STAT4011 Project II
Data Analysis with Supervised Learning Algorithms

Name: King Yeung CHAN
SID: 1155119394

17/12/2020

Introduction

We often curious about the true parameters encountered in a problem. In statistic, estimation helps to fulfil our curiosity. However, static analysis may not perform well when we are facing a real-life matter. Fortunately, there are some dynamic approaches to improve such an issue. We want to learn the pattern of data instead of generalising the overall data behaviour. Statistical learning favours our ultra goal to predict the solutions of real-life tasks.

The project focuses on data analysis with supervised learning models on two famous datasets 'House' and 'Titanic'. We propose two sections in exploring the datasets. The first part will be the Regression Problem on estimating the final prices of houses, whereas the second part will be the Classification Problem for labelling the possible status of passengers, either survived or died. As the project aims to compare a variety of supervised learning models, we will focus on the implementation rather than the description of each model. Having said that, we still provide a brief introduction along with supplement from Wikipedia for the proposed models. Without further ado, let us dive into the analysis.

Preliminaries

We navigate the working directory to aid the data import. Also, we will report the cross-validation error and the prediction error of each model being used. Cross-validation is a method to prevent overfitting on a model by dividing the train data into k-folds and taking the average of mean squared error (MSE) for each fold. The prediction error is the squared distance between the actual value and the predicted value. Unlike the regression problem, the error measurement in the classification problem is based on the number of correct classification against the total number of prediction. Moreover, we evaluate the performance of the classification algorithms with the balanced accuracy in practice. The accuracy is defined as $F = \frac{1}{2} \times (\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$. Although it is out of scope for the project, we make a record for reference. Hence, we initialise 'Cross-Validation Error', 'Prediction Error' and 'Balanced Error' to store the corresponding errors.

```
setwd("/Users/jackchan/Documents/I go to school by bus/03 The Chinese University of Hong Kon
g/Year 3/Semester 1/STAT4011 Statistics Projects/Project II")

result = list(reg_pro = data.frame(`Cross-Validation Error` = rep(0, 4), `Prediction Error`
 = rep(0, 4)), cla_pro = data.frame(`Cross-Validation Error` = rep(0, 4), `Prediction Error`
= rep(0, 4), `Balanced Error` = rep(0, 4)))
rownames(result$reg_pro) = c("Linear Regression", "Lasso Regression", "Partial Least Squares
Regression", "Random Forest")
rownames(result$cla_pro) = c("k-Nearest Neighbors", "Logistic Regression", "Linear Discrimin
ant", "Boosting")
```

Regression Problem

As the name suggested, we consider a situation as a regression problem when the response variable (RV) is quantitative. We want to predict the value of RV given unseen data in explanatory variables (EV). In the regression problem, we will implement the analysis with 'House' data.

Data Background

```
house = read.csv("House.csv", header = TRUE)
dim(house)
```

```
## [1] 500  15
```

What is your dream house looks like? How would you describe your desired one? The 'House' data consists of 500 observations with house prices as RV and 15 EVs. It is no doubt that the RV is a quantitative one. We plan to fit different models and predict the final costs of houses. Before our modelling, we will kick off with the exploratory data analysis (EDA).

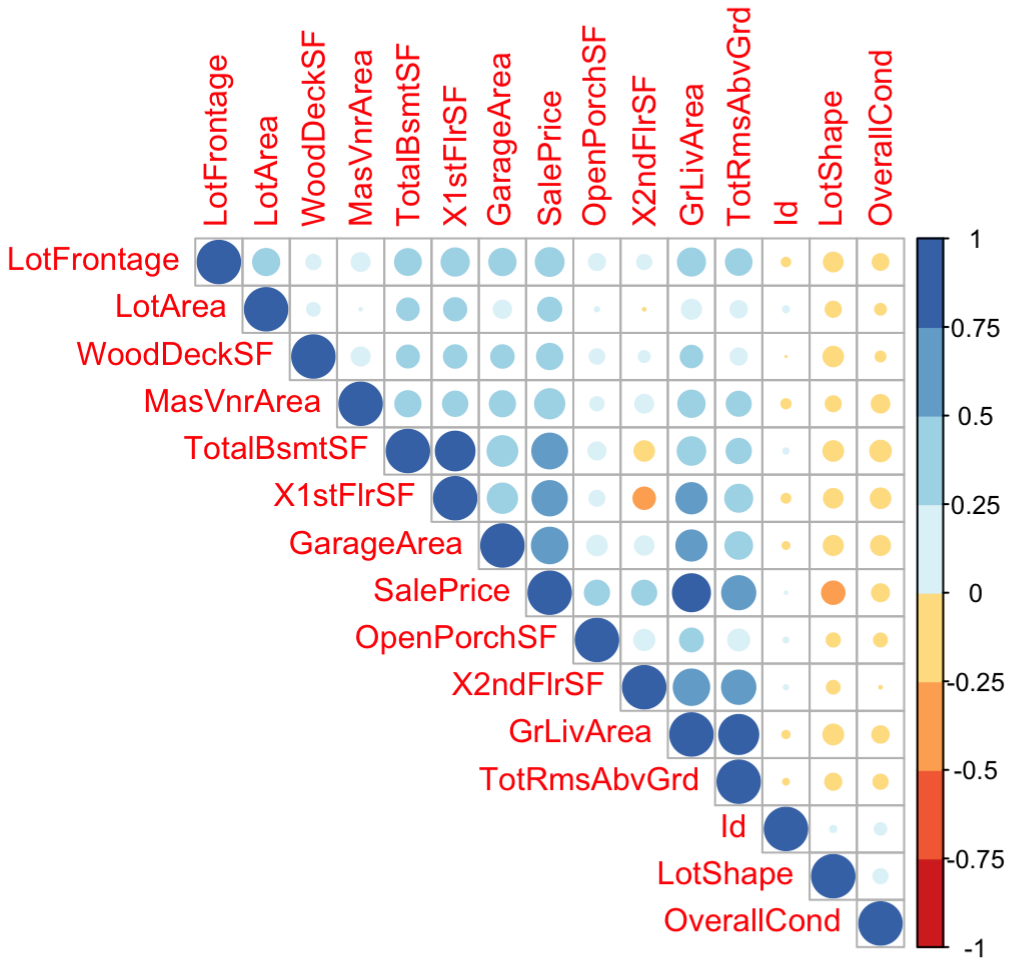
Exploratory Data Analysis

```
summary(house[, -1])
```

```
##   LotFrontage        LotArea        LotShape   OverallCond      MasVnrArea
##  Min.   : 21.00   Min.   :  1526   IR1:171   Min.   :1.000   Min.   :   0.0
##  1st Qu.: 60.00   1st Qu.:  7590   IR2: 16   1st Qu.:5.000   1st Qu.:   0.0
##  Median : 70.00   Median :  9375   IR3:  3   Median :5.000   Median :   0.0
##  Mean   : 70.96   Mean   : 11143   Reg:310   Mean   :5.552   Mean   : 112.7
##  3rd Qu.: 82.00   3rd Qu.: 11604             3rd Qu.:6.000   3rd Qu.: 180.0
##  Max.   :174.00   Max.   :215245             Max.   :9.000   Max.   :1600.0
##  NA's   :87                                                  NA's   :1
##   TotalBsmtSF     X1stFlrSF      X2ndFlrSF        GrLivArea     TotRmsAbvGrd
##  Min.   :   0   Min.   : 483   Min.   :   0.0   Min.   : 520   Min.   : 3.000
##  1st Qu.: 796   1st Qu.: 884   1st Qu.:   0.0   1st Qu.:1148   1st Qu.: 5.000
##  Median :1016   Median :1096   Median :   0.0   Median :1463   Median : 6.000
##  Mean   :1071   Mean   :1162   Mean   : 344.5   Mean   :1516   Mean   : 6.486
##  3rd Qu.:1341   3rd Qu.:1392   3rd Qu.: 720.0   3rd Qu.:1767   3rd Qu.: 7.000
##  Max.   :3206   Max.   :3228   Max.   :1818.0   Max.   :3608   Max.   :12.000
##
##    GarageArea       WoodDeckSF      OpenPorchSF       SalePrice
##  Min.   :   0.0   Min.   :  0.00   Min.   :  0.00   Min.   : 34900
##  1st Qu.: 312.0   1st Qu.:  0.00   1st Qu.:  0.00   1st Qu.:128425
##  Median : 470.5   Median :  0.00   Median : 27.50   Median :165550
##  Mean   : 462.8   Mean   : 94.93   Mean   : 46.89   Mean   :182517
##  3rd Qu.: 576.0   3rd Qu.:168.00   3rd Qu.: 72.00   3rd Qu.:216625
##  Max.   :1166.0   Max.   :857.00   Max.   :523.00   Max.   :555000
##
```

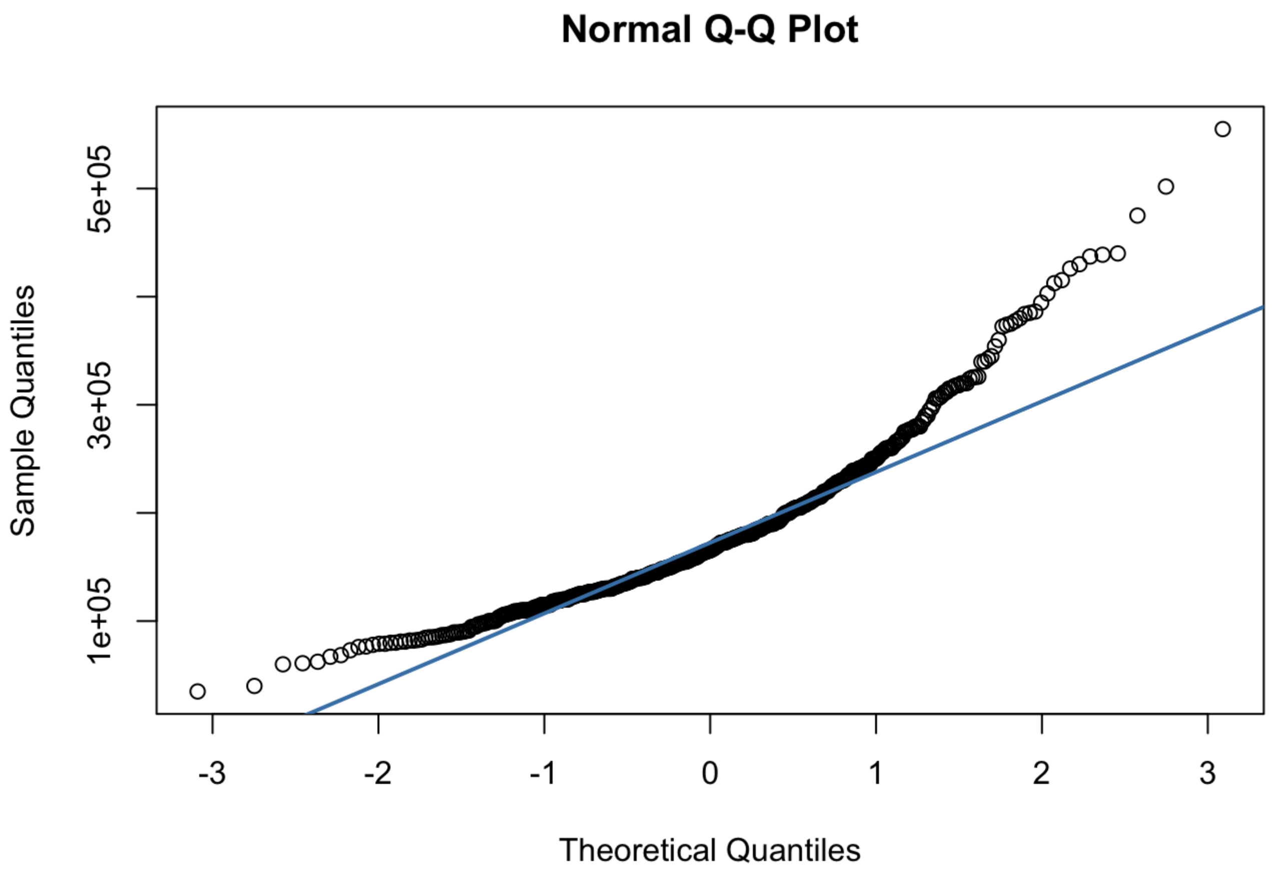
We summarise the 'House' data with their statistics. We suspect the distributions of some variables are left-skewed. Those EVs may cause to biasedness in our analysis. Moreover, there are missing values in some EVs which we will deal with them later.

```
library(corrplot)
library(RColorBrewer)
house_cor = house
house_cor$LotShape = as.numeric(house_cor$LotShape)
corrplot(cor(na.omit(house_cor)), type = "upper", order = "hclust", col = brewer.pal(n = 8,
 name = "RdYlBu"))
```



Furthermore, we obtain a graph to describe the correlation among variables. We inspect 'SalePrice' highly correlates with 'GrLivArea' which may be a significant EV to develop our models. We also notice some EVs associate with each other, and it may be a signal of multicollinearity. We will take care of it while constructing models.

```
qqnorm(house$SalePrice)
qqline(house$SalePrice, col = "steelblue", lwd = 2)
```



When we are trying to inspect the distribution of RV, problematically, the above graph does not suggest 'SalePrice' is following a normal distribution. It is a critical issue in the regression problem as the normality requirement for the ordinary least squares (OLS) method. Put another way, we should not use the regression model for the analysis. It will ruin our investigation and the project. As such, we assume the normality of RV here to continue the report although it is not appropriate for some models we use in the later section.

Missing Data and Data Cleansing

As we discovered there are missing values appeared in the available data, we need to impute them to facilitate the analysis. There are 87 missing data in 'LotFrontage' and 1 missing data in 'MasVnrArea'. To preserve the information provided by the data as much as possible, we attempt to use a regression model to replace the missing values.

```
shapiro.test(na.omit(house$LotFrontage))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  na.omit(house$LotFrontage)
## W = 0.96393, p-value = 1.526e-08
```

However, the Shapiro-Wilk normality test does not suggest us to use a regression model because it violates the normality assumption of the OLS method. We, hence, substitute the missing data by its mean. We believe it is the best alternative to maintain the information.

Moreover, there is a missing value in 'MasVnrArea'. We believe it is missing completely at random, and it is hard to replace the data as we know some people may not appreciate the area of masonry veneer. From the previous summary table, we can see that over 50% of houses do not consider the space. For the safety of analysis, we propose to fill it with its mode.

Last but not least, the house id does not favour our modelling. So, we drop the 'id' for simplicity.

```
house[which(is.na(house$LotFrontage)), "LotFrontage"] = mean(na.omit(house$LotFrontage))
house[which(is.na(house$MasVnrArea)), "MasVnrArea"] =  (na.omit(house$MasVnrArea))[which.max
(tabulate(match(na.omit(house$MasVnrArea), unique(na.omit(house$MasVnrArea)))))]
house = house[-1]
```

Preliminaries

We divide the 'House' data into two parts. One will be a train data which helps to develop models, and the other one will be a test data which serves to evaluate the accuracy of models. The division helps to facilitate the model evaluation by finding the prediction error. Of course, we will consider model selection during each development. Since there are some turning parameters which controls the complexity of our proposed model, we will use cross-validation to tune those parameters.

```
RNGkind(sample.kind = "Rounding")
set.seed(4011)
index = sample(1:nrow(house), nrow(house) * 0.1)
train = rep(T, nrow(house))
train[index] = F
test = !train
```

Linear Regression Model

Linear regression is one of the most common approaches to learn the data pattern. It uses OLS method to minimise the residual sum of squares (RSS), such that we have the famous formula $\hat{Y} = X\hat{\beta}$.

```
lin_reg = lm(SalePrice ~ ., data = house, subset = train)
summary(lin_reg)
```

```
##
## Call:
## lm(formula = SalePrice ~ ., data = house, subset = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -111775  -17386   -1089   17534  189170
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.791e+04  1.372e+04  -4.222 2.95e-05 ***
## LotFrontage   8.429e+01  9.386e+01   0.898 0.369662
## LotArea       2.237e-01  1.318e-01   1.697 0.090354 .
## LotShapeIR2   2.096e+04  9.775e+03   2.145 0.032534 *
## LotShapeIR3   6.950e+03  2.280e+04   0.305 0.760583
## LotShapeReg  -8.378e+03  3.776e+03  -2.219 0.027014 *
## OverallCond   5.211e+03  1.535e+03   3.394 0.000752 ***
## MasVnrArea    3.941e+01  9.547e+00   4.128 4.38e-05 ***
## TotalBsmtSF   5.172e+01  7.767e+00   6.659 8.37e-11 ***
## X1stFlrSF     5.477e+01  2.944e+01   1.860 0.063529 .
## X2ndFlrSF     5.582e+01  2.842e+01   1.964 0.050148 .
## GrLivArea     2.012e+01  2.790e+01   0.721 0.471340
## TotRmsAbvGrd -3.252e+03  2.032e+03  -1.600 0.110266
## GarageArea    1.078e+02  1.032e+01  10.445  < 2e-16 ***
## WoodDeckSF    1.646e+01  1.425e+01   1.155 0.248616
## OpenPorchSF   8.399e+01  2.813e+01   2.986 0.002989 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 35950 on 434 degrees of freedom
## Multiple R-squared:  0.8018, Adjusted R-squared:  0.7949
## F-statistic:   117 on 15 and 434 DF,  p-value: < 2.2e-16
```

For the full model, which contains all the EVs, the adjusted R^2 is 79.49. Although it seems that EVs are pretty good to explain the variation of RV, there are some inefficient parties. Model selection helps to eliminate those unnecessary EVs in the model. We will pick the parsimonious model under the criteria of AIC and BIC with stepwise selection.

```r
library(MASS)
library(dplyr)
AIC = lin_reg %>% stepAIC(trace = T)
BIC = lin_reg %>% stepAIC(trace = log(nrow(house)))
```

```r
AIC$call == BIC$call
```

```
## [1] TRUE
```

```r
model_terms = strsplit(as.character(AIC$call$formula), " ")
model_terms = model_terms[[3]][model_terms[[3]] != "1" & model_terms[[3]] != "+"]
model_terms
```

```
##  [1] "LotArea"      "LotShape"     "OverallCond"  "MasVnrArea"   "TotalBsmtSF"
##  [6] "X1stFlrSF"    "X2ndFlrSF"    "TotRmsAbvGrd" "GarageArea"   "OpenPorchSF"
```

Under the criteria of AIC and BIC, coincidentally, both of them suggest the same parsimonious model by removing 'LotFrontage', 'LotArea', 'GrLivArea' and 'WoodDeckSF' from the full model. The parsimonious model is developed based on the train data and the criteria of AIC and BIC. We, consequently, fit the 'relative best' model as the final model to estimate the house prices.

```r
library(boot)
lin_reg = glm(paste0("SalePrice ~ ", as.character(AIC$call$formula)[3]), data = house, subse
t = train)
summary(lin_reg)
```

```
##
## Call:
## glm(formula = paste0("SalePrice ~ ", as.character(AIC$call$formula)[3]),
##     data = house, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -110651   -17496    -1097    18179   186902
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.476e+04  1.299e+04  -4.216 3.02e-05 ***
## LotArea       2.626e-01  1.281e-01   2.050 0.040981 *
## LotShapeIR2   2.020e+04  9.687e+03   2.085 0.037609 *
## LotShapeIR3   3.208e+03  2.265e+04   0.142 0.887414
## LotShapeReg  -9.260e+03  3.721e+03  -2.489 0.013194 *
## OverallCond   5.265e+03  1.528e+03   3.446 0.000624 ***
## MasVnrArea    3.894e+01  9.511e+00   4.094 5.04e-05 ***
## TotalBsmtSF   5.212e+01  7.752e+00   6.724 5.55e-11 ***
## X1stFlrSF     7.641e+01  1.048e+01   7.288 1.48e-12 ***
## X2ndFlrSF     7.662e+01  7.100e+00  10.791  < 2e-16 ***
## TotRmsAbvGrd -3.128e+03  1.998e+03  -1.566 0.118167
## GarageArea    1.106e+02  1.005e+01  11.011  < 2e-16 ***
## OpenPorchSF   8.454e+01  2.811e+01   3.008 0.002785 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1291256990)
##
##     Null deviance: 2.8296e+12  on 449  degrees of freedom
## Residual deviance: 5.6428e+11  on 437  degrees of freedom
## AIC: 10732
##
## Number of Fisher Scoring iterations: 2
```

Although there are two inefficient coefficient estimates, we accept the model as it achieves the parsimonious model among all possible models. In other words, the fitted model is ready to serve for our analysis.

```
result$reg_pro[1, 1] = cv.glm(house[train, ], lin_reg, K = 10)$delta[1]
result$reg_pro[1, 2] = mean((predict(lin_reg, house[test, ]) - house$SalePrice[test]) ^ 2)
result$reg_pro[1, ]
```

```
##                    Cross.Validation.Error Prediction.Error
## Linear Regression              1442086829        960622735
```

To evaluate the model accuracy, we use 10-folds cross-validation to get the generalised MSE. The MSE for linear regression is 1442086829, and the prediction error is 960622735. One thing to remind here is that the RV is not normally distributed, which violates the OLS assumption. It may be an improper use of the linear regression model.

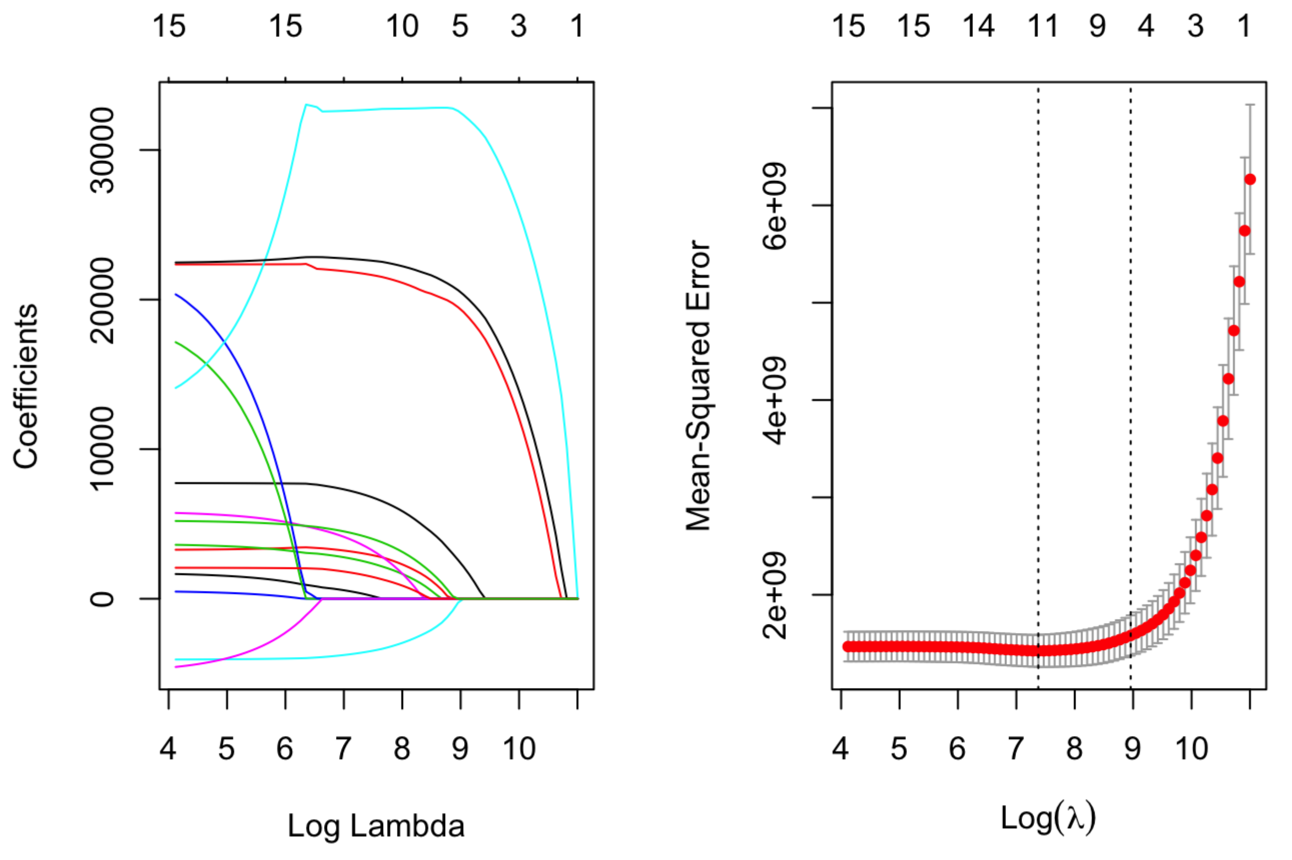
Lasso Regression Model

Instead of using AIC and BIC to select the model, lasso regression introduces a penalty term that forces some of the coefficient estimates to be exactly equal to zero. Ridge regression does similar things, but it will include all the EVs in the final model, which is not very pleasant, so we only focus on lasso regression in the project. The idea of lasso regression minimises the RSS and a shrinkage penalty, which is defined as $\arg\min(RSS + \lambda \sum |\beta_j|)$. The greater the λ, the more penalisation will be given to the coefficients. Since λ is a tuning parameter, we will use cross-validation to select it. In addition, the coefficient estimates are scale variant, and we need to standardising them to avoid them change substantially.

```
x = model.matrix(SalePrice ~ ., data = house)[, -1]
x = scale(x)
y = house$SalePrice

library(glmnet)
las_reg = glmnet(x[train, ], y[train], alpha = 1)
lambda = cv.glmnet(x[train,], y[train], alpha = 1, number = 10)
```

```
par(mfrow = c(1, 2))
plot(las_reg, xvar = "lambda", label = TRUE)
plot(lambda)
```



On the left-hand side, each line represents a coefficient estimate. The graph describes what we have just mentioned a second before. When the value of λ increases, the coefficient estimates toward zero and even equal to zero, which is equivalent to model selection for some coefficients exact zero. Since we want to select a model that minimises the MSE, we need to find a value of λ that satisfies our goal. We utilise 10-folds cross-validation to determines the size of penalisation. On the right-hand side, the diagram illustrates the MSE at different levels of λ. We notice that the model will have the smallest cross-validation error when λ is about $\log(7.4)$. We, then, fit the lasso regression with such value to estimate the house costs.

```
las_reg = glmnet(x[train, ], y[train], alpha = 1, lambda = lambda$lambda.min)
coefficients(las_reg)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept)  182594.1285
## LotFrontage     288.2479
## LotArea        2999.2683
## LotShapeIR2    2469.0266
## LotShapeIR3        .
## LotShapeReg   -3559.8687
## OverallCond    3481.8178
## MasVnrArea     6923.5162
## TotalBsmtSF   21710.8150
## X1stFlrSF          .
## X2ndFlrSF          .
## GrLivArea     32684.1379
## TotRmsAbvGrd       .
## GarageArea    22638.1071
## WoodDeckSF     1559.2033
## OpenPorchSF    4140.6753
```

The final model drops 'LotShapeIR3', 'X1stFlrSF', 'X2ndFlrSF' and 'TotRmsAbvGrd' out from the full model. The model suggests a different EV elimination than that of of the linear regression we modelled previously. Again, we record the cross-validation error and prediction error of the model for the model evaluation.

```
result$reg_pro[2, 1] = min(lambda$cvm)
result$reg_pro[2, 2] = mean((predict(las_reg, s = lambda$lambda.min, newx = x[test, ]) - y[t
est]) ^ 2)
result$reg_pro[2, ]
```

```
##                    Cross.Validation.Error Prediction.Error
## Lasso Regression               1423777022        946892882
```

With 10-folds cross-validation, the MSE is 1423777022, and the prediction error is 946892882. Both MSE in cross-validation and prediction is slightly smaller than the linear regression. Because they are using different model selection criteria, the performance will also be divergent.

Partial Least Squares Regression Model

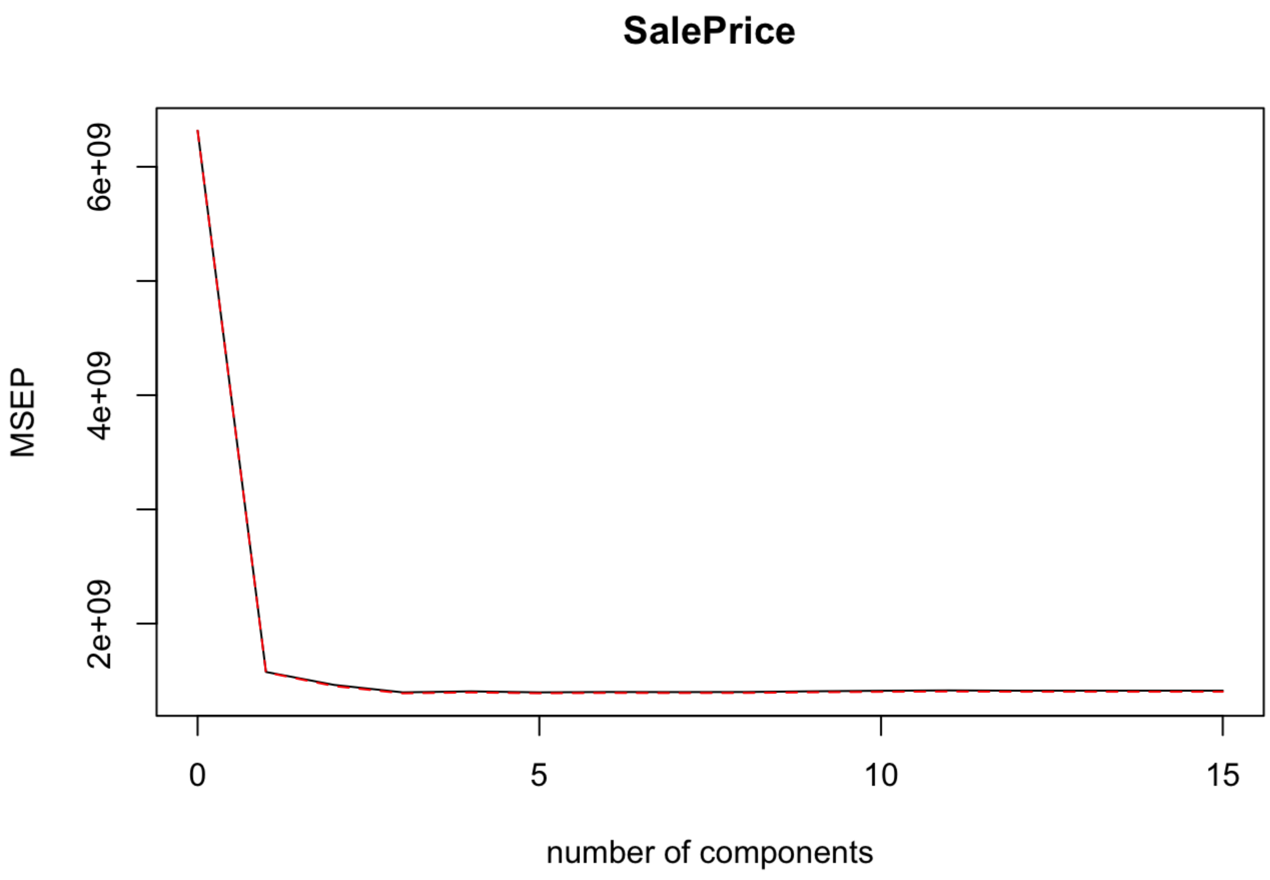
Principal component regression (PCR) takes the advantages of principal component analysis (PCA) to transform EVs and reduce their dimensions. Yet, PCR focus on preserving the variation within EVs. In the project, we are interested in the association between RV and EVs instead of EVs themselves. Partial least squares regression (PLS) behaves similar but concentrate on the relationship between RV and EVs, which favour our project more. Thus, we adopt PLS regression as an additional learning model for the 'House' data.

```
library(pls)
pls_reg = plsr(SalePrice ~ ., data = house, subset = train, scale = T, validation = "CV", nu
mber = 10)
summary(pls_reg)
```

```
## Data:    X dimension: 450 15
##  Y dimension: 450 1
## Fit method: kernelpls
## Number of components considered: 15
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           79473    39709    38250    37393    37509    37391    37437
## adjCV        79473    39667    38109    37275    37379    37282    37323
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       37424    37434    37517     37580     37622     37589     37590
## adjCV    37311    37321    37398     37454     37493     37463     37465
##        14 comps  15 comps
## CV        37590     37590
## adjCV     37464     37464
##
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X            26.79    32.43    38.35    45.11    52.91    60.60    66.79
## SalePrice    75.95    79.19    79.80    80.04    80.13    80.15    80.15
##            8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X            71.94    76.56     78.45     82.66     84.36     89.29     95.20
## SalePrice    80.15    80.16     80.17     80.17     80.18     80.18     80.18
##            15 comps
## X            100.00
## SalePrice     80.18
```

From the above summary table, we notice that the Root MSE (RMSE) is 37391 with 5 components under 10-folds cross-validation. Besides, 5 components already explain 80.13 of correlation between RV and EVs. Simply put, the PLS regression with 5 components perform well in explaining the variation of RV.

```r
validationplot(pls_reg, val.type = "MSEP")
```



The plot describes the values of MSE with different numbers of components. The difference in errors is not very significant, and it is hard to eyeball the smallest one. Thankfully, we already know 5 components are the best one to explain the variation. It is complicated to interpret the result of PLS regression, so we may only compute the errors in cross-validation and prediction.

```r
result$reg_pro[3, 1] = 37391 ^ 2
result$reg_pro[3, 2] = mean((predict(pls_reg, x[test, ], ncomp = 5) - y[test]) ^ 2)
result$reg_pro[3, ]
```

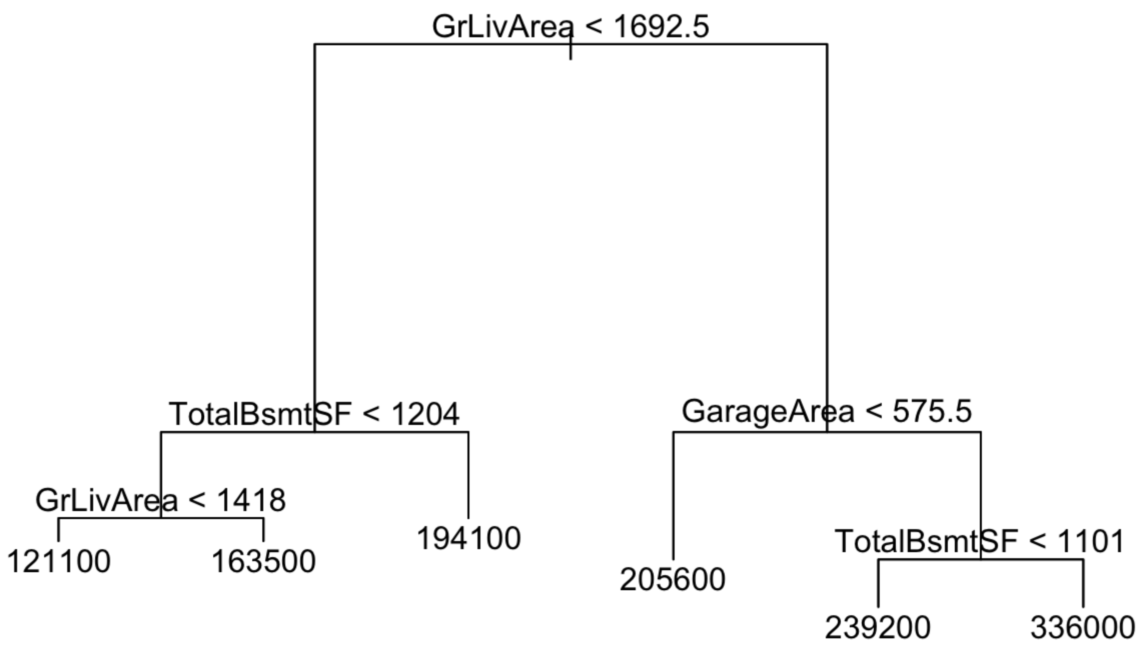
```
##                                    Cross.Validation.Error Prediction.Error
## Partial Least Squares Regression               1398086881      65366530722
```

The MSE of PLS regression is 1398086881, and its prediction error is 65366530722. Both errors are larger than the other regression methods used beforehand. It seems 'House' data is not suitable to reduce EVs' dimension for estimation. Recall the PLS model summary table, there is only 52.91 of variation maintained in 5 components. For PLS regression, it emphasises the association between RV and EVs. However, the house price varies when there comes to a new feature in the house. We foresee the PCR regression behaves similarly as preserving the correlation among EVs may not function well on the relation with RV.

Radnom Forests Model

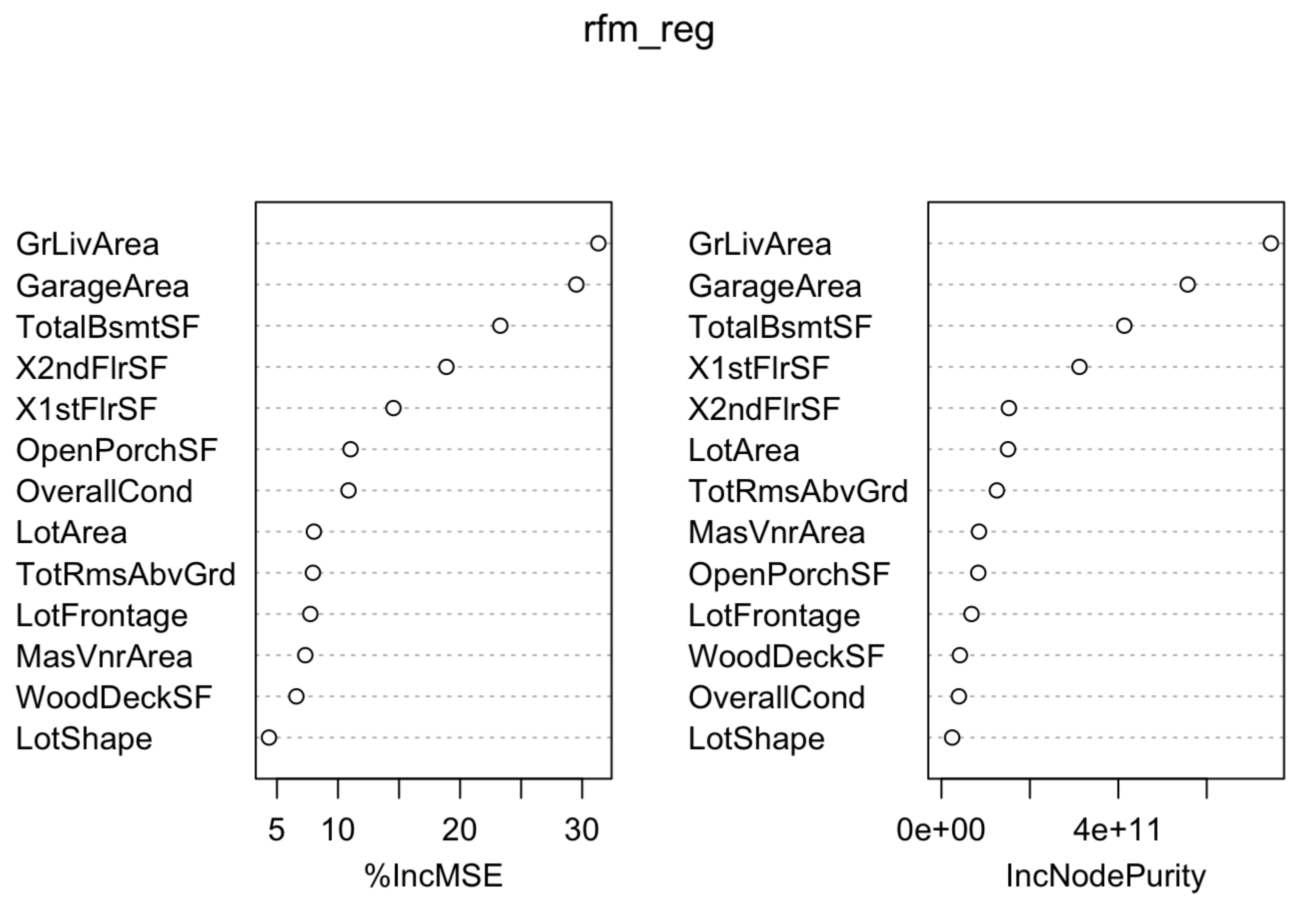
So far we have gone through a few traditional regression methods. We want to model the 'House' data in a different aspect. In artificial intelligence, the decision tree is one of the simplest and yet most successful forms of learning algorithms in inductive learning. We propose to use a regression tree to stratify the EVs' space into smaller regions and predict the final house prices. However, one decision tree may consist of biasedness to the prediction. We, consequently, consider the use of random forest, an improved version of bagging. Bagging constructs multiple decision trees and predicts the final result by taking the average of predicted RV. As some trees in bagging may be correlated on each other, we want to decorrelate them by considering a random selection of $p/3$ EVs in each split, that is the idea of random forest.

```
library(tree)
tre_reg = tree(SalePrice ~ ., data = house, subset = train)
tre_reg = prune.tree(tre_reg, best = 6)
plot(tre_reg)
text(tre_reg, pretty = 0)
```



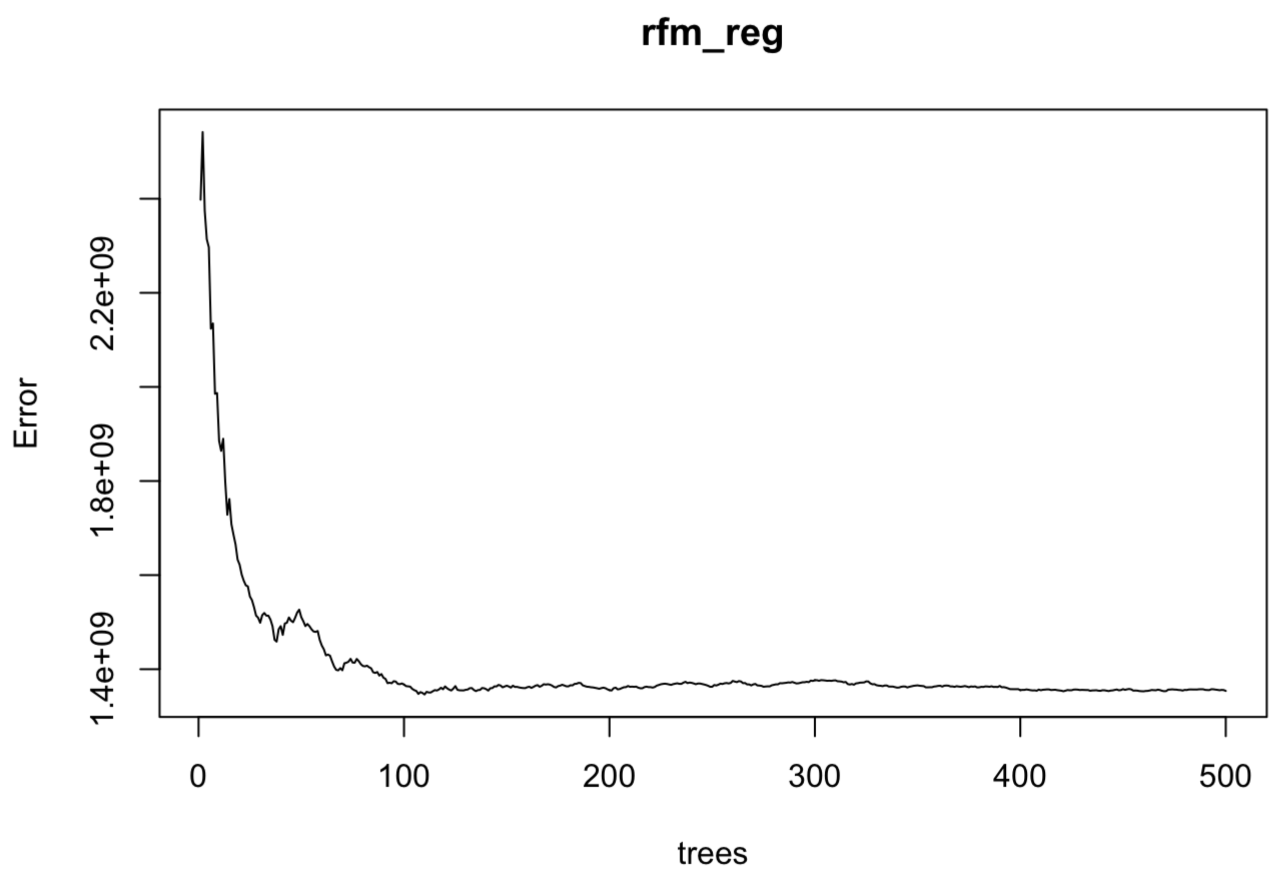
The above tree diagram demonstrates a tree-liked algorithm may appear in the random forest. For each tree, the space of EVs is segmented based on the smallest RSS for each EV. To avoid overfitting, we may prune the tree according to the advice from cross-validation. The diagram illustrates how it would be when we prune the tree and retain 6 leaf nodes. In the random forest, the random selection of EVs replaces the pruning process for a simpler tree in the forest.

```
library(randomForest)
rfm_reg = randomForest(SalePrice ~ ., data = house, subset = train, mtry = round((ncol(hous
e) - 1) / 3), importance = T)
varImpPlot(rfm_reg)
```



The above graphs show the importance of each EV while modelling the random forest. Although two pictures use different measurements, the top three important EVs are 'GrLivArea', 'GarageArea' and 'TotalBsmtSF'. Recall the EDA that we have discussed at the beginning of the section, the correlation of 'GrLivArea' and 'SalePrice' shows a signal that it is essential in model development. We inspect the higher correlation with RV, the more important of EV.

```
plot(rfm_reg)
```



We have developed 500 trees to form a forest as the final model. As we can see in the above diagram, the error decreases as the number of trees increase. It shows the beauty of the random forest, which reduces the MSE by growing a lot of trees.

```
result$reg_pro[4, 1] = tail(rfm_reg$mse, 1)
result$reg_pro[4, 2] = mean((predict(rfm_reg, newdata = house[test, ]) - y[test]) ^ 2)
result$reg_pro[4, ]
```

```
##                 Cross.Validation.Error Prediction.Error
## Random Forest             1353853738        648139582
```

The construction of each tree is using a different bootstrap sample from the original data. In other words, the random forest does not require cross-validation to select the best one (B.Leo and C.Adele). We report the MSE as the cross-validation error here. The MSE of the random forest is 1353853738, and the prediction error is 648139582. Both errors are less than the proposed model before especially the prediction error is significantly less than others.

Summary of Regression Problem

```
sqrt(result$reg_pro)
```

```
##                                   Cross.Validation.Error Prediction.Error
## Linear Regression                               37974.82         30993.91
## Lasso Regression                                37732.97         30771.62
## Partial Least Squares Regression                37391.00        255668.79
## Random Forest                                   36794.75         25458.59
```

For reading-friendly, we convert the MSE to RMSE to express the errors in each model. It is crystal clear that the random forest model is the best among the four proposed models, which has the lowest cross-validation error and prediction error. The other models seem to have similar performance in fitting the 'House' data. However, the prediction of PLS regression is significantly larger than the others. PLS regression acts the worst on adapting new environment change in the given data. For 'House' data, the recommended model is the random forest, whereas the least suggestion is PLS regression. Since the random forest is dividing the space of EVs, there are some other methods adopt a similar concept, such as regression spline. It motivates a further investigation on the same set of data in the future.

Classification Problem

Sometimes, we may encounter a problem where RV is qualitative. We consider the circumstance as a classification problem. We want to estimate the likelihood of RV belongs to a particular class. In the classification problem, we will implement the analysis with 'Titanic' data.

Data Background

```
titanic <- read.csv("Titanic.csv", header = TRUE)
dim(titanic)
```

```
## [1] 500   9
```

Once upon a time, there was an accidence that Titanic sank after colliding with an iceberg. Under the terrible situation, there was not enough resource to save all lives. The 'Titanic' data consists of 500 passengers information in the accidence. We want to develop models to investigate whether a passenger will survive in the crash. As usual, we will start with EDA before our investigation.

Exploratory Data Analysis

```
summary(titanic[, -1])
```

```
##     Survived         Pclass          Sex            Age             SibSp      
##  Min.   :0.000   Min.   :1.000   female:185   Min.   : 0.75   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:2.000   male  :315   1st Qu.:20.12   1st Qu.:0.000  
##  Median :0.000   Median :3.000                Median :28.00   Median :0.000  
##  Mean   :0.386   Mean   :2.326                Mean   :29.20   Mean   :0.574  
##  3rd Qu.:1.000   3rd Qu.:3.000                3rd Qu.:37.75   3rd Qu.:1.000  
##  Max.   :1.000   Max.   :3.000                Max.   :71.00   Max.   :8.000  
##                                               NA's   :102                    
##      Parch            Fare          Embarked
##  Min.   :0.00   Min.   :  0.000    :  1    
##  1st Qu.:0.00   1st Qu.:  7.925   C: 92    
##  Median :0.00   Median : 14.456   Q: 45    
##  Mean   :0.38   Mean   : 31.782   S:362    
##  3rd Qu.:0.00   3rd Qu.: 30.071            
##  Max.   :5.00   Max.   :512.329            
## 
```

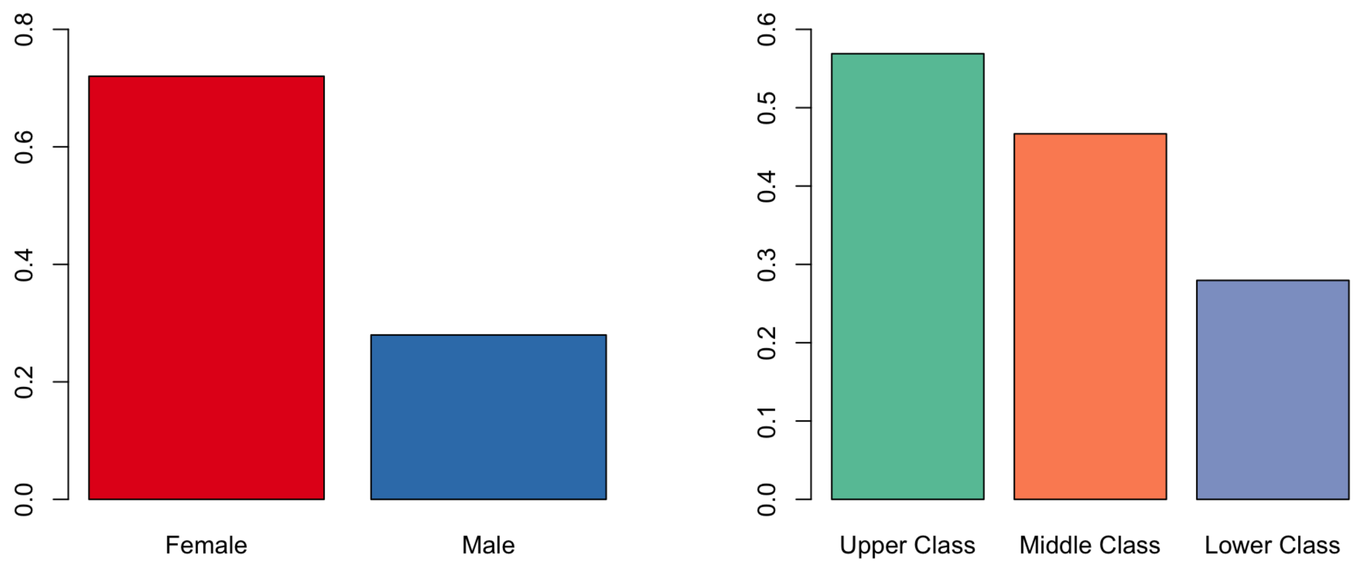
As we know that only 'age', 'sibsp', 'parch' and 'fare' are quantitative data, it is difficult to interpret the findings from the summary statistics. Likewise, there are missing data appeared in the dataset, which happened in 'Age' and 'Embarked'. Unfortunately, there are so many missing data occurred in 'Age', and we will handle the critical issue in the next section.

```
female = sum(titanic[which(titanic$Sex == "female"), "Survived"])/ sum(titanic$Survived == 1
)
male = sum(titanic[which(titanic$Sex == "male"), "Survived"])/ sum(titanic$Survived == 1)

upperClass = mean(titanic[which(titanic$Pclass == 1), "Survived"])
```

```
middleClass = mean(titanic[which(titanic$Pclass == 2), "Survived"])
lowerClass = mean(titanic[which(titanic$Pclass == 3), "Survived"])

library(RColorBrewer)
par(mfrow = c(1, 2))
barplot(c(female, male), names = c("Female", "Male"), ylim = c(0, 0.8), col = brewer.pal(3,
"Set1"))
barplot(c(upperClass, middleClass, lowerClass), names = c("Upper Class", "Middle Class", "Lo
wer Class"), ylim = c(0, 0.6), col = brewer.pal(3, "Set2"))
```

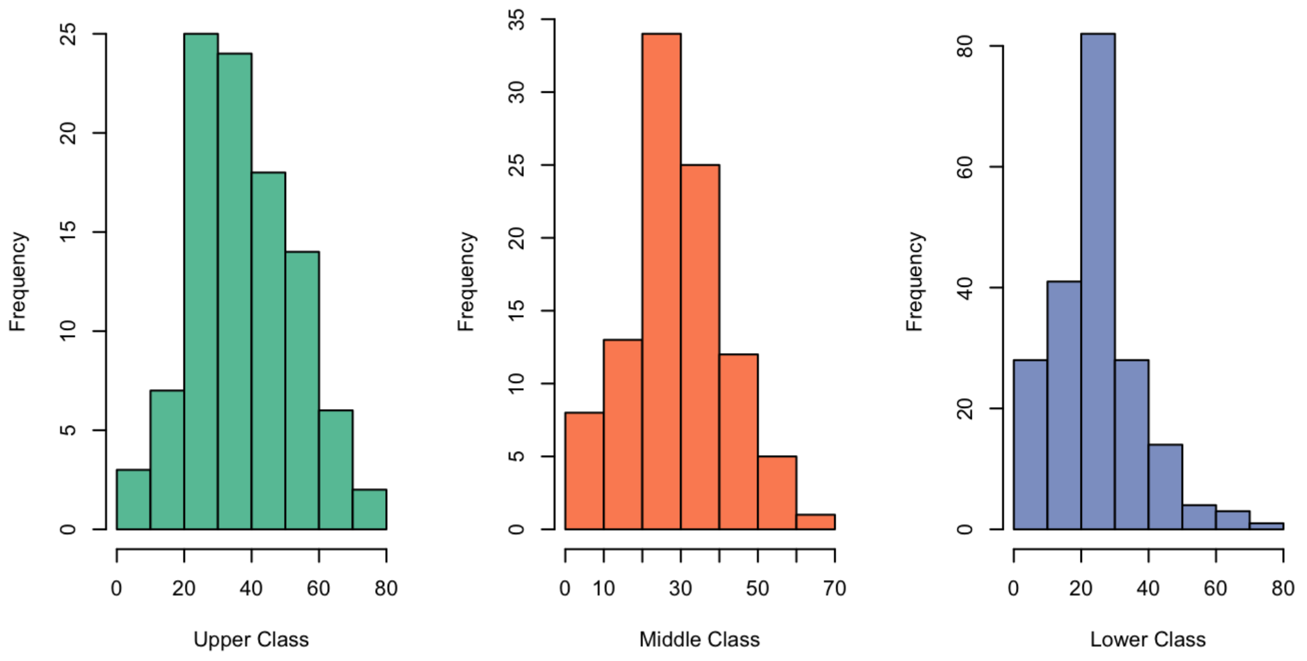


If we just eyeball the summary findings, it is tough to find out any useful information from the above table. So, we plot the proportion of survived passengers by gender and their tickle class. On the left-hand side, we discover females seem to have a higher survival rate than males. In the old days, people emphasis the gentleness towards women. Those gentlemen possibly sacrifice themselves and let women get into the lifeboats first. That is one of the possible reason females have a higher survival rate. Come back, Jack! Furthermore, on the right-hand side, the graph shows passengers who hold upper-class tickle has a higher chance to survive. We suspect lifeboats are located near the position of the upper-class seat. After the collision, the upper-class passengers are more easily to get into the lifeboats.

Missing Data and Data Cleansing

There are 102 missing values in 'Age' which occupied 20% of the data. We assume they are missing at random, and it is out of scope to make further investigation. We cannot impute those missing data casually since we consider 'Age' as an important feature in the 'Titanic' data. It implicitly describes the survival ability of that person, such as physical power and mobility. We assume most of the passengers were not born with a silver spoon in their mouths. Junior tends to prefer lower class due to their financial ability. In other words, we hypothesise passengers with higher ticket class are older people.

```
par(mfrow = c(1, 3))
hist(titanic[which(titanic$Pclass == 1), "Age"], xlab = "Upper Class", main = NULL, col = br
ewer.pal(3, "Set2")[1])
hist(titanic[which(titanic$Pclass == 2), "Age"], xlab = "Middle Class", main = NULL, col = b
rewer.pal(3, "Set2")[2])
hist(titanic[which(titanic$Pclass == 3), "Age"], xlab = "Lower Class", main = NULL, col = br
ewer.pal(3, "Set2")[3])
```



As expected, passengers who hold lower-class tickle are mostly younger, and those in the higher class tend to be older. From the above phenomenon, we propose to replace the missing 'Age' values by 90% trimmed mean of their corresponding ticket class.

In addition, there is only a missing data appeared in 'Embarked', we believe it is missing completely at random. Since we lack the background about the port of embarkation, we may only rely on the available data to replace the missing one. From the previous summary table, we know the majority of passengers onboard from Southampton. We guess that passenger with missing value in 'Embarked' also on board from Southampton. Thus, we may only replace the missing data with its mode again.

Besides, we label the class of 'Survived' to facilitate our works. For those with '1' in value, we rename it as 'Survived'. Otherwise, we rename it as 'Died'.

As some of the learning algorithms we use in the later section do not support categorical variables, we treat those variables as quantitative instead, when we are applying those algorithms. We will convert the categorical variables into factor data type here as well.

Last but not least, the passenger id does not favour our modelling. We drop the 'PassengerId' for simplicity.

```
titanic[which(is.na(titanic$Age)), "Age"] = ifelse(titanic$Pclass == 1, round(mean(na.omit(t
itanic[which(titanic$Pclass == 1), "Age"]), trim = 0.1)), ifelse(titanic$Pclass == 2, round
(mean(na.omit(titanic[which(titanic$Pclass == 2), "Age"]), trim = 0.1)), round(mean(na.omit
(titanic[which(titanic$Pclass == 3), "Age"]), trim = 0.1))))[which(is.na(titanic$Age))]
titanic[which(titanic$Embarked == ""), "Embarked"] = "S"
titanic$Embarked = as.factor(as.character(titanic$Embarked))
titanic$Sex = as.factor(titanic$Sex)
titanic$Survived = factor(ifelse(titanic$Survived == 1, "Survived", "Died"))
titanic = titanic[-1]
```

Preliminaries

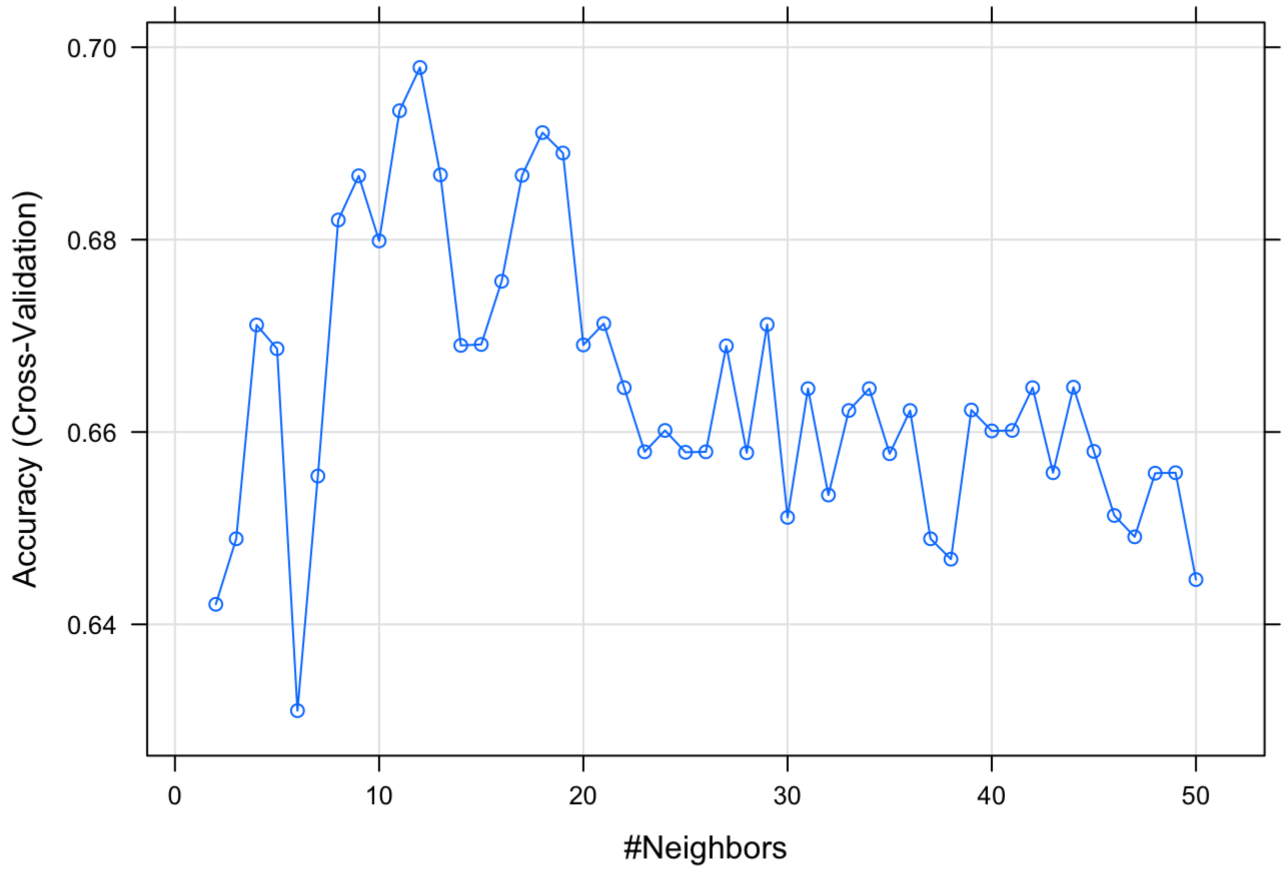
To inhibit the chance of overfitting, again, we split the 'Titantic' data into two parts. The first part will be a train data for training models, and the remaining portion will be treated as a test data which uses to examine the precision of models. Furthermore, we will utilise cross-validation to choose the tuning parameters that appear in the proposed models. Model selection will be considered, as well.

```
set.seed(4011)
index = sample(1:nrow(titanic), nrow(titanic) * 0.1)
train = rep(T, nrow(titanic))
train[index] = F
test = !train
```

k-Nearest Neighbors Model

In the real world, we often interested in finding similar items as a prediction. We group objects into different clusters by the similarity of characteristics to predict the potential outcomes. For the k-nearest neighbour (KNN) method, we compute the Euclidean distance of passengers and infer them with similar properties. The classifier is formulated as $C_K(x) = \frac{\sum N_K(x,x_i)y_i}{K}$. As the name suggested, the KNN method finds adjacent passengers to classify the labels. The prediction varies as the value of K is arbitrary. Thus, we use 10-folds cross-validation to select the optimal value of K.

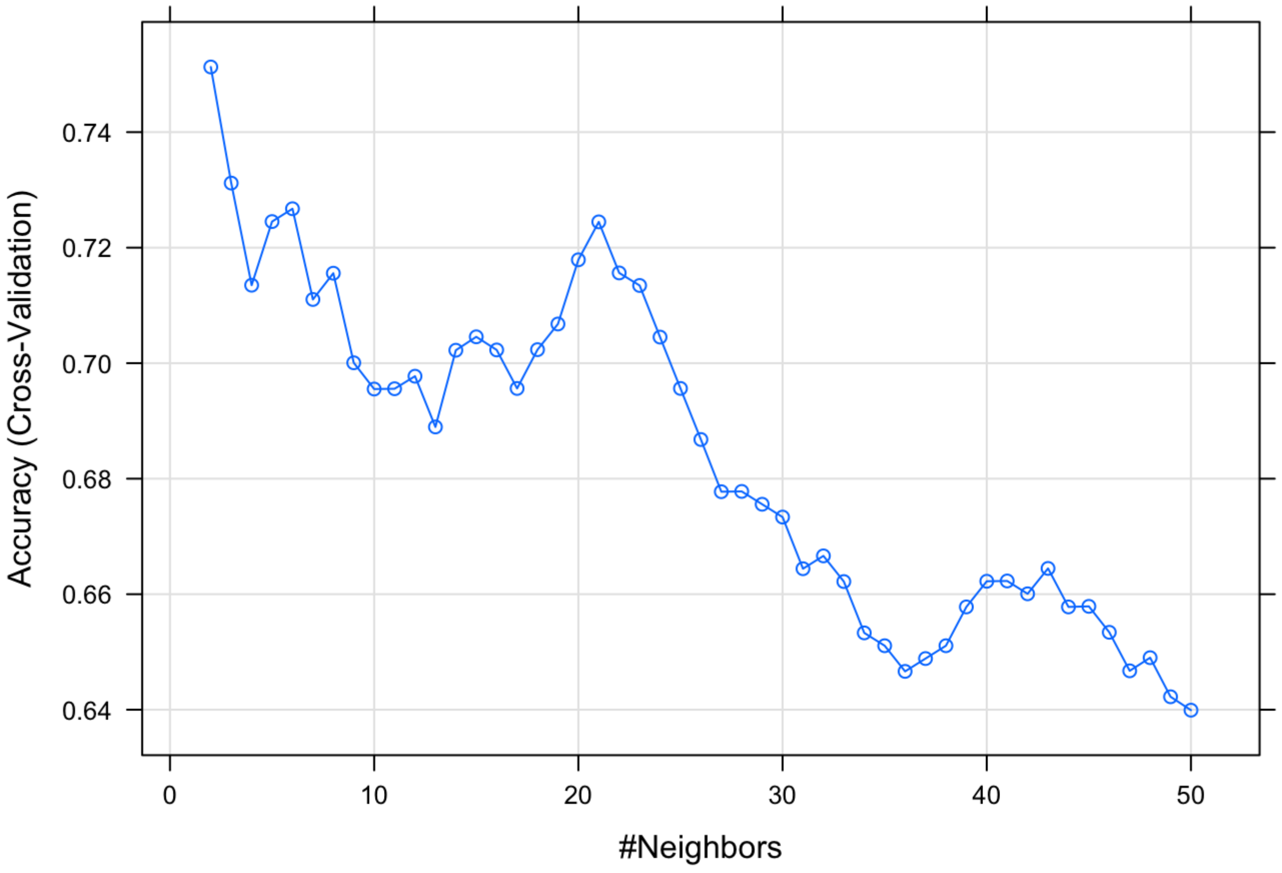
```
library(caret)
library(e1071)
ctrl = trainControl(method = "cv", number = 10)
knn_mod = train(Survived ~ ., data = titanic, subset = train, method = "knn", trControl = ct
rl, tuneGrid   = expand.grid(k = 2:50))
plot(knn_mod)
```



The above graph illustrates the number of neighbours against accuracy. Everyone can notice that the algorithm performs the best when considering 12 neighbours nearby, and 6 is the worse case. One interesting thing is when the number of neighbours increase, the power of the method becomes ill. If we think about the drawbacks of KNN, it is sensitive to irrelevant EVs and the scale of data (C.Marina). We, hence, attempt to perform model selection based on the result we have discussed in the EDA previously. We pick 'Pclass', 'Sex' and 'Age' as the EVs in the final model.

Nevertheless, 'Sex' is a categorical variable, and we cannot normalise the variables. Thus, we simply treat it as quantitative data instead.

```
knn_mod = train(Survived ~ Pclass + Age + Sex, data = titanic, subset = train, method = "kn
n", trControl = ctrl, tuneGrid   = expand.grid(k = 2:50))
plot(knn_mod)
```



After choosing the final model, it is noticeable that the accuracy indeed improved for the first 30 neighbours. In other words, if we consider 30 neighbours nearby, we can still obtain about 70% accuracy in predicting the status of passengers from the train data. Lastly, we designate 2 neighbours of passengers as the turning parameter since it attains the highest accuracy in 10-folds cross-validation.

```
knn_pre = predict(knn_mod, newdata = titanic[test, ])

library(MLmetrics)
result$cla_pro[1, 1] = 1 - knn_mod$results[knn_mod$bestTune[1, 1], "Accuracy"]
result$cla_pro[1, 2] = 1 - mean(knn_pre == titanic$Survived[test])
result$cla_pro[1, 3] = 1 - F1_Score(knn_pre, titanic$Survived[test], NULL)
result$cla_pro[1, ]
```

```
##                       Cross.Validation.Error Prediction.Error Balanced.Error
## k-Nearest Neighbors                0.2688208             0.26      0.1780822
```

The reported MSE for the KNN is 0.2688208, and the prediction error is 0.26. The performance seems quite honourable in classifying the status of passengers with about 70% accuracy, although we do not normalise the EVs in the algorithm. Other possible modelling approaches may do better if we can overcome the problem of the scale variant.

Logistic Regression Model

In the regression problem, we heavily rely on using the regression model. If we relax the OLS method assumption a little bit, we can use a generalised linear model (GML) to estimate the probability of survival. Given RV is binary data, either survived or died, we have a GML with the logit link function. Its name is logistic regression, which is formulated as $logit(\pi) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$.

```
log_reg = glm(Survived ~ ., data = titanic, subset = train, family = binomial)
summary(log_reg)
```

```
##
## Call:
## glm(formula = Survived ~ ., family = binomial, data = titanic,
##     subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6281  -0.6058  -0.4016   0.6021   2.4906
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.641970   0.894741   6.306 2.87e-10 ***
## Pclass      -1.133810   0.230100  -4.927 8.33e-07 ***
## Sexmale     -2.881748   0.276937 -10.406  < 2e-16 ***
## Age         -0.047707   0.011788  -4.047 5.19e-05 ***
## SibSp       -0.368878   0.155983  -2.365    0.018 *
## Parch       -0.033683   0.215581  -0.156    0.876
## Fare        -0.002005   0.004204  -0.477    0.633
## EmbarkedQ    0.535952   0.537185   0.998    0.318
## EmbarkedS   -0.251430   0.344383  -0.730    0.465
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 604.89  on 449  degrees of freedom
## Residual deviance: 392.43  on 441  degrees of freedom
## AIC: 410.43
##
## Number of Fisher Scoring iterations: 5
```

For the full model, the deviance is 392.43, which is far away from the saturated model. Moreover, we construct a likelihood ratio test to verify the effectiveness of the coefficient estimates. The test statistic G^2 is 212.46 with 8 degree of freedom. The reported p-value approximates 0. Although the full model is adequate, there are only a few coefficient estimates are effectual. When there are some inefficient parties in an adequate model, the issue of multicollinearity may happen in the given EVs. We, hence, elect the parsimonious model under the criteria of AIC and BIC with stepwise selection.

```
AIC = log_reg %>% stepAIC(trace = T)
BIC = log_reg %>% stepAIC(trace = log(nrow(titanic)))
```

```
AIC$call == BIC$call
```

```
## [1] TRUE
```

```
model_terms = strsplit(as.character(AIC$call$formula), " ")
model_terms = model_terms[[3]][model_terms[[3]] != "1" & model_terms[[3]] != "+"]
```

```
model_terms
```

```
## [1] "Pclass" "Sex"    "Age"    "SibSp"
```

From the suggestion given by AIC and BIC, accidentally, both criteria recommend the same parsimonious model. The model with "Pclass", "Sex", "Age" and "SibSp" is considered as parsimonious under the criteria of AIC and BIC from the train data. Therefore, we will fit the 'relative best' model to guess whether a passenger will be safe or not.

```
log_reg = glm(paste0("Survived ~ ", as.character(AIC$call$formula)[3]), data = titanic, subs
et = train, family = binomial)
summary(log_reg)
```

```
##
## Call:
## glm(formula = paste0("Survived ~ ", as.character(AIC$call$formula)[3]),
##     family = binomial, data = titanic, subset = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7316  -0.5881  -0.4175   0.6416   2.4310
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.26333    0.71792   7.331 2.28e-13 ***
## Pclass      -1.03102    0.17712  -5.821 5.84e-09 ***
## Sexmale     -2.91497    0.27125 -10.747  < 2e-16 ***
## Age         -0.04792    0.01167  -4.106 4.03e-05 ***
## SibSp       -0.42979    0.14446  -2.975  0.00293 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 604.89  on 449  degrees of freedom
## Residual deviance: 396.05  on 445  degrees of freedom
## AIC: 406.05
##
## Number of Fisher Scoring iterations: 5
```

From the summary table, we can see that all the coefficient estimates are efficacious. The deviance to the saturated model is 396.05, which is very close to the previous one. Yet, the model is the merely choice since it is the greedy model among all possible fits.

```
log_pro = predict(log_reg, newdata = titanic[test, ], type = "response")
log_pre = rep("Died", length(log_pro))
log_pre[log_pro > 0.5] = "Survived"

result$cla_pro[2, 1] = cv.glm(titanic[train, ], log_reg, K = 10)$delta[1]
result$cla_pro[2, 2] = 1 - mean(log_pre == titanic$Survived[test])
result$cla_pro[2, 3] = 1 - F1_Score(log_pre, titanic$Survived[test], NULL)
result$cla_pro[2, ]
```

```
##                       Cross.Validation.Error Prediction.Error Balanced.Error
## Logistic Regression                0.1421774             0.28            0.2
```

We evaluate the model with 10-folds cross-validation to obtain the generalised MSE. The MSE for our fitted logistic regression is 0.1414204, and the prediction error is 0.28. The performance of logistic regression very close to KNN, however, logistic regression is a little bit puny than KNN.

Linear Discriminant Analysis

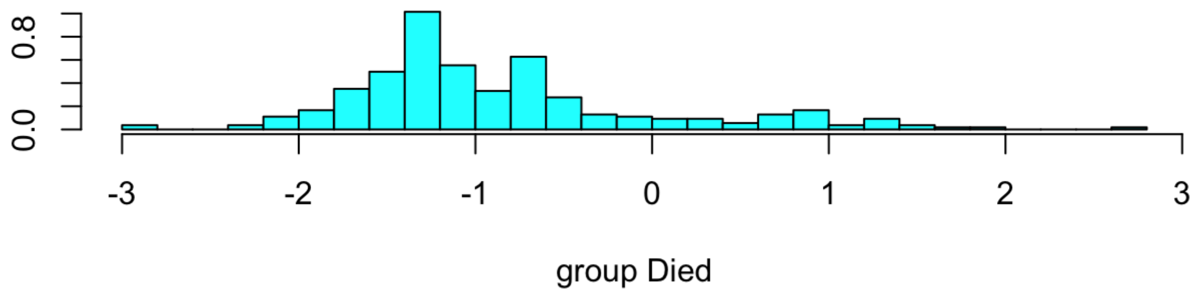
Under a similar setting with the logistic regression algorithm, our minds come to the linear discriminant analysis (LDA). LDA is analogous to PCA that we have mentioned before. Both of them are trying to reduce the dimensions of EVs to facilitate the analysis. Yet, LDA focuses on maximising the separability among known RV instead of preserving the most variation in EVs. To classify a passenger, we need to find the discriminant function which is defined as $\delta_k(x) = x \times \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$. The idea of LDA is creating new axes for EVs and project data points onto the new axes in a way to minimise the separation of RV. The target is to maximise the distance between means and minimise the scatter (or variation) within the possible values in RV.

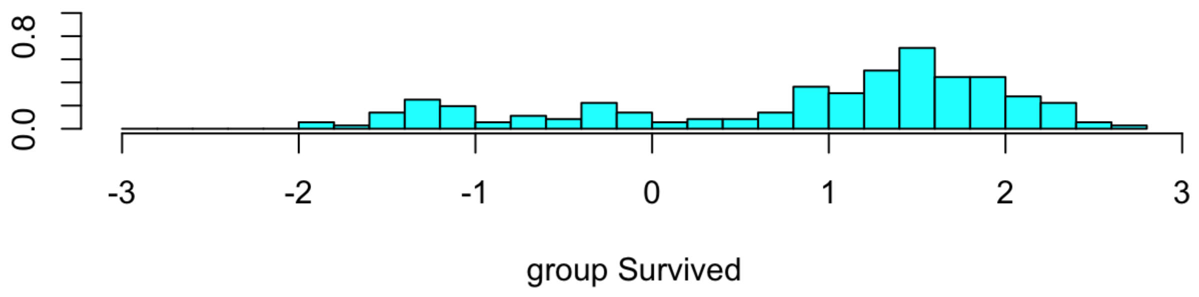
```
lda_mod = lda(Survived ~ ., data = titanic, subset = train)
lda_mod
```

```
## Call:
## lda(Survived ~ ., data = titanic, subset = train)
##
## Prior probabilities of groups:
##      Died  Survived
## 0.6022222 0.3977778
##
## Group means:
##             Pclass   Sexmale      Age     SibSp     Parch     Fare  EmbarkedQ
## Died     2.505535 0.8523985 29.88192 0.6051661 0.3394834 23.30841 0.07380074
## Survived 2.072626 0.2737430 27.20670 0.5027933 0.4245810 41.97593 0.12849162
##           EmbarkedS
## Died     0.7859779
## Survived 0.6201117
##
## Coefficients of linear discriminants:
##                    LD1
## Pclass    -0.6899883194
## Sexmale   -2.2477194312
## Age       -0.0279230927
## SibSp     -0.1996041093
## Parch     -0.0019909586
## Fare      -0.0009634991
## EmbarkedQ  0.3783874156
## EmbarkedS -0.1528437962
```

From the above summary table, LDA maximised the group means and minimised their scatters for the RV, either 'Survived' or 'Died'. Moreover, LDA also suggests the probability of labelling 'Died' for a passenger is 0.6022222, which is indeed a miserable fact.

```
plot(lda_mod)
```





Visually, we can see that if a passenger who lies on the x-axis (created by LDA after dimension reduction) with a higher value, that person will have a higher survival rate than those with a lower value. Furthermore, the probabilities for both categories in RV seem to follow some distributions, although the shape is not very clear. It is also a property of LDA which defines the decision boundary.

```r
lda_cv = function(titanic, train, K) {
  data = titanic[train, ]
  rownames(data) = NULL
  n = nrow(data)
  fold = K
  foldIndex = sample(fold, n, replace = T)
  error = rep(0, fold)
  for(i in 1:fold) {
    train = foldIndex != i
    test = foldIndex == i
    lda_cvm = lda(Survived ~ ., data = data, subset = train)
    lda_cvp = predict(lda_cvm, newdata = data[test, ])
    error[i] = 1 - mean(lda_cvp$class == data$Survived[test])
  }
  return(mean(error))
}

lda_pre = predict(lda_mod, newdata = titanic[test, ])

result$cla_pro[3, 1] = lda_cv(titanic, train, 10)
result$cla_pro[3, 2] = 1 - mean(lda_pre$class == titanic$Survived[test])
result$cla_pro[3, 3] = 1 - F1_Score(lda_pre$class, titanic$Survived[test], NULL)
result$cla_pro[3, ]
```

```
##                        Cross.Validation.Error Prediction.Error Balanced.Error
## Linear Discriminant                 0.1942697             0.22      0.1549296
```

For the moment, LDA performs the best among all the proposed models. The 10-folds cross-validation error is 0.1942697, and the prediction error is 0.22. There is almost 80% accuracy for classifying a passenger in the accidence. Dimension reduction seems to be robust in such a situation. Unlike PLS regression we used in the earlier section, LDA can eliminate the dimension but also preserve the accuracy in labelling a passenger. When we talk about something high-level idea in algebra, our mind comes up with the most powerful method we used in the regression problem.

Boosting

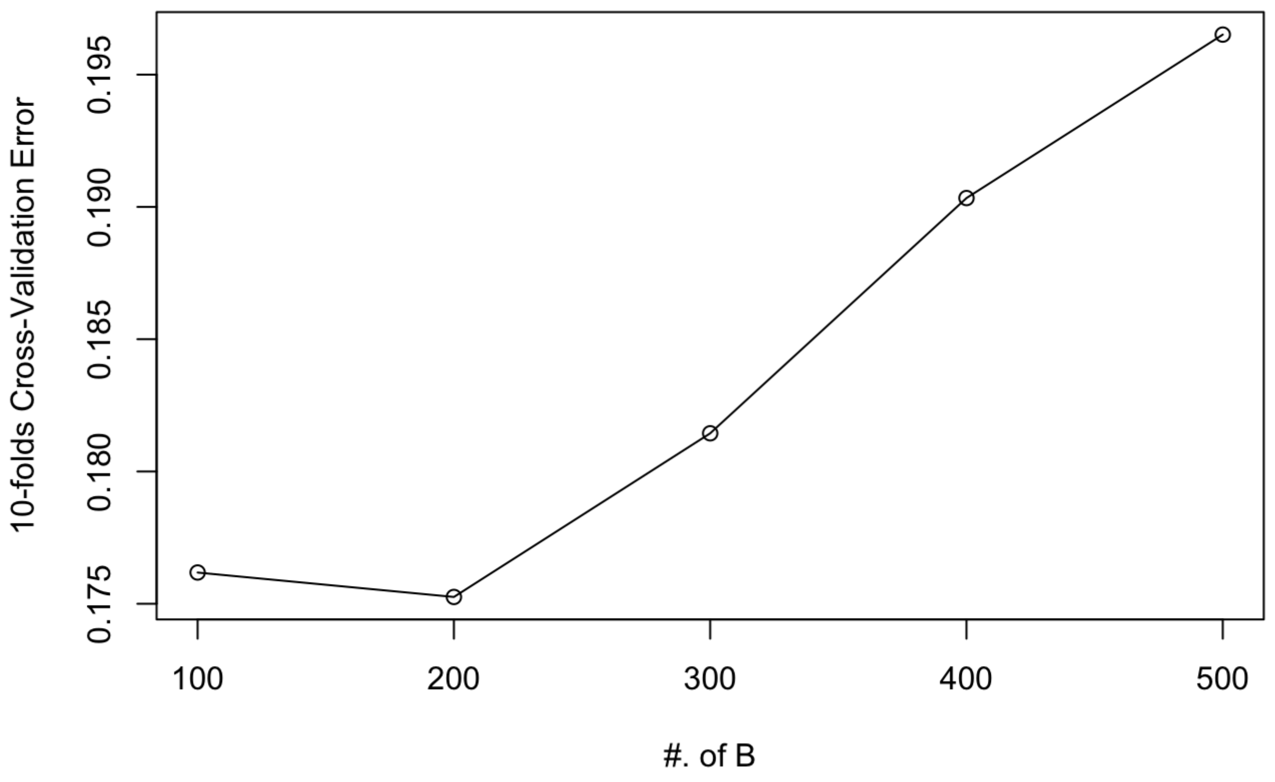
In the regression problem, we used a random forest to predict the house prices in a brawny way. For the classification problem, we try to use a gentle way to learn the data pattern like those men who sacrifice themselves to save women lives. AdaBoost is an extension of boosting which grows trees sequentially. Each tree will only split once based on the feedback given by the previous trees. As said, the learning process is very slow, but it is robust to study some region of the EVs' space that other algorithms do not focus on.

```r
library(ada)
library(rpart.plot)

ctrl = rpart.control(cp = -1, maxdepth = 14, maxcompete = 1, xval = 0)

ada_cv = function(titanic, train, K) {
  data = titanic[train, ]
  rownames(data) = NULL
  n = nrow(data)
  fold = K
  foldIndex = sample(fold, n, replace = T)
  B = seq(100, 500, 100)
  len = length(B)
  BE = rep(0, len)
  for(j in 1:len) {
    error = rep(0, fold)
    for(i in 1:fold) {
      train = foldIndex != i
      test = foldIndex == i
      ada_cvm = ada(Survived ~ ., data = data[train, ], type = "gentle", control = ctrl, ite
r = B[j])
      ada_cvp = predict(ada_cvm, newdata = data[test, ])
      error[i] = 1 - mean(ada_cvp == data$Survived[test])
    }
    BE[j] = mean(error)
    ## print(paste0("Cross-Validation error for B = ", B[j], " is ", BE[j]))
    ## [1] "Cross-Validation error for B = 100 is 0.176181824450102"
    ## [1] "Cross-Validation error for B = 200 is 0.175257384947834"
    ## [1] "Cross-Validation error for B = 300 is 0.181443500419664"
    ## [1] "Cross-Validation error for B = 400 is 0.190334876645067"
    ## [1] "Cross-Validation error for B = 500 is 0.19651327271708"
  }
  return(list(BE, B[which(BE == min(BE))], min(BE)))
}

cve = ada_cv(titanic, train, 10)
plot(seq(100, 500, 100), cve[[1]], type = "o", xlab = "#. of B", ylab = "10-folds Cross-Vali
dation Error")
```



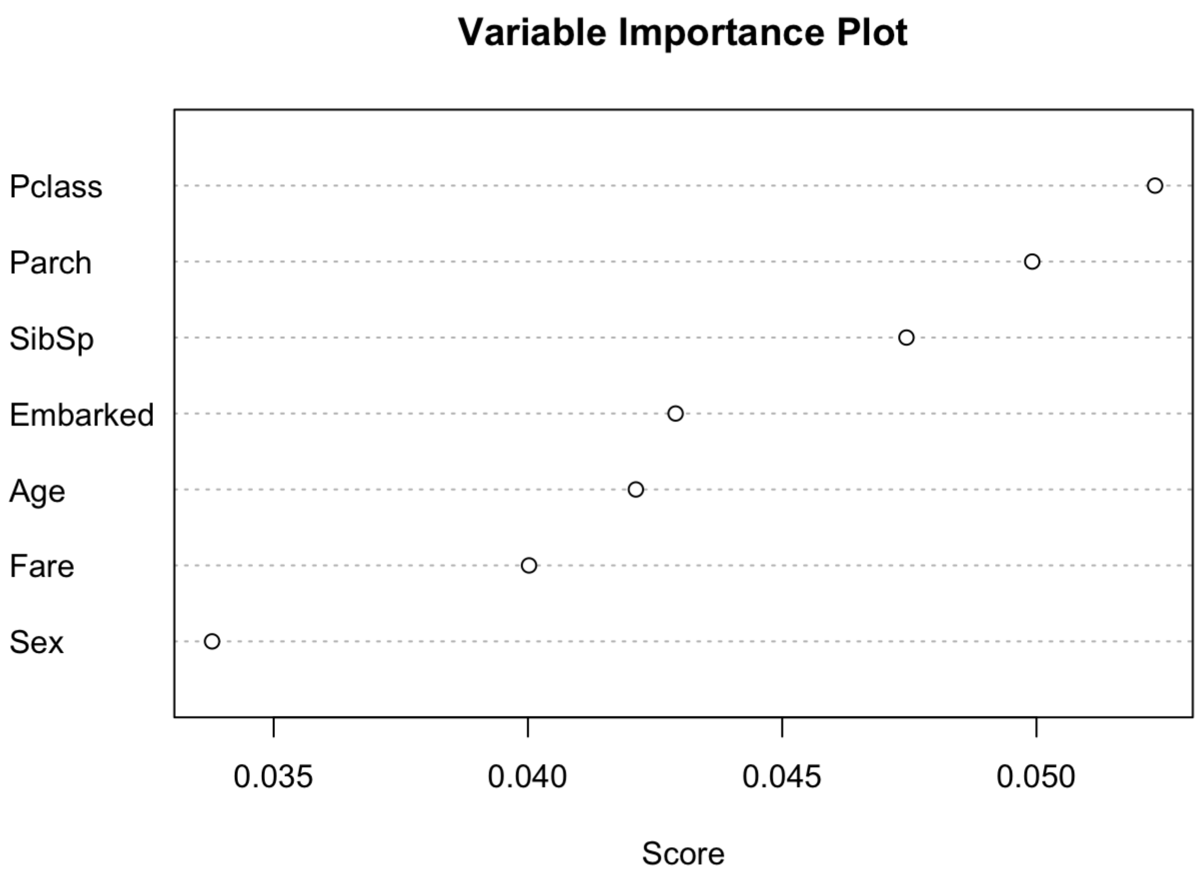
The learning process of AdaBoost is controlled by a shrinkage parameter λ, which usually be 0.01 or 0.001. In our algorithm, we restrict it to be 0.1 to speed up learning time a little bit. Also, the number of trees B in the forest depends on the value of λ, which is a tuning parameter. We, hence, use 10-folds cross-validation to select it. For simplicity and limited computing power, we only consider in a unit of 100 trees per iteration, and the maximum is 500 trees. From the above graph, we notice the optimal value for B is 200. The cross-validation error increase afterwards. So, we will grow a forest with 200 trees to achieve our goal.

```
ada_mod = ada(Survived ~ ., data = titanic[train, ], type = "gentle", control = ctrl, iter =
cve[[2]])
summary(ada_mod)
```

```
## Call:
## ada(Survived ~ ., data = titanic[train, ], type = "gentle", control = ctrl,
##     iter = cve[[2]])
##
## Loss: exponential Method: gentle   Iteration: 200
##
## Training Results
##
## Accuracy: 0.92 Kappa: 0.831
```

The above summary table tells us a crazy fact that it attains 92% accuracy in the train data, and we believe AdaBoost will perform extremely well in the test data as well.

```
varplot(ada_mod)
```



Similar to the random forest, Adaboost also records the importance of each EV. 'Pclass' ranks the most crucial for splitting a node. It implies that the chance of survival highly depends on the tickle class that a passenger holds. Recall the discussion in EDA, the graph further verifies the importance of 'Pclass' in the 'Titanic' data. It gives us noteworthy information on classifying the status of passengers for sure.

```
ada_pre = predict(ada_mod, newdata = titanic[test, ])

result$cla_pro[4, 1] = cve[[3]]
result$cla_pro[4, 2] = 1 - mean(ada_pre == titanic$Survived[test])
result$cla_pro[4, 3] = 1 - F1_Score(ada_pre, titanic$Survived[test], NULL)
result$cla_pro[4, ]
```

```
##          Cross.Validation.Error Prediction.Error Balanced.Error
## Boosting              0.1752574              0.2      0.1428571
```

Not surprising, the performance of AdaBoost beats other models that we have implemented. The prediction error is 0.2, and the 10-folds cross-validation error is 0.1752574. The AdaBoost algorithm can be improved with a smaller learning rate and larger forest. Having said that, the boosting method already accomplishes an excellent result for classifying the status of passengers.

Summary of Classification Problem

```
result$cla_pro
```

```
##                     Cross.Validation.Error Prediction.Error Balanced.Error
## k-Nearest Neighbors              0.2688208             0.26      0.1780822
## Logistic Regression              0.1421774             0.28      0.2000000
## Linear Discriminant                  0.1942697            0.22      0.1549296
## Boosting                             0.1752574            0.20      0.1428571
```

In the above summary table, although the cross-validation error for logistic regression attains the lowest one among all the proposed models, the performance is not very well. It is reasonable since logistic regression is the only model that we have implement model selection for the parsimonious fit. Beside, although we measure the performance using prediction error, we have recorded the balanced accuracy for each model. It takes the advantages of the confusion matrix by considering the type of errors in prediction. From the above summary table, no matter which measurement is used, the result remains the same. It is no doubt that boosting is the best among other proposed models. The logistic regression is a little bit inferior but still achieve 70% accuracy. When we come to the classification problem, it is more favourable to apply dimensional reduction. The idea of LDA is simpler than boosting but still can result in a comparable effect. Furthermore, KNN uses distance measurement to find similar passengers for prediction. If the used EVs that can be scaled, the outcome may be ameliorated. Again, it motivates us to have further study in other learning algorithms for the classification problem.

Summary

The project spotlights on the 'House' and 'Titanic' data from supervised learning algorithms. From both the regression problem and the classification problem, we notice that tree-based methods are extremely muscular in predicting the house prices and the survival passengers. Well done, the random forest and Adaboost. Yet, there are many more learning algorithm can achieve the same goal but not covered in the project. We have gone through some popular supervised learning algorithms, but the results can be concluded from unsupervised learning models. The outputs for those learning methods can result in comparable results as well. Further investigation on implementing those models may give us more insight into the 'real-problem' that we encountered, and we will leave it into the near future.

Reference

B.Leo, C.Adele (2004) *Random Forests*. University of California, Berkeley. 23 November 2020 https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm retrieved

C.Marina (2020) *A Quick Introduction to KNN Algorithm*. Great Learning Blog. 29 November 2020 https://www.mygreatlearning.com/blog/knn-algorithm-introduction/ retrieved

A work by Jack CHAN

1155119394@link.cuhk.edu.hk