Classification

5.1 Data preparation for classification

First of all, *Min-Max* normalization is applied, so that every value is scaled between 0 and 1. If we want to apply supervised classification methods in the original dataset, the Y value (house price per unit area) should be set bins to make labels. Bins are set according to the distribution of the normalized price value, and then we can transform house price values into four classes: low, medium, moderately high and high.

The following figures show the distribution of price values and the binning standard of the house price per unit area:



0.0 ~ .2	low
.2 ~ .4	Medium
.4 ~ .6	Moderately High
.6 ~ 1.0	High

Binning Standard

5.2 Feature Selection

We can improve the model by feeding in only those features that are uncorrelated and non-redundant[1]. This is mainly why feature selection works. It helps in decreasing the training time and also reducing the complexity. In this section, two feature selection algorithms are applied: the Boruta algorithm and the tree-based algorithm.

The Boruta algorithm is a wrapper built around the random forest classification algorithm[2]. It tries to capture all the important, interesting features you might have in your dataset with respect to an outcome variable. Features selected by Boruta algorithm are: 'X2 house age', 'X3 distance to the nearest MRT station', 'X5 latitude', 'X6 longitude'.

Tree-based estimators can be used to compute feature importance, which in turn can be used to discard irrelevant features[3]. Features selected by tree-based estimators are: 'X2 house age', 'X3 distance to the nearest MRT station', 'X6 longitude'.

5.3 Resampling

Since the distribution of classes is skewed, two types of resampling methods are applied: oversampling and undersampling. The basic idea of oversampling is to add a random set of copies of minority class examples to the data, so the minority class is oversampled by random oversampling(ROS), while the main idea of undersampling is to delete some examples from

the majority class, so the majority class is undersampled by repeated edited nearest neighbours(RENN).

After oversampling, each class has 226 samples each. After undersampling, class distribution of undersampling with train_set_boruta is:

[('High', 4), ('Low', 84), ('Medium', 123), ('Moderately High', 23)].

Class distribution of undersampling with tr_set is:

[('High', 4), ('Low', 80), ('Medium', 112), ('Moderately High', 23)].

5.4 Datasets and Classification Models

After the feature selection and now we have 5 datasets in total, the following table shows the description of these 5 datasets:

Dataset	Total instance	Description
Original	414	No resampled and no feature selection
Ros_Boruta	904	ROS resampled and Boruta selection
Ros_Tr	904	ROS resampled and Tree-based selection
Renn_Boruta	234	RENN resampled and Boruta selection
Renn_Tr	219	RENN resampled and Tree-based selection

Table: Dataset Description

where ROS resampled is over-sampling and RENN resampled is under-sampling.

The applied four classification models in the Scikit-learn package are decision tree classifier of the tree mode, LinearSVC using the support vector machine algorithm, GaussianNB based on naive Bayes algorithm and Adaboost under the category of ensemble model.

5.5 Multi-class Classifier Evaluation

5.5.1 Evaluation Criteria

After all these four models have been trained against five datasets, a reasonable measurement should be decided to evaluate the performance of different models. In this imbalanced dataset, there are many evaluation metrics that we could use, like accuracy, precision, recall, F1-score, etc. Among them, the F1 score could be a good choice because it combines both precision and recall, both of which are insensitive about true negatives. The definition of F1 score is:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

For the F1-score of multi-class evaluation, there exist two types of F1-score: micro-average one and macro-average one. The micro-F1 score treats instances equally, while the other one treats classes equally, but in this dataset, each class's number of instances varies a lot, so the

micro-average method is better and micro-average F1 score is selected as the evaluation criteria.

	DT	LSVC	GNB	ADB
Original	0.7952	0.7590	0.7108	0.8072
Ros_Boruta	0.8785	0.5525	0.7348	0.5856
Ros_Tr	0.8619	0.5304	0.6243	0.5470
Renn_Boruta	1.0000	0.8511	0.9149	0.5745
Renn_Tr	0.9545	0.8182	0.9318	0.6364

F1 score of models

The preliminary comparison shows that, for some certain models like Decision Tree and GaussianNB, the F1 score generated from the original dataset is lower than other datasets, which implies that, for these two classifiers, feature selection and resampling, to some extent, optimize the performance. For LinearSVC, over-sampling decreases classification performance. For Adaboost, the model trained by the original dataset performs better than the one trained by feature selected and resampling ones. The comparison also shows that Decision Tree has the highest F1 score on average, then followed by GaussianNB and LinearSVC. Adaboost performs the worst in the measurement of the F1 score.

5.5.2 Friedman Test

If we want to compare k algorithms over n data sets, we need to use specialized significance tests to avoid that our confidence level drops with each additional pairwise comparison between algorithms[4]. The Friedman test is suitable for this situation.

We ranked the performance of all these 4 algorithms per data set, from the best performance (rank 1) to worst performance (rank k). The following table shows ranks of F1 score of all k algorithms per data set:

	DT	LSVC	GNB	ADB
Original	2	3	4	1
Ros_Boruta	1	4	2	3
Ros_Tr	1	4	2	3
Renn_Boruta	1	3	2	4
Renn_Tr	1	3	2	4
avg rank	1.2	3.4	2.4	3

Ranks of F1 score

The definition of the null hypothesis and alternative hypothesis for the Friedman test:
If P_i refers to the performance of the algorithm i, the null hypothesis H_0 and alternative hypothesis H_alpha for the test are:

$$H_0 : P_1 = P_2 = ... = P_k$$

$$H_\alpha : There\, is\, at\, least\, one\, pair\, of\, algorithm\, (i, j) such\, that\, P_i \neq P_j$$

According to the Friedman Test, three quantities are calculated to figure out the Friedman statistic:

$$\bar{R} = \frac{1+k}{2} = 2.5 \quad , \quad n\sum_j (R_j - \overline{R})^2 = 13.8 \quad , \quad \frac{1}{n(k-1)}\sum_{ij}(R_{ij} - \bar{R})^2 = 1.67$$

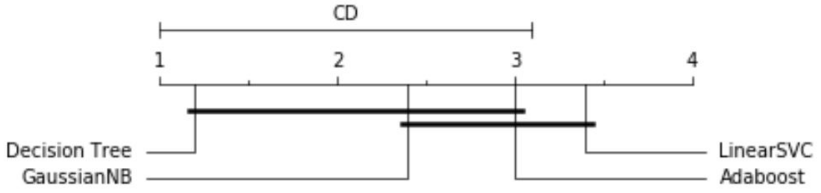
The Friedman statistic equals the second quantity divided by the third one, which is 8.26. The critical value for k = 4, n = 5 at the α = 0.05 level is (α = 0.05, degree of freedom = 3) 7.81473, so we reject the null hypothesis that all algorithms perform equally and turn to the alternative hypothesis which means there exists at least one pair of algorithms, at the 0.05 significance level, shows a significant difference.

5.5.3 Nemenyi Test

If we want to perform further analysis on a pairwise level. We can use the Nemenyi test. The main idea is that if we compare the critical difference(CD) value with the difference of average rank between every two algorithms. If the difference exceeds CD, then they have a significant difference. The definition of CD in the Nemenyi test is as follows:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}$$

q_α depends on the significance level α and k, it is 2.569 in this case, so CD = 2.098.



CD diagram for the pairwise Nemenyi test

By comparing the difference between algorithms and the length of CD in the CD diagram, we can tell that the performance of the top-ranked algorithm “Decision Tree” is significantly better than the bottom one “LinearSVC” and slightly better than the other two.

6 Conclusion and Future work

In this project, our group members worked together to analyze the real estate dataset and used various machine learning and data mining technologies to handle several problems. Due to the limitation of the size of this dataset, we should find larger reality datasets with more features and columns and tackle the real problem according to the procedure we have done in this project.

7 References

[1]M. Pathak, "Boruta Feature Selection in R", DataCamp Community, 2018. [Online]. Available: https://www.datacamp.com/community/tutorials/feature-selection-R-boruta. [Accessed 16 Mar 2020]

[2]M. Kursa and W. Rudnicki, "Feature Selection with the Boruta Package", *Journal of Statistical Software*, vol. 36, no. 11, p. 2, 2010. [Accessed 16 Mar 2020]

[3]"1.13. Feature selection — scikit-learn 0.22.2 documentation", *Scikit-learn.org*. [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html. [Accessed 20 Mar 2020]

[4]P. Flach, 2012, Machine Learning: The Art and Science of Algorithms that Make Sense of Data, 2nd ed. Cambridge, p. 355.