HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

---------***----------

MAJOR ASSIGNMENT REPORT

MODULE: PROBABILITY AND STATISTICS

CODE: MT2013

Students:

TRẦN NGUYỄN GIA PHÁT

NGUYỄN PHÚC ANH

HỒ ANH DŨNG

NGUYỄN KHÔI NGUYÊN

Cohort: CC01

Semester: 221

Academic year: 2022-2023

Lecturer: PHAN THỊ HƯỜNG

Submission date: 4th December 2022

FOR EXAMINERS ONLY

Grade (in number):

………………..

Grade (in words):

………………..

Examiner 1

(Signature & Full name)

………………..

Examiner 2

(Signature & Full name)

………………..

Ho Chi Minh City, December 2022

STATEMENT OF AUTHORSHIP

Except where reference is made in the text of the report, this assignment contains no material published elsewhere or extracted in a whole or in part from an assignment which we have submitted or qualified for or been awarded another degree or diploma.

No other person's work has been used without the acknowledgments in the report.

This report has not been submitted for the evaluation of any other models or the award of any degree or diploma in other tertiary institutions.

Ho Chi Minh City, December 2022

ACKNOWLEDGEMENT

We would like to express my gratitude towards Ms Phan Thi Huong, for helping us to coordinate in writing this report. We have explored meaningful findings during this work.

We are also very grateful to article writers on the internet for providing priceless knowledge about the field we are doing. Without their contributions, this report may not be done completely.

DISCLAIMER

For a concise explanation, only the snippets containing necessary calculations are shown and may differ from the final code. Comments are minimized for readability. Due to page limitations, the report only covers some of the source codes' contents. Therefore, an investigation of the attached files is recommended for better understanding. The results has been embedded into the notebook at the first openning.

Advanced prediction models which do not belong to the scope of this score will be summarized by how their work as concepts instead of in-depth algorithm explanations.

Please only run the codes provided in the Appendices for examination. Readers may look up essential installation, required libraries, and guidelines in the same section.

ABOUT THE AUTHORS

· Trần Nguyễn Gia Phát	**2153681**
· Nguyễn Phúc Anh	**2153166**
· Hồ Anh Dũng	**2052921**
· Nguyễn Khôi Nguyên	**2152198**

INTRODUCTION

Statistics and probability are the branches of mathematics that deal with the collection and analysis of data. Probability is the study of chance, a fundamental discipline we use daily. At the same time, statistics are more concerned with how data is processed using various methods of analysis and collection techniques. Since these two subjects are always closely related, it is only possible to study one by studying the other.

In this report, the writers will give solutions for problems involving (a) Exploratory Data Analysis, (b) graphing data, (c) analysis of variance, and (d) linear regression model.

This report also provides code for building different prediction and classification models using statistical optimization methods and machine learning. Predictive modeling is a mathematical process used to predict future events or outcomes by analyzing patterns in a given input data set. It is a crucial part of predictive analytics, which uses current and historical data to forecast activity, behavior, and trends in business and research.

TABLE OF CONTENT

I. METHODOLOGY

1.1. Exploratory Data Analysis (EDA)

Exploratory data analysis (EDA) is a methodology for obtaining insights from data, often using data visualization and statistical graphics to help identify patterns and trends, detect outliers and understand the relationships between variables. EDA is used to extract essential features for the prediction models.

Graphing the raw data allows us to have a general idea of the behavior and distribution of the variables:

- **Histograms** are used to visualize the distribution of a numerical variable.
- **Box plots** are used to show distributions of numeric data values, especially when you want to compare them between multiple groups.
- **Pair plots** are used to understand the best features to explain a relationship between two variables or to form the most separated clusters.
- **Correlation Matrix:** A correlation matrix is a table that shows the correlation coefficients between variables. Each cell in the table shows the relationship between two variables.

1.2. Analysis of Variance (ANOVA)

An **ANOVA** test is a statistical test used to determine whether there is a statistically significant difference between two or more categorical groups by testing the difference in means using the variance.

One-way **ANOVA** has a categorical independent variable (a factor) and a normally distributed continuous dependent variable (such as the range or ratio level). Independent variables divide items into two or more mutually exclusive levels, categories, or groups.

Hypothesis testing for **ANOVA** can generally be stated as follows:

- $\boldsymbol{H_0}$ (Null hypothesis): There is no difference in means between groups of independent subjects.

- $\boldsymbol{H_1}$ (Alternative hypothesis): There are two or more independent groups with statistically different means.

Some assumptions to consider before running an **ANOVA** test:

1. The sample population is normally distributed.
2. Homogeneity of variance between groups of independent variables (the variance is approximately the same for each group of samples).
3. All sample observations must be independent of each other.

After you have run an **ANOVA** and found significant results, you can run Tukey's HSD to find out which specific groups' means (compared with each other) are different.

Tukey's Honest Significant Difference (HSD) test is a post-hoc test based on the studentized range distribution. The test compares all possible pairs of means. Any absolute difference between means has to exceed the value of HSD to be statistically significant.

The **Tukey HSD** test statistic formula is:

$$HSD = \frac{M_i - M_j}{\sqrt{\frac{MS_w}{n_h}}}$$

where

- $\boldsymbol{M_i - M_j}$ is the difference between the means of pair groups.
- $\boldsymbol{MS_w}$ is the Mean Square Within and $\boldsymbol{n}$ is the number in each category.

1.3. Multivariate Linear Regression (MLR)

Most data analyzed statistically does not necessarily have a response variable or an explanatory variable. Multivariate regression is a technique used to measure the relationship between a continuous dependent variable and two or more independent variables. Relationships that result from correlations between variables are called linear. After applying multivariate regression to a data set, we use this technique to predict the behavior of a response variable based on its predictors.

The null hypothesis and alternative hypothesis could be expressed as:

- $\boldsymbol{H_0}$ (Null hypothesis):: There is no relationship between explanatory variables and response variable.
- $\boldsymbol{H_1}$ (Alternative hypothesis): There is at least one explanatory variable is associated linearly with the response variable.

The general equation of MLR:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_0 X_n + \epsilon$$

where

- $\boldsymbol{Y}$ is the dependent variable.
- $\boldsymbol{X_i}$ is the i^{th} independent variable.
- $\boldsymbol{\beta_0}$ is the intercept of Y when all X_i are zeroes.
- $\boldsymbol{\beta_i}$ is the coefficient for each X_i.
- $\boldsymbol{\epsilon}$ is the independent error term for the model.

Model performance metrics:

- **R-squared (R2)**: is the portion of the variation in the outcome explained by the predictor, is the squared correlation between the observed outcome value and the value predicted by the model. The higher the value, the better the model.
- **Root mean square error (RMSE):** measures the average error made by a model in predicting an observation. The lower the RMSE, the better the model.

Diagnostic plots of a linear regression model:

1. **Residuals vs. Leverage Plot**: This plot is used to identify influential observations. If any point on this graph is outside of Cook's distance (dotted line), it is an influential observation.
2. **Scale-Location Plot**: This plot is used to test the assumption of equal variance among the residuals of a regression model (homoscedasticity). If the red line is roughly horizontal across the graph, the assumption of equal variances may be correct.
3. **Normal Q-Q Plot**: This plot is used to determine whether the residuals of a regression model follow a normal distribution. If the points on this graph fall

roughly along a straight diagonal line, we can assume that the residuals are normally distributed.

4. **Residuals vs. Fitted Plot**: This plot is used to determine if the residuals show a non-linear pattern. If the red line in the middle of the graph is roughly horizontal, you can assume that the residuals follow a linear pattern.

1.4. Stepwise Regression

Stepwise regression is a technique that allows us to build a regression model from a set of predictor variables by introducing them stepwise into the model and dropping them until there is no longer a statistically valid reason for their entry or deletion.

There are three types of stepwise regression:

- **Forward** Stepwise Selection
- **Backward** Stepwise Selection
- **Both-Direction** Stepwise Selection

The **Akaike Information Criterion (AIC)** estimates prediction errors in the sample and is similar to adjusted R-squared measures in regression output summaries. It effectively penalizes us for adding more variables to the model. Lower scores may indicate a more fitted model than a higher AIC model.

1.5. Classification models

1.5.1. Logistic Regression

Logistic regression is a supervised machine learning algorithm that performs binary classification tasks by predicting the probability of an outcome, event, or observation. Logistic regression uses a logistic function called a sigmoid function to map predictions and their probabilities.

The following equation is the formula of logistic regression:

$$y = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_0 X_n)}}$$

where

- $\boldsymbol{y}$ is the predicted output.
- $\boldsymbol{X_i}$ is the i^{th} independent variable.
- $\boldsymbol{\beta_0}$ is the intercept term.
- $\boldsymbol{\beta_i}$ is the coefficient for each X_i.

1.5.2. Gradient Boosting Machine

Gradient Boosting is a machine learning boosting approach representing a decision tree for vast and complex data. It is based on the assumption that the next potential model will reduce the gross prediction error when merged with the last set of models. The decision trees are used to provide the most accurate predictions possible.

There are three factors of an gradient boosting model:

1. A loss function to be optimized.
2. A weak learner to make predictions.
3. An additive model to add weak learners to minimize the loss function.

1.5.3. Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost) is a scalable, distributed gradient-boosted decision tree (GBDT) machine learning framework. XGBoost is an improvement of the Gradient Boosting Algorithm. It is the top machine-learning library for regression, classification, and ranking tasks and supports parallel tree boosting.

Some of the advantages of XGBoost are:

- Due to the parallel processing process, it has faster performance than gradient boosting.
- It controls the overfitting problem.
- It gives a better performance result on many datasets.
- It is used for classification, regression, and ranking with custom loss functions.

1.5.4. Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised learning algorithm that uses a linear separating hyperplane to predict the class labels of training data. An SVM classifies data by finding the best hyperplane that separates all data points in one class from data in another class (the best hyperplane in an SVM is the most significant margin between two classes).

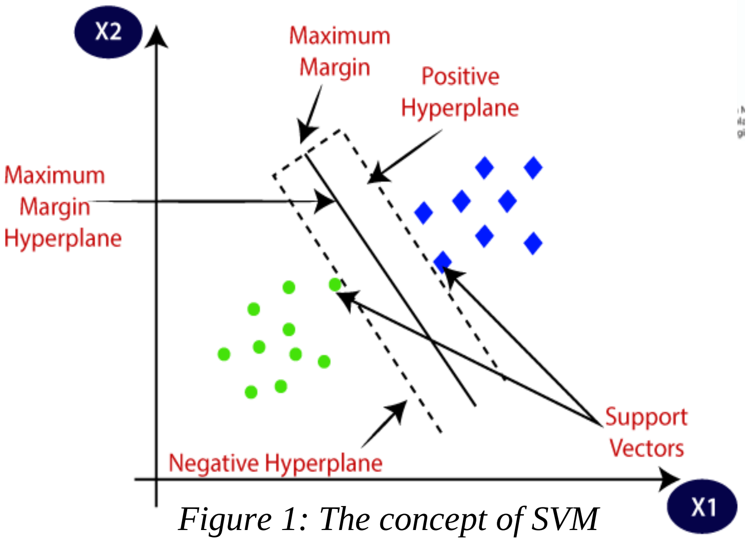


Figure 1: The concept of SVM

SVM is widely used because of its high performance, flexibility, and efficiency.

1.5.5. Random Forest

Random Forest is a powerful and versatile supervised machine learning algorithm that grows and combines multiple decision trees to create a "forest". It uses binning and randomization to build each individual tree and tries to create a forest of uncorrelated trees whose committee predictions are more accurate than individual trees.

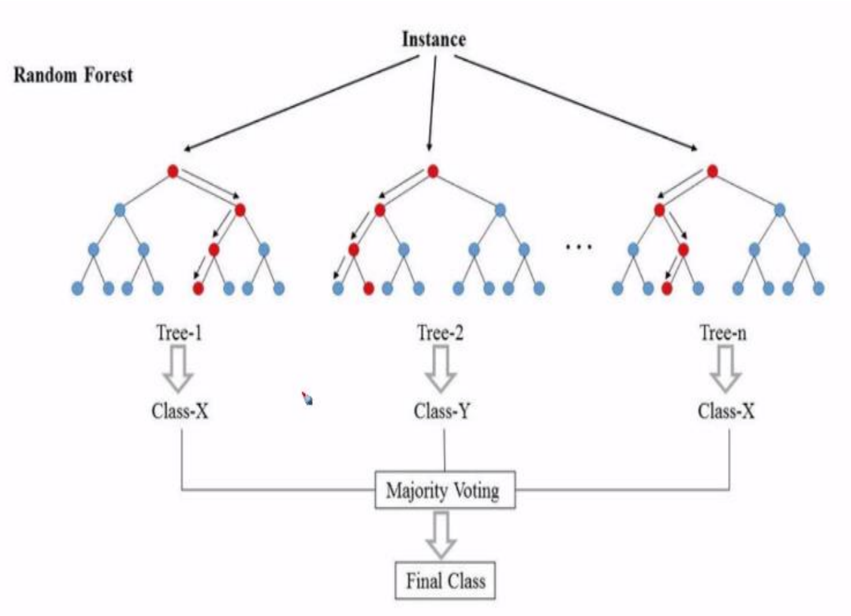


Figure 2: The concept of Random Forest

The logic of the random forest model is that multiple uncorrelated models (individual decision trees) perform much better as a group than alone. When using a Random Forest for classification, each tree provides a vote. The forest chooses the classification with the majority of the votes. Individual decision trees can produce errors, but most groups are correct, moving the overall outcome in the right direction.

1.5.6. K – Nearest Neighbours (KNN)

The K-Nearest Neighbor (KNN) algorithm is a supervised nonparametric classifier that uses proximity to make predictions about clusters of individual data points. It can be used for both regression and classification problems but is often used as a classification algorithm, assuming that similar points can be found close together.

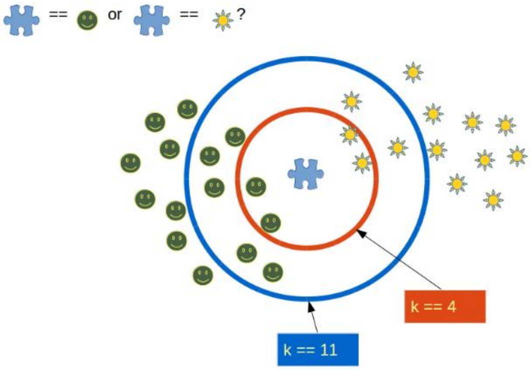
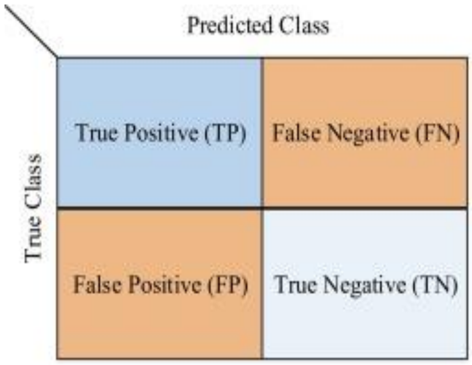


Figure 3: The concept of KNN

1.6. Confusion Matrix

The confusion matrix is an N x N matrix used to evaluate the performance of a classification model, where N is the number of target classes. The matrix compares the actual target value with the value predicted by the machine learning model.



	Predicted Class	
True Class	True Positive (TP)	False Negative (FN)
	False Positive (FP)	True Negative (TN)

Figure 4: Elements of a confusion matrix

Model performance metrics:

1. A good model is one with high TP and TN ratios and low FP and FN ratios.
2. When you have an unbalanced data set, it is best to use a confusion matrix as a criterion for evaluating machine learning models.

1.7. ROC Curve and AUC

A receiver operating characteristic (ROC) curve is a graph showing a rating model's performance at all rating thresholds. This curve represents the actual positive rate and the false positive rate.

AUC stands for "area under the ROC curve." AUC measures cumulative performance across all possible classification thresholds. One way to interpret the AUC is to see it as the probability that the model overestimates a random positive sample over a random negative sample.

II. ACTIVITY 1

2.1. Dataset Description

The table below contains basic information of the dataset:

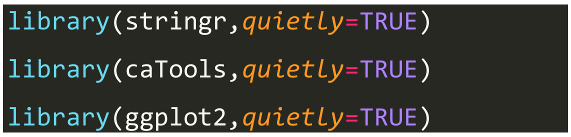
Title	KC_Housesales_Data
Source	https://www.kaggle.com/swathiachath/kc-housesales-data
Purpose	Predict House sale prices using Multi-Linear Regression
Description	The dataset was obtained from Kaggle. The dataset contains house sold prices and house details in King County, an area in the Washington state of the US, from May 2014 to May 2015.

Variable description table:

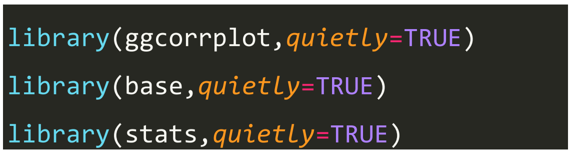
Variable	**Description**	**Variable**	**Description**
date	Date of house was sold	condition	1=poor; 5=excellent
price	Price sold	grade	Grade given by King County: 1=poor; 5=excellent
bedrooms	Number of bedroom	sqft_above	Square footage of house apart from basement
bathrooms	Number of bathroom per bedroom	sqrt_basement	Basement square footage
sqft_living	House square footage	yr_built	House built year
sqft_lot	Lot square footage	yr_renovated	House renovated year
floors	Floor level	zipcode	House zip code
waterfront	House has a view to a waterfront	lat	House latitude coordinate
view	House has been viewed	long	House longitude coordinate

2.2. Working with R

a) Import libraries



```
library(stringr,quietly=TRUE)
library(caTools,quietly=TRUE)
library(ggplot2,quietly=TRUE)
```



```
library(ggcorrplot,quietly=TRUE)
library(base,quietly=TRUE)
library(stats,quietly=TRUE)
library(reshape2,quietly=TRUE)
library(caret,quietly=TRUE)
library(leaps,quietly=TRUE)
library(dplyr,quietly=TRUE)
library(MASS,quietly=TRUE)
```

```
library(GGally,quietly=TRUE)
library(randomcoloR,quietly=TRUE)
library(car,quietly=TRUE)
library(ggpubr,quietly=TRUE)
library(tidyverse,quietly=TRUE
```

b) Load and clean data

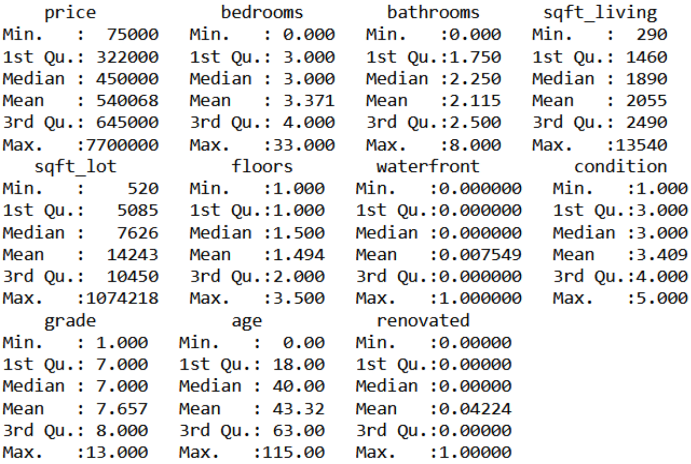
```
data <- read.csv('./Activity 1.csv')
data<-na.omit(data)

data$date <- str_sub(data$date, 1, 4) #Extract the year from date of building sold

data$date <- as.numeric(data$date) #changing it to numeric for subtraction
operation
data$age <- ifelse(data$date - data$yr_built>0,data$date - data$yr_built,0) #age
of building when it was sold
data$sqft_living <-ifelse(data$date==2014,data$sqft_living,data$sqft_living15)
#sqft living when the house was sold
data$sqft_lot <-ifelse(data$date==2014,data$sqft_lot,data$sqft_lot15) #sqft living
when the house was sold
data$view<-NULL
data$sqft_basement<-NULL
data$sqft_above<-NULL
data$sqft_living15<-NULL
data$sqft_lot15<-NULL
data$date <- NULL
data$yr_built <- NULL
data$X.2 <- NULL
data$X.1 <- NULL
data$X <- NULL
data$id <- NULL
data$renovated <- ifelse(data$yr_renovated>0,1,0)
data$yr_renovated <-NULL
data$zipcode <-NULL
data$long <-NULL
data$lat <- NULL
```

First we load dataset to ***data*** variable, fill out any missing value, transform some data fields and then remove unnecessary features by setting them to *NULL*.

c) Exploratory Data Analysis

```
summary(data)
```

We use ***summary()*** function to sum up basic information of the dataset



```
     price            bedrooms        bathrooms      sqft_living   
 Min.   :  75000   Min.   : 0.000   Min.   :0.000   Min.   :  290  
 1st Qu.: 322000   1st Qu.: 3.000   1st Qu.:1.750   1st Qu.: 1460  
 Median : 450000   Median : 3.000   Median :2.250   Median : 1890  
 Mean   : 540068   Mean   : 3.371   Mean   :2.115   Mean   : 2055  
 3rd Qu.: 645000   3rd Qu.: 4.000   3rd Qu.:2.500   3rd Qu.: 2490  
 Max.   :7700000   Max.   :33.000   Max.   :8.000   Max.   :13540  
    sqft_lot           floors        waterfront          condition    
 Min.   :    520   Min.   :1.000   Min.   :0.000000   Min.   :1.000  
 1st Qu.:   5085   1st Qu.:1.000   1st Qu.:0.000000   1st Qu.:3.000  
 Median :   7626   Median :1.500   Median :0.000000   Median :3.000  
 Mean   :  14243   Mean   :1.494   Mean   :0.007549   Mean   :3.409  
 3rd Qu.:  10450   3rd Qu.:2.000   3rd Qu.:0.000000   3rd Qu.:4.000  
 Max.   :1074218   Max.   :3.500   Max.   :1.000000   Max.   :5.000  
     grade             age           renovated      
 Min.   : 1.000   Min.   :  0.00   Min.   :0.00000  
 1st Qu.: 7.000   1st Qu.: 18.00   1st Qu.:0.00000  
 Median : 7.000   Median : 40.00   Median :0.00000  
 Mean   : 7.657   Mean   : 43.32   Mean   :0.04224  
 3rd Qu.: 8.000   3rd Qu.: 63.00   3rd Qu.:0.00000  
 Max.   :13.000   Max.   :115.00   Max.   :1.00000  
```

Figure 5: Summary statistics of main features

```
variables=colnames(data)
ggpairs(data, columns= variables)
```

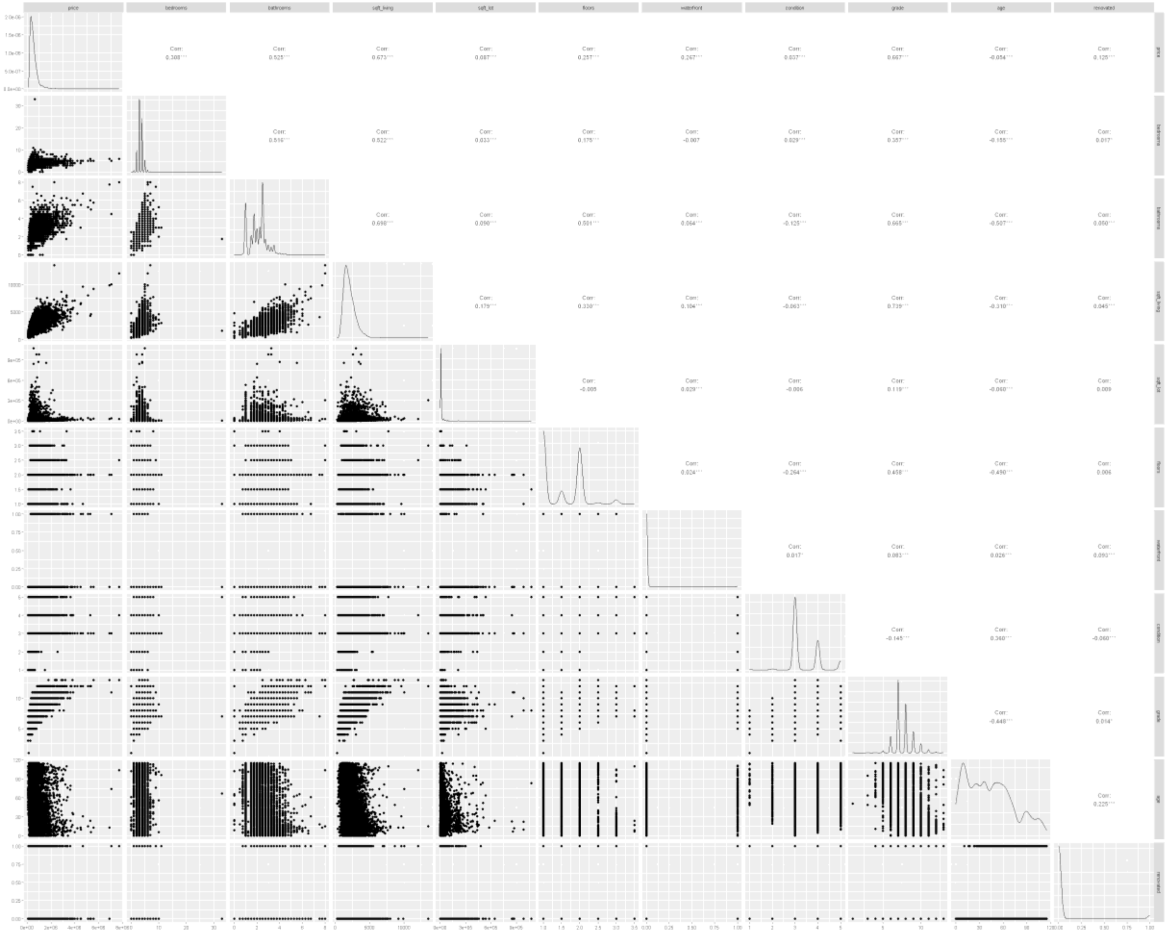


Figure 6: Pair plots between each two variables

Next we compare every pair of available variables by ***ggpairs(****data,columns****)***, which gives a detailed plot of the relation between each group.

```r
for(var in variables){
    temp <-gsub(",","", data[var]);
    temp <-as.numeric(unlist(data[var]));
    hist(temp,
         main=var,
         col = randomColor(),
         xlab="Units",
         freq = FALSE
            );
}
```

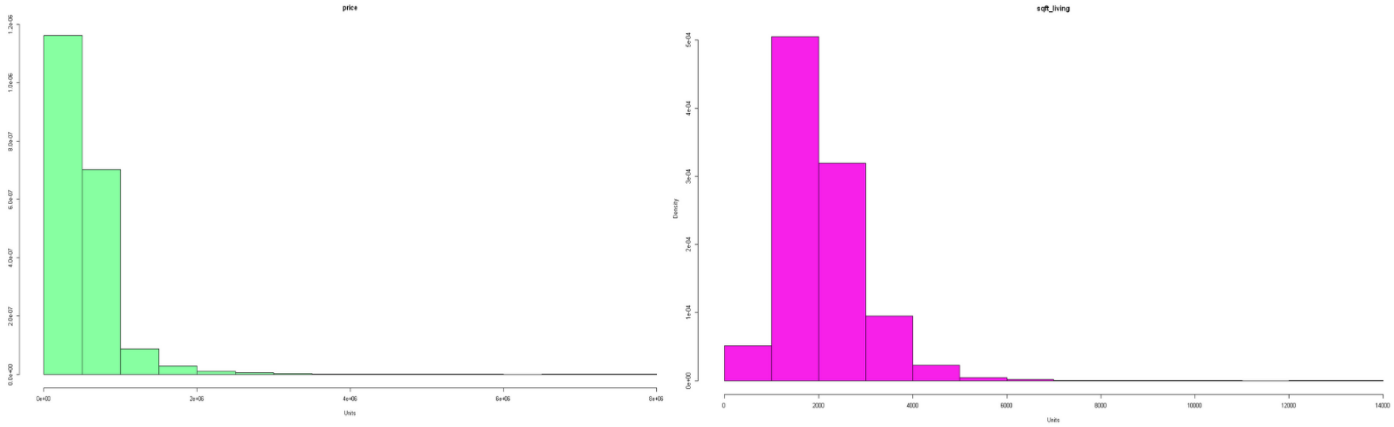


Figure 7: Distribution of Price and Sqft_living using histogram

For each column saved in ***variables,*** we cast it to the numeric data type and plot a histogram using ***hist()***. (See above parameters)

The remain graphs with the same type can be found in the attached source code.

Note that in the complete code, the colors for elements in plots will be random at each run time because of the ***randomColor()*** function.

```r
for ( i in variables){
boxplot(data[i],
        col="orange",
        xlab = i,
        cex.lab=1.5,
        border="brown")
   }
```

Using ***boxplot()*** for each column, we obtain the result:

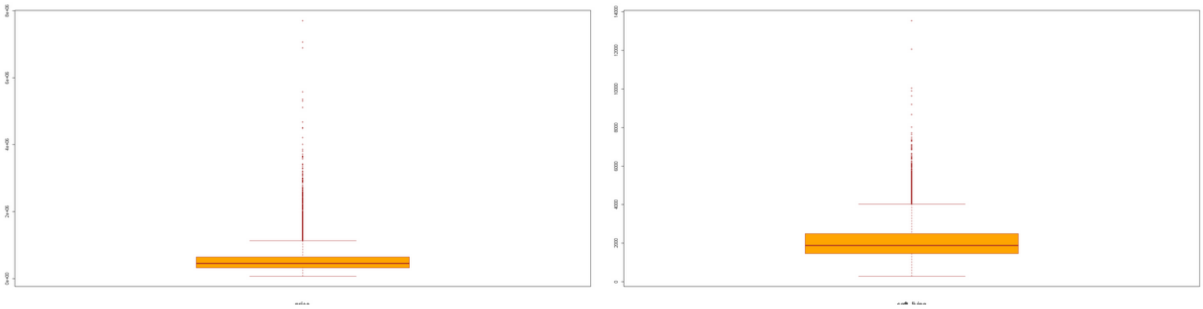


Figure 8: Price and Sqft_living plotted using Boxplot

The plots are quite small in size as of display issues, readers may double click in the graph in *Jupyter Notebook* or rewrite the code in *R Studio* for better viewing.

d) Split data into separate subsets

```
#make this split test reproducible
set.seed(42)
#use 70% of dataset as training set and 30% as test set
split <- sample.split(data,SplitRatio = 0.70)
train_df <- subset(data,split==TRUE)
test_df <- subset(data,split==FALSE)
```

We split the full dataset into train set and test set. It is good practice for training prediction models since it helps to avoid overfitting of the model. (See above snippet to know how to split randomly)

d) Perform ANOVA test

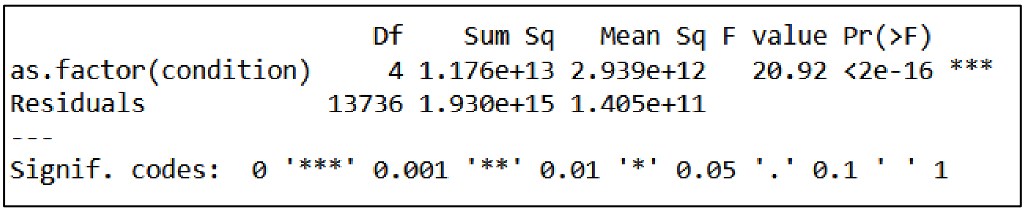
For **Price and Condition**, we want to know that whether different levels of **Condition** affect the house price.

Hypothesis:

- $\boldsymbol{H_0}$: The mean price is equal for all levels of **condition**.
- $\boldsymbol{H_1}$: At least one of the **condition** level has a mean **price** that is not the same as the other **condition** level.

```
one_way <- aov(price ~ as.factor(condition), data = train_df)
summary(one_way)
```

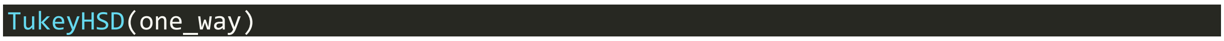
Using ***aov()***, we store the test result into ***one_way*** variable, then summary the information using ***summary()***:



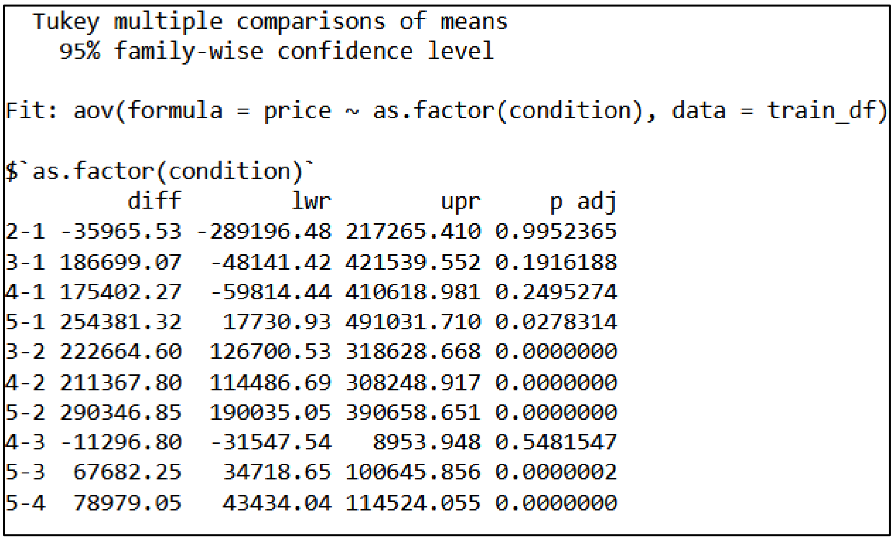
```
                       Df    Sum Sq   Mean Sq F value Pr(>F)    
as.factor(condition)     4 1.176e+13 2.939e+12   20.92 <2e-16 ***
Residuals            13736 1.930e+15 1.405e+11                   
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 9: Summary result of ANOVA test (Price vs Condition)

The p-value of the **condition** variable is low (p < 0.001, indicated by the ’***’), which implies that the **condition** rate impacts the house sold price. Hence the H_0 is rejected. There is a significant difference in the average price of house based on the **condition** of house.



```
TukeyHSD(one_way)
```



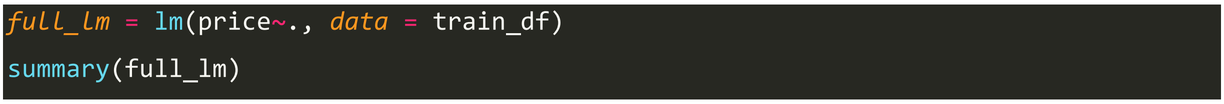
```
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = price ~ as.factor(condition), data = train_df)

$`as.factor(condition)`
         diff        lwr        upr     p adj
2-1 -35965.53 -289196.48 217265.410 0.9952365
3-1 186699.07  -48141.42 421539.552 0.1916188
4-1 175402.27  -59814.44 410618.981 0.2495274
5-1 254381.32   17730.93 491031.710 0.0278314
3-2 222664.60  126700.53 318628.668 0.0000000
4-2 211367.80  114486.69 308248.917 0.0000000
5-2 290346.85  190035.05 390658.651 0.0000000
4-3 -11296.80  -31547.54   8953.948 0.5481547
5-3  67682.25   34718.65 100645.856 0.0000002
5-4  78979.05   43434.04 114524.055 0.0000000
```

Figure 10: Tukey’HSD test to determine the difference between each two condition levels

According to the results, there are statistically significant differences (p < 0.05) between **condition** groups 5-1 (5 and 1), 3-2, 4-2, 5-2, 5-3 and 5-4. The difference between groups 2-1, 3-1, 4-1 and 4-3 are not statistically significant.

The same tests for **Price vs. Grade** and **Price vs. Renovation** can be performed similarly. Therefore, we will not include them in this report. Readers may find the results in ‘*Activity 1.ipynb*’ file.

e) Fitting linear regression model

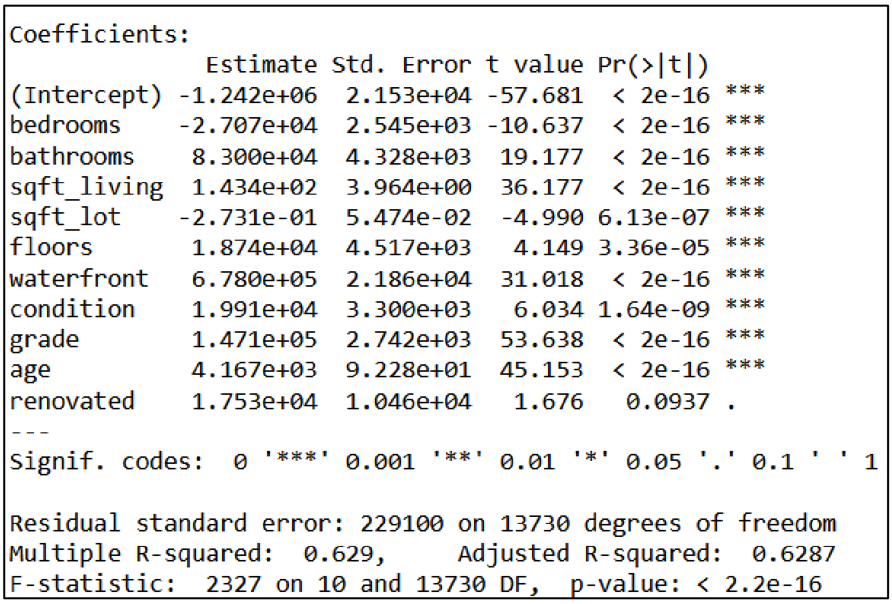


```
full_lm = lm(price~., data = train_df)
summary(full_lm)
```

First, we examine with all the features in the train dataset and summary the result

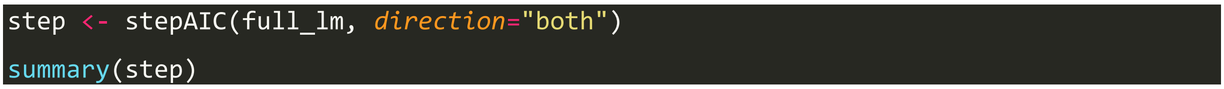
The complete model of the multiple linear regression model is significant, with 0.629 as the adjusted R-squared value.

Now we will apply stepwise regression to reduce the number of predictors as much as possible.



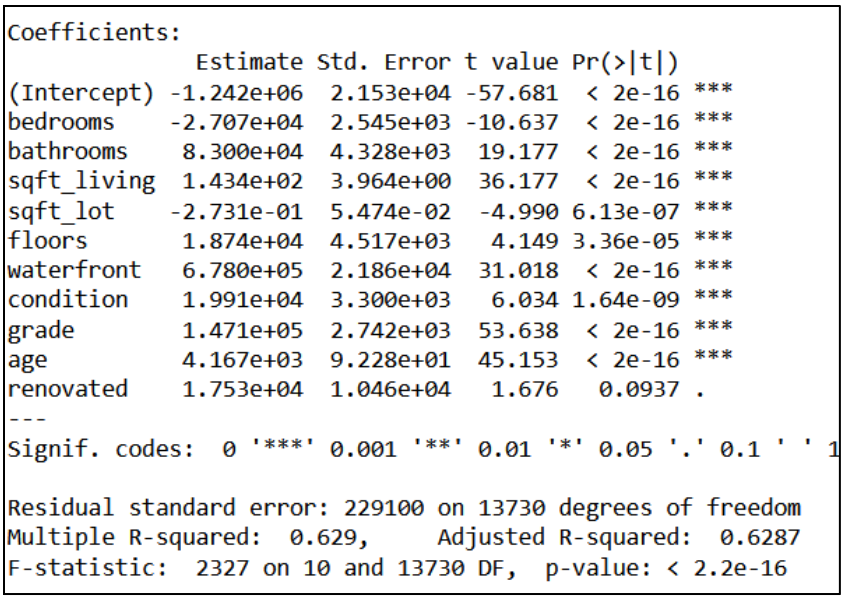
```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.242e+06  2.153e+04 -57.681  < 2e-16 ***
bedrooms    -2.707e+04  2.545e+03 -10.637  < 2e-16 ***
bathrooms    8.300e+04  4.328e+03  19.177  < 2e-16 ***
sqft_living  1.434e+02  3.964e+00  36.177  < 2e-16 ***
sqft_lot    -2.731e-01  5.474e-02  -4.990 6.13e-07 ***
floors       1.874e+04  4.517e+03   4.149 3.36e-05 ***
waterfront   6.780e+05  2.186e+04  31.018  < 2e-16 ***
condition    1.991e+04  3.300e+03   6.034 1.64e-09 ***
grade        1.471e+05  2.742e+03  53.638  < 2e-16 ***
age          4.167e+03  9.228e+01  45.153  < 2e-16 ***
renovated    1.753e+04  1.046e+04   1.676   0.0937 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 229100 on 13730 degrees of freedom
Multiple R-squared:  0.629,	Adjusted R-squared:  0.6287 
F-statistic:  2327 on 10 and 13730 DF,  p-value: < 2.2e-16
```

Figure 11: Full linear regression model



```
step <- stepAIC(full_lm, direction="both")
summary(step)
```

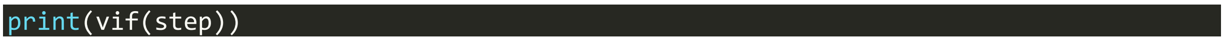
Notice that the R-squared value is still the same. The stepwise regression also shows that all variables should be retained in the model.



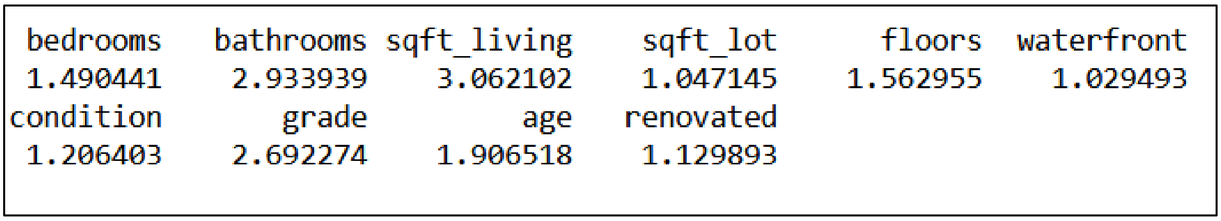
```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.242e+06  2.153e+04 -57.681  < 2e-16 ***
bedrooms    -2.707e+04  2.545e+03 -10.637  < 2e-16 ***
bathrooms    8.300e+04  4.328e+03  19.177  < 2e-16 ***
sqft_living  1.434e+02  3.964e+00  36.177  < 2e-16 ***
sqft_lot    -2.731e-01  5.474e-02  -4.990 6.13e-07 ***
floors       1.874e+04  4.517e+03   4.149 3.36e-05 ***
waterfront   6.780e+05  2.186e+04  31.018  < 2e-16 ***
condition    1.991e+04  3.300e+03   6.034 1.64e-09 ***
grade        1.471e+05  2.742e+03  53.638  < 2e-16 ***
age          4.167e+03  9.228e+01  45.153  < 2e-16 ***
renovated    1.753e+04  1.046e+04   1.676   0.0937 .  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 229100 on 13730 degrees of freedom
Multiple R-squared:  0.629,	Adjusted R-squared:  0.6287 
F-statistic:  2327 on 10 and 13730 DF,  p-value: < 2.2e-16
```

Figure 12: Optimized linear regression model using stepwise

Multicollinearity check using ***vif()***:



```
print(vif(step))
```



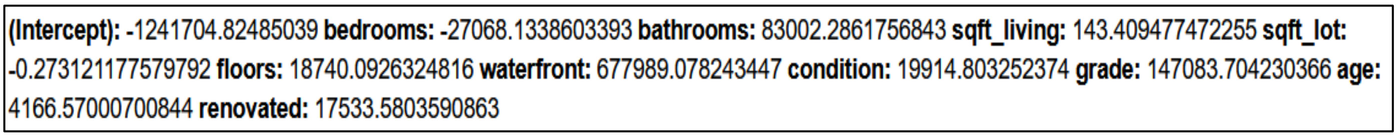
```
   bedrooms   bathrooms sqft_living    sqft_lot      floors  waterfront 
   1.490441    2.933939    3.062102    1.047145    1.562955    1.029493 
  condition       grade         age   renovated 
   1.206403    2.692274    1.906518    1.129893 
```

Figure 13: Multicollinearity check

Small VIF values, VIF < 3, indicate a low correlation among variables under ideal conditions. The default VIF cutoff value is 5; only variables with a VIF less than five will be included in the model. If VIF is above 10, it indicates serious multicollinearity between the predictors. There is no multicollinearity in this model.

Final model:

```
final_lm <-step
final_lm$coefficients
```



(Intercept): -1241704.82485039 **bedrooms:** -27068.1338603393 **bathrooms:** 83002.2861756843 **sqft_living:** 143.409477472255 **sqft_lot:** -0.273121177579792 **floors:** 18740.0926324816 **waterfront:** 677989.078243447 **condition:** 19914.803252374 **grade:** 147083.704230366 **age:** 4166.57000700844 **renovated:** 17533.5803590863

Figure 14: Final model's coefficients for each variable

e) Model evaluation

```
plot(final_lm)
```

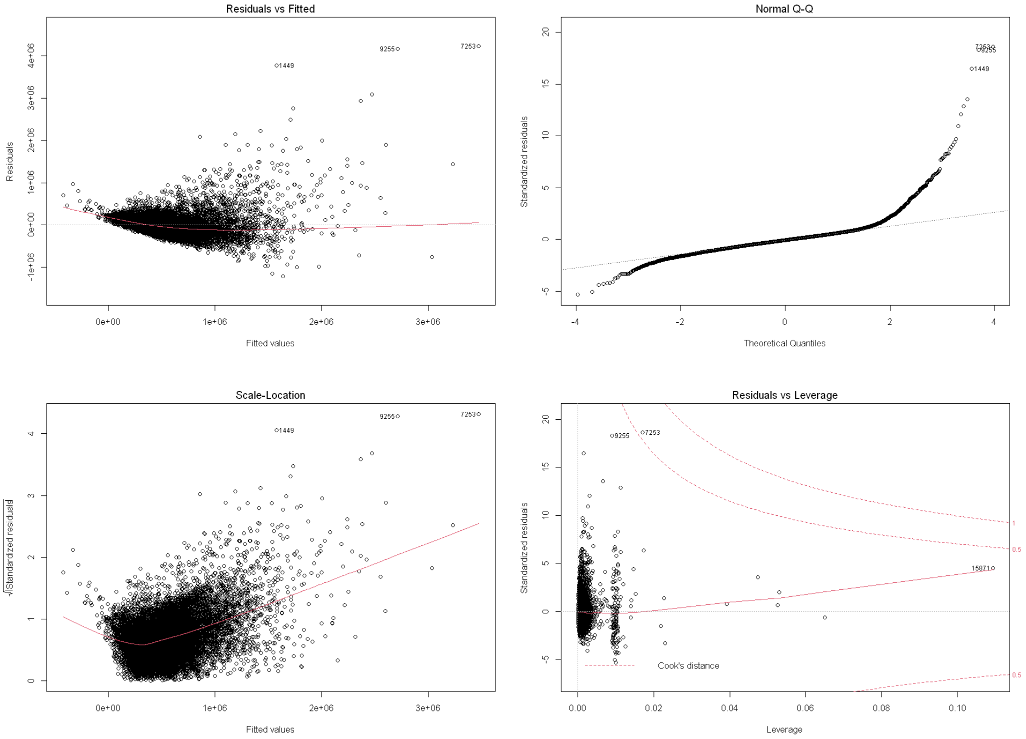
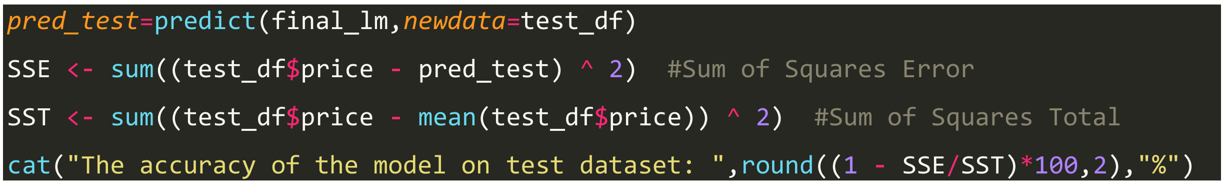


Figure 15: Diagnostic plots of linear regression model

- **Residual vs Fitted Plot** shows scatted values around indicated the relationship between price and predictors is linear.
- **Normal Q-Q Plot** shows a straight line indicated the residuals are normally distributed.
- **Scale-Location Plot** verifies that there is no clear pattern among the residuals, however the red line is not roughly horizontal, therefore this model is not fully reliable.
- **Residuals vs Leverage Plot** shows there are no leverage out of the border of Cook’s distance (the red dashed lines), hence our regression model does not contain any influential points



```r
pred_test=predict(final_lm,newdata=test_df)
SSE <- sum((test_df$price - pred_test) ^ 2)  #Sum of Squares Error
SST <- sum((test_df$price - mean(test_df$price)) ^ 2)  #Sum of Squares Total
cat("The accuracy of the model on test dataset: ",round((1 - SSE/SST)*100,2),"%")
```

The accuracy of the model on test dataset: 63.97 %

In conclusion,

The provided Multiple Linear Regression model shows that **bedrooms, bathrooms, sqft_living, sqft_lot, floors , waterfront, condition, grade** and **age** are significant factors affecting house price. The prediction model can predict the house price in King Country by taking into account the nine variables above at approximately 60-70% accuracy.

However, this accuracy level could be better for a prediction model because linear regression is simple and not the best option for some datasets. We will try out advanced models in the next session and compare them.

III. ACTIVITY 2

3.1. Dataset Description

The table below contains basic information of the dataset:

Title	Credit Card Fraud Detection
Source	https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud
Purpose	Recognize fraudulent credit card transactions
Description	The dataset contains transactions made by credit cards in September 2013 by European cardholders, presents transactions that occurred in two days, where occured 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

3.2. Working with R

a) Import libraries

Same as Activity 1, but with additional machine learning libraries

```
library(xgboost,quietly=TRUE)
library(gbm, quietly=TRUE)
library(randomForest, quietly=TRUE)
library(LogicReg,quietly=TRUE)
library(class,quietly=TRUE)
library(e1071,quietly=TRUE)
library(pROC)
library(ROCR)
library(ROSE)
```

b) Load and clean data

```
data <- read.csv('./Activity 2.csv')
data<-na.omit(data)
data$Time<-NULL
data = data %>% distinct()
data$Class <- as.factor(data$Class)
```

c) Exploratory Data Analysis

```
correlations <- cor(data,method="pearson")
```

```
corrplot(correlations, number.cex = .9, method = "circle", type = "full",
tl.cex=0.8,tl.col = "black")
```

We see that most of the data features are uncorrelated. This is because many characteristics undergo Principal Component Analysis (PCA) algorithms before being published. Features V1 to V28 can be principal components extracted from the actual feature distribution by PCA.

The top 3 independent variables {V12, V14, V17} may partly contain important information about whether a giving credit card transaction is fraudulent or not.

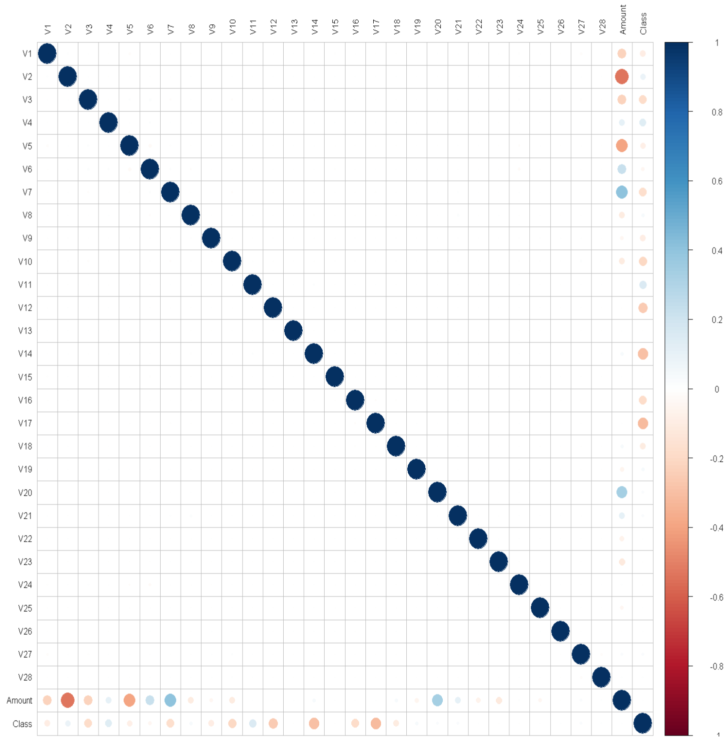


Figure 16: Correlation Matrix of dataset

```
class = table(data$Class)
lb = paste0(round(prop.table(class)*100,2),"%")
pie(class,labels = lb, col =  colors,main="Percentage of Valid vs Fraudulent
transaction")leged(x=1,y=1,legend=c("Valid","Fraud"),fill= randomColor(2), bty =
"n")
```

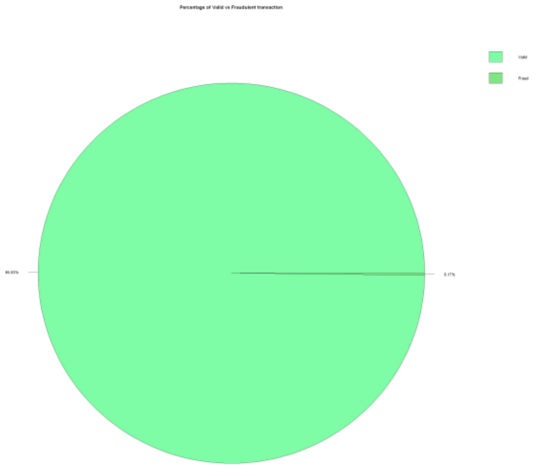


Figure 17: Class imbalance (Fraud vs non-fraud)

d) Fitting various models

We will fit different classification models into the dataset and evaluate their AUC scores; from that, we will choose the model with the highest AUC value.

Before fitting, notice that the dataset is heavily imbalanced, so we need a sampling method to avoid overfitting. (Dataset splitting is as same as Activity 1).

Upsampling

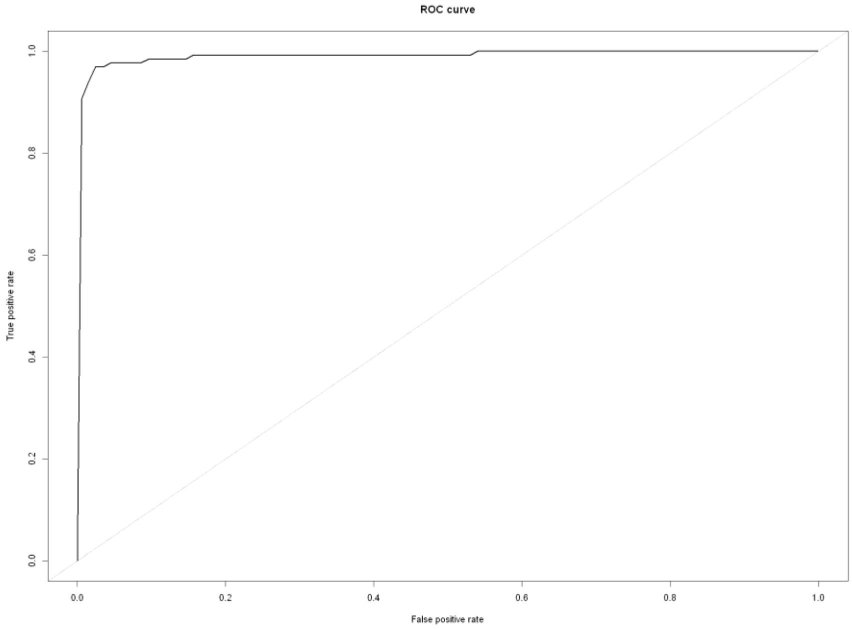
This method works with the minority class. It reproduces the observations of the minority class to balance the data. It is also known as oversampling.

An advantage of this method is that no information is lost. The downside of using this method is that because oversampling simply adds replicate observations to the original data set, it adds more observations of different types and leads to overfitting. This method can only be used on the train data set.

```
up_train <- upSample(x = train_df[, -ncol(train_df)],y = train_df$Class)
```

Logicstic Regression

```
model_lr <- glm(Class~.,data=up_train, family='binomial')
predict_lr = predict(model_lr, newdata = test_df,type = "response")
roc.curve(test_df$Class, predict_lr, plotit =TRUE)
```

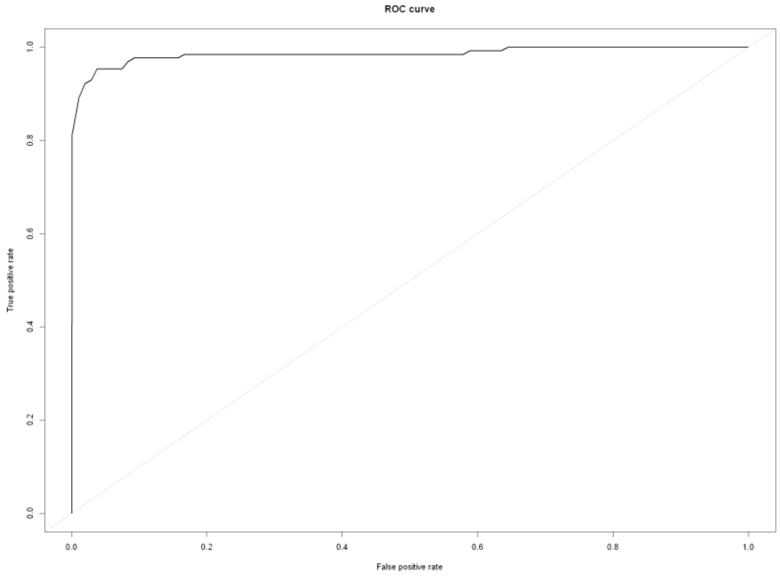


Area under the curve (AUC): 0.990

Figure 18: ROC curve of Logistic Regression

XGBoost

```
model_xgb <- xgboost(data = xgboost_train,
max.depth=3,verbose = FALSE, nrounds=50)
predict_xgb = predict(model_xgb, newdata=xgboost_test)
roc.curve(test_df$Class, predict_xgb, plotit =TRUE)
```

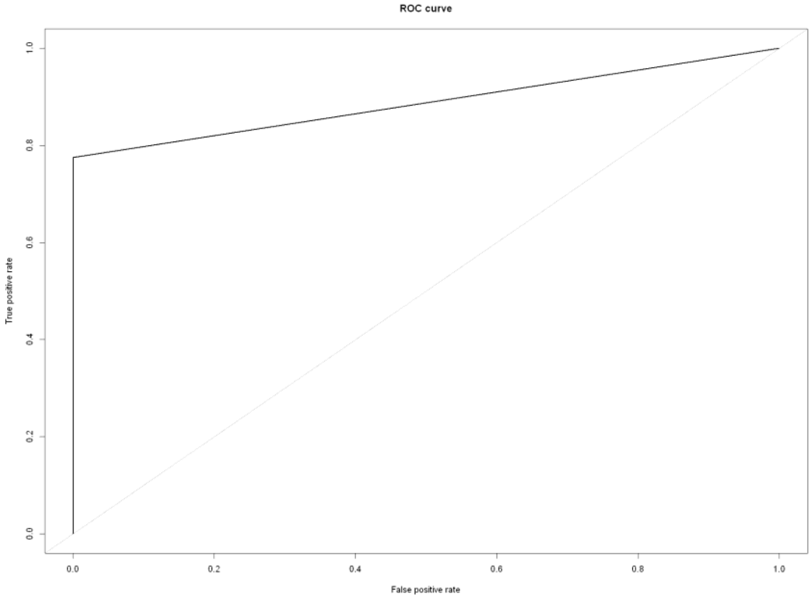


```
Area under
the curve
(AUC):
0.985
```

Figure 19: ROC curve of XGBoost

Random Forest

```
model_rf = randomForest(x = up_train[-30],y = up_train$Class,ntree = 100)
predict_rf = predict(model_rf, newdata = test_df)
roc.curve(test_df$Class, predict_rf, plotit =TRUE)
```



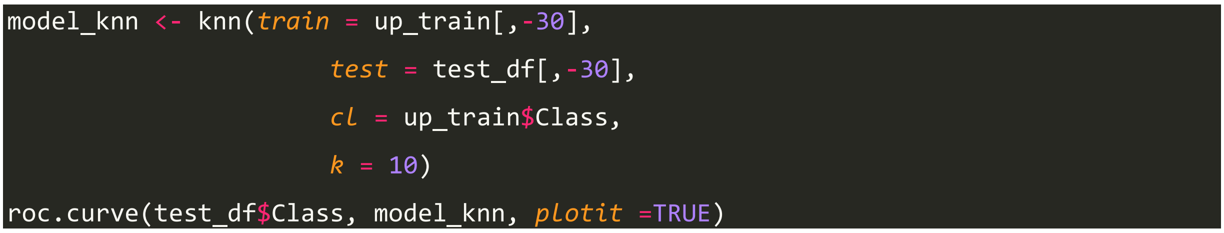
```
Area under
the curve
(AUC):
0.985
```

Figure 20: ROC curve of Random Forest

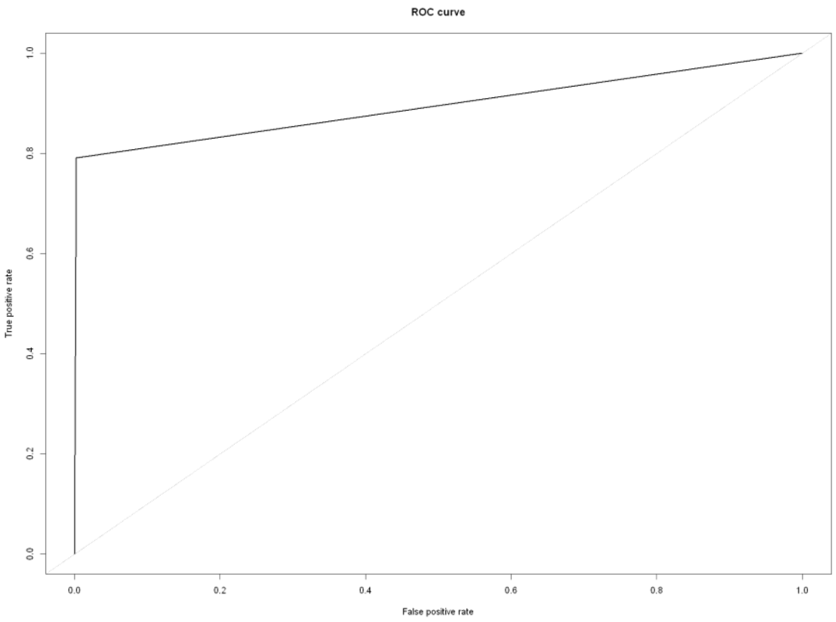
Support Vector Machine

Due to the high training time of the algorithm because of extensive input train data, despite the help of parallel computing speedup, we cannot demonstrate the model in the report. However, we keep the SVM section in the source file (codes are commented on). Readers who wish to perform SVM can give it a try, given enough patience, time, and required process memory.

K-Nearest Neighbours (KNN)



```
model_knn <- knn(train = up_train[,-30],
                      test = test_df[,-30],
                      cl = up_train$Class,
                      k = 10)
roc.curve(test_df$Class, model_knn, plotit =TRUE)
```



Area under the curve (AUC): 0.894

Figure 31: ROC curve of KNN

The best AUC score of 0.990 was achieved using a Logistic Regression model though both Random Forest and XGBoost models performed well. Therefore, the Logistic Regression model best suits this classification problem.

IV. CONCLUSION

In the real world, the accuracy of prediction models has always been better, usually falling between 70-90%. However, there are some constraints with prediction models in general. One of them is that the models use past data to make predictions for the future; this assumes that the past is an accurate representation of the future. For specific contexts, this might not be the case; for example, using historical data to predict the weather or a natural disaster could have devastating consequences if the predictions are incorrect. That said, future predictors are still extremely useful in many situations, and different models have been developed to tackle specific problems.

While accuracy figures for individual prediction models can fluctuate from project to project depending on the context in which they are used, there are some critical considerations for how they should be used to improve the outcomes of the projects in which they are applied. For example, data must be clean and accurate to run an effective prediction model. If no data has been validated against the original objectives, then it is not meaningful to use in forecasting models. Understanding the strengths and weaknesses of different predictive methods and when each should be applied to achieve the most effective results is also essential.

While the models in this report return from reasonable to significant accuracy, there are some drawbacks during the implementation phases and execution of the final product:

- Some of the non-linear models are pretty slow in generating predictions (SVM, Random Forest)
- They require a considerable amount of training data to make predictions (k-NN)
- The linear regression model did not perform as well as some of the more complex models due to simplicity and therefore requires a large amount of input data to produce reliable results.

V. DATA SOURCES

Datasets are available at:

Activity 1: https://www.kaggle.com/swathiachath/kc-housesales-data

Activity 2: https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

Source code is available at:

https://github.com/Zaphat/Probability-Statistics-Project

VI. REFERENCES

1. Aspexit. (n.d.). *How to validate a predictive model*. Retrieved November 21, 2022, from https://www.aspexit.com/how-to-validate-a-predictive-model/
2. BioSTTS. (n.d.). *Post hoc tests – tukey HSD*. Retrieved November 21, 2022, from https://biostats.w.uib.no/post-hoc-tests-tukey-hsd/
3. Data Science without Code. (n.d.). *How to know if your machine learning model has good performance*. Retrieved November 21, 2022, from https://www.obviously.ai/post/machine-learning-model-performance
4. IBM. (n.d.). *What is the K-nearest neighbors algorithm* Retrieved November 21, 2022, from https://www.ibm.com/topics/knn
5. IHS. (n.d.). *Variance Inflation Factor (VIF)* . Retrieved November 21, 2022, from http://onlinehelp.ihs.com/Energy/AnalyticsExplorer/AE_6.0/Webhelp/content/analyticsexplorer/VIF.htm
6. Kaggle. (n.d.). *Credit card fraud detection with R + (sampling)*. Retrieved November 21, 2022, from https://www.kaggle.com/code/atharvaingle/credit-card-fraud-detection-with-r-sampling
7. MachineLearningMastery. (n.d.). *A gentle introduction to the gradient boosting algorithm for machine learning*. Retrieved November 21, 2022, from https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
8. Mathworks. (n.d.). *Choosing the Best Machine Learning Classification model and avoiding overfitting*. Retrieved November 21, 2022, from https://www.mathworks.com/campaigns/offers/next/choosing-the-best-machine-learning-classification-model-and-avoiding-overfitting.html
9. Medium. (n.d.). *How outliers can pose a problem in linear regression*. Retrieved November 21, 2022, from https://medium.com/swlh/how-outliers-can-pose-a-problem-in-linear-regression-1431c50a8e0
10. Medium. (n.d.). *What is a confusion matrix*. Retrieved November 21, 2022, from https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5

11. Newcastle University. (n.d.). *Coefficient of Determination, R-squared*. Retrieved November 21, 2022, from https://www.ncl.ac.uk/webtemplate/ask-assets/external/maths-resources/statistics/regression-and-correlation/coefficient-of-determination-r-squared.html

12. Spiceworks. (n.d.). *What is logistic regression*. Retrieved November 21, 2022, from https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

13. Statology. (n.d.). *A complete guide to stepwise regression in R*. Retrieved November 21, 2022, from https://www.statology.org/stepwise-regression-r/

14. Statology. (n.d.). *How to interpret diagnostic plots in R*. Retrieved November 21, 2022, from https://www.statology.org/diagnostic-plots-in-r/

15. Statology. (n.d.). *How to interpret residual standard error*. Retrieved November 21, 2022, from https://www.statology.org/how-to-interpret-residual-standard-error/

16. STHDA. (n.d.). *Regression Model Accuracy Metrics: R-square, AIC, BIC, CP and more*. Retrieved November 21, 2022, from http://www.sthda.com/english/articles/38-regression-model-validation/158-regression-model-accuracy-metrics-r-square-aic-bic-cp-and-more/

17. STHDA. (n.d.). *Simple linear regression in R*. Retrieved November 21, 2022, from http://www.sthda.com/english/articles/40-regression-analysis/167-simple-linear-regression-in-r/

18. The Analysis Factor. (n.d.). *Outliers: To drop or not to drop*. Retrieved November 21, 2022, from https://www.theanalysisfactor.com/outliers-to-drop-or-not-to-drop/

19. Towardsdatascience. (n.d.). *How to pick the right model*. Retrieved November 21, 2022, from https://towardsdatascience.com/a-simple-way-to-pick-the-right-model-d362272b453d

20. WallStreetMojo. (n.d.). *Heteroskedasticity*. Retrieved November 21, 2022, from https://www.wallstreetmojo.com/heteroskedasticity/

VII. APPENDICES

Download the source codes for Activity 1 and 2:

1. Go to https://github.com/Zaphat/Probability-Statistics-Project

2. Select Code, click on Download Zip and unzip the downloaded file.

Install Jupyter Notebook using Anaconda:

1. Go to: https://www.anaconda.com/ and download the latest version.
2. Install and open Anaconda, then launch Jupiter Notebook.

Install R environment:

1. Run Anaconda Prompt from the Start Menu as Administrator.

2. Type the following command then press Enter:

```
conda install -c r r-irkernel
```

Lauch the code:

1. From Start Menu browse for Anaconda folder and click on Jupyter Notebook, a web page will be opened in the default browser.
2. Navigate to where the code files are located, click on the file name to open.

3. From *Cells* choose *Run All*.

Install R libraries:

1. Run Anaconda Prompt from the Start Menu as Administrator.

2. Type the following command then press Enter:

```
conda install -c conda-forge r-XXX --y
```

where XXX is the name of the library package you want to install.