AMA4602 Midterm project

Leung chun kit

21018713D

Data source : https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009

The data set are related to red and white variants of the Portuguese "Vinho Verde" wine.

Input variables (based on physicochemical tests):

1 - fixed acidity

2 - volatile acidity

3 - citric acid

4 - residual sugar

5 - chlorides

6 - free sulfur dioxide

7 - total sulfur dioxide

8 - density

9 - pH

10 - sulphates

11 - alcohol

Output variable (based on sensory data):

12 - quality (score between 0 and 10)

I am going to use those features to build a model to predict the quality score of wines.

Import librarys

```
library(corrplot)
library(RColorBrewer)
library(ggfortify)
library(ggplot2)
library(dplyr)
library(class)
library(mlbench)
library(tibble)
library(rpart)
library(randomForest)
library(corrplot)
```

```r
library(tidyverse)
library(leaps)
library(tidyverse)
library(caret)
library(e1071)
library(caTools)
library(caret)
library(car)
library(MASS)
library(ggplot2)
```

Set seed

```r
set.seed(111)
```

Load the data

```r
data <-read.csv('winequality-red.csv',header = TRUE)
attach(data)
```

Quick check for the data

```r
is.null(data)
```

```
## [1] FALSE
```

```r
dim(data)
```

```
## [1] 1599   12
```

```r
head(data)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
```

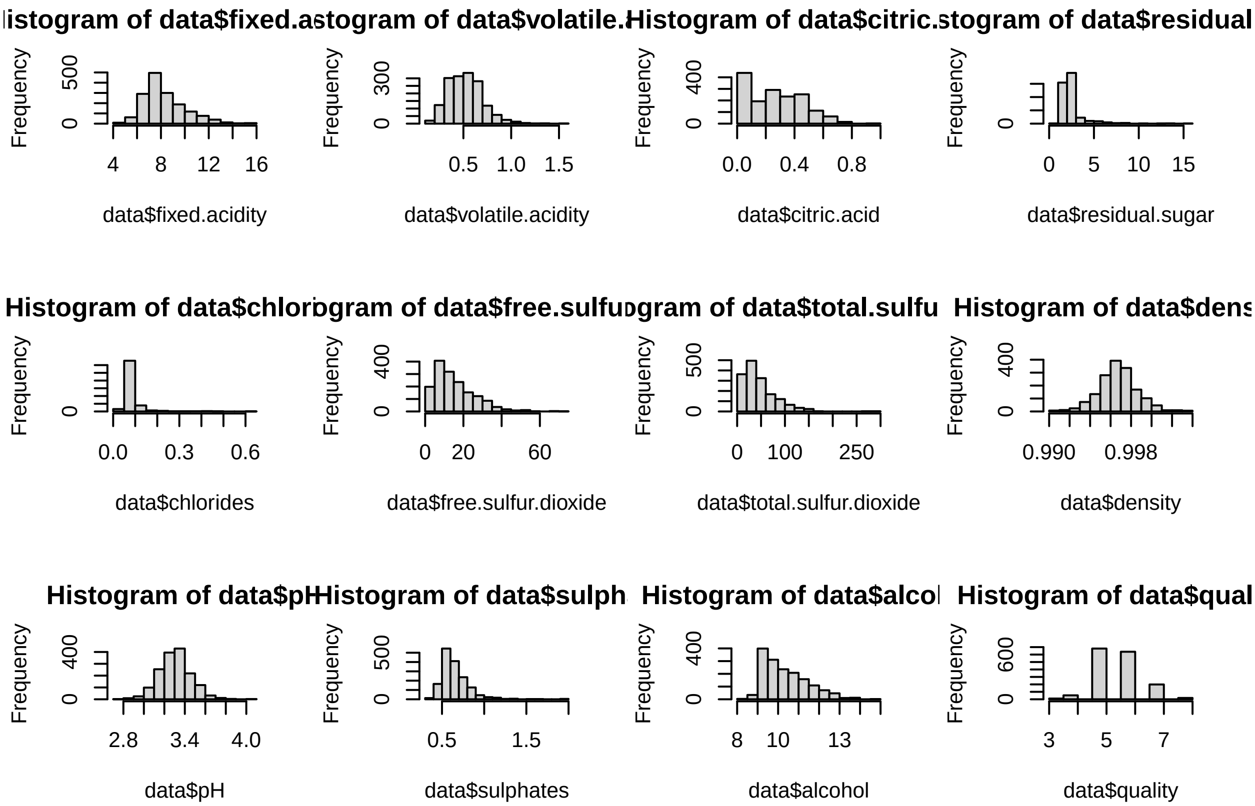
```
## 4                  17                    60  0.9980 3.16      0.58     9.8
## 5                  11                    34  0.9978 3.51      0.56     9.4
## 6                  13                    40  0.9978 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

The data set has 1599 observations and 12 columns. The first 11 columns are the features and the 12 column is the response.

There is not missing value in the dataset.

Distribution of the data

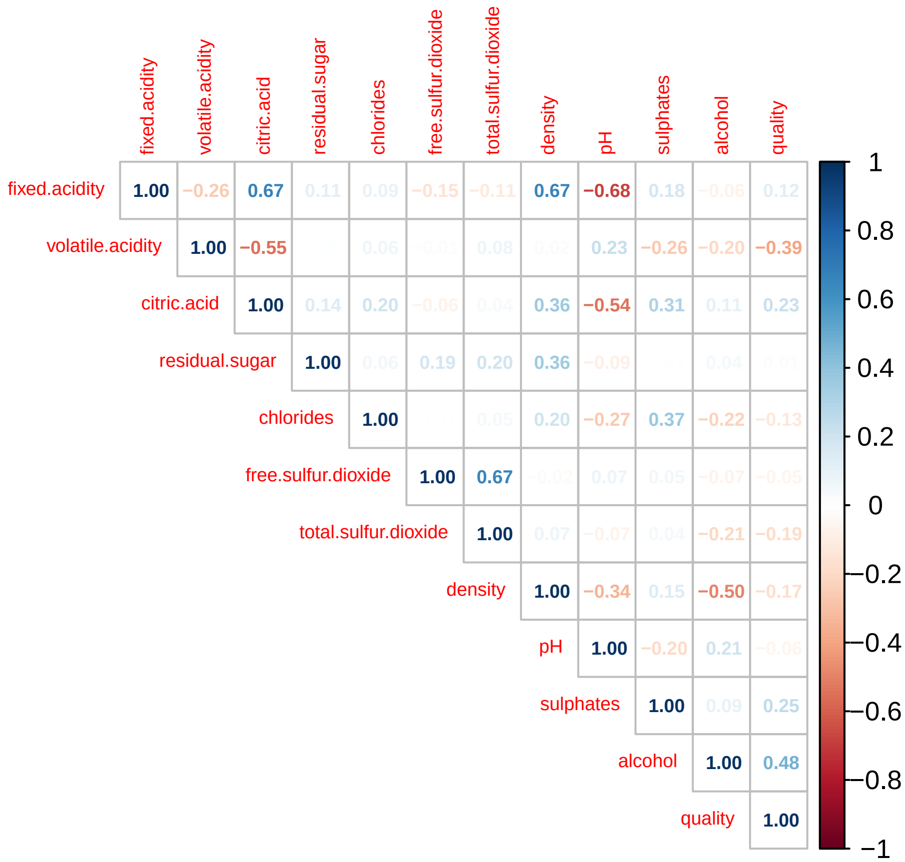
```
par(mfrow =c(3,4))
hist(data$fixed.acidity)
hist(data$volatile.acidity)
hist(data$citric.acid)
hist(data$residual.sugar)
hist(data$chlorides)
hist(data$free.sulfur.dioxide)
hist(data$total.sulfur.dioxide)
hist(data$density)
hist(data$pH)
hist(data$sulphates)
hist(data$alcohol)
hist(data$quality)
```



The scatter plot among difference feature and response

Correlation between diffience features

```
M <-cor(data)
corrplot(cor(data),
         method = "number",
         type = "upper", number.cex = 0.6 ,tl.cex = 0.6)
```



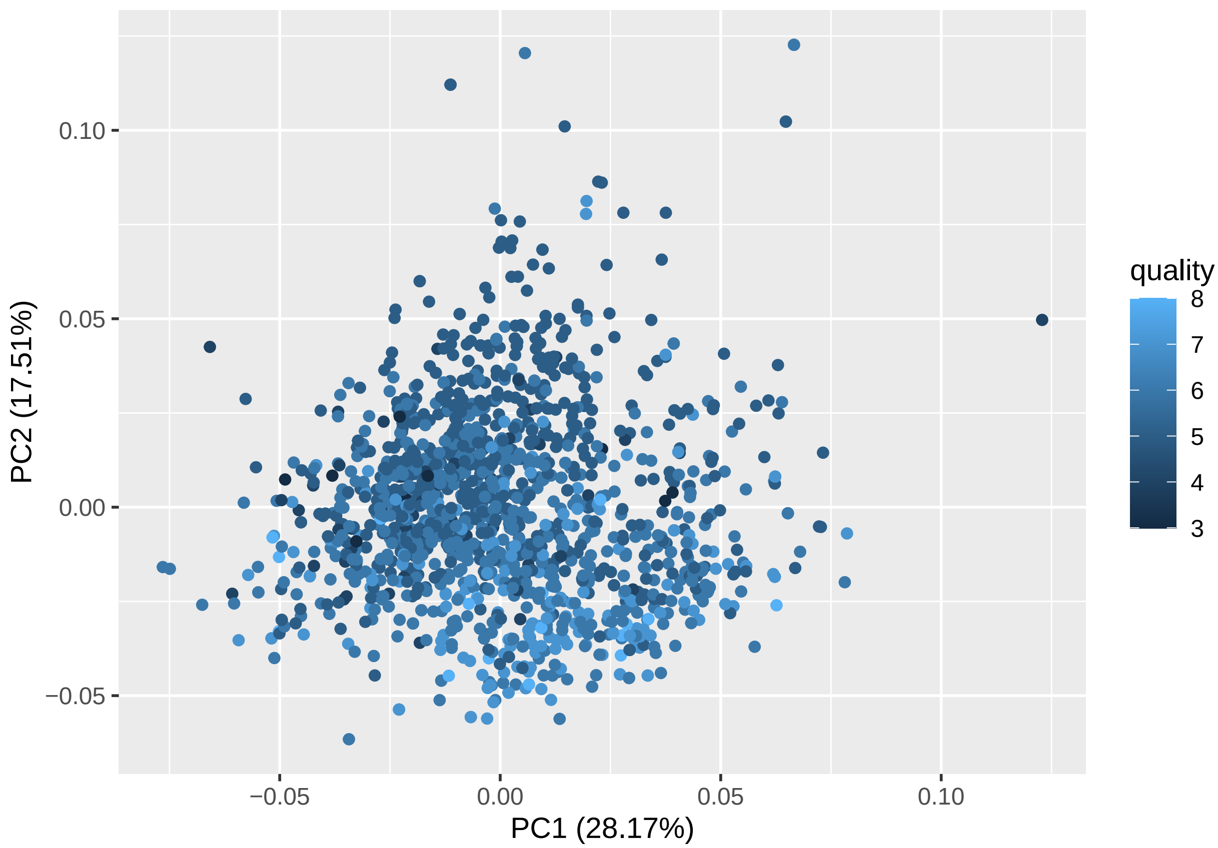
The heat map of correlation matrix shows the correlation matrix between different features and response. Aclochol has the highest correlation with the quality of wine.

PCA analysis

```r
# PCA analysis

# data.pr <- prcomp( data[c(1:11)], center = TRUE, scale = FALSE)
data.pr <- prcomp( data[c(1:11)], center = TRUE, scale = TRUE)
summary(data.pr)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.7604 1.3878 1.2452 1.1015 0.97943 0.81216 0.76406
## Proportion of Variance 0.2817 0.1751 0.1410 0.1103 0.08721 0.05996 0.05307
## Cumulative Proportion  0.2817 0.4568 0.5978 0.7081 0.79528 0.85525 0.90832
##                            PC8     PC9    PC10    PC11
## Standard deviation     0.65035 0.58706 0.42583 0.24405
## Proportion of Variance 0.03845 0.03133 0.01648 0.00541
## Cumulative Proportion  0.94677 0.97810 0.99459 1.00000
```

```r
autoplot(data.pr, data = data, colour = 'quality')
```



The first two PC only can explain 45.68% of variation of data. And red wire with difference type still overlap with each other on the first two PC. We may consider to use more pc to explain the variation of data.

Data splitting

```
index = sample(nrow(data), floor(nrow(data) * 0.8))
train = data[index,]
test = data[-index,]

dim(train)
```

```
## [1] 1279   12
```

```
dim(test)
```

```
## [1] 320  12
```

```
xTrain = train[1:11]
yTrain = train$quality

xTest = test[1:11]
yTest = test$quality
```

```
yTrain_str = as.character(yTrain)
yTest_str = as.character(yTest)

train.control <- trainControl(method = "cv", number = 10)
```

We use 80% data as the training data and 20 % as testing data. There are total 1279 observations as training data and 320 observations as testing data. we create two type of response, once treat the response (quality) as numeric and once treat the response as Category (string) (with _cat)

Model training

we are going to fit the data to difference statistic model in order to find out the best model to predict the quality of the wine.

We are using the train function to fit the data to the model

K fold cross validation are used to split the training data to 10 part For each time, use one part of the data to valid the result and use the rest of the data to train the model. We can use the mean of the 10 times iterations to evaluate the model's ability. We will use the accuracy rate and the MAE to evaluate the performance of the model.

For some statistical model, it require some parameter. For example, knn model require the parameter k (which is the number of nearest neighbors to used for predicting new observation) The train function will try different parameter and return the model using the best parameter (Lowest RMSE or highest accuracy)

For the model using numeric response, the predict value may be float value. However, the response should be integer. Therefore, the predict response by those model will be rounded to integer.

```
cal_MAE <-function(predict,real_value){

if (is.numeric(predict) == FALSE) {

  predict = as.integer(predict) +2
  }

MAE = mean(abs(predict - real_value))

return(MAE)
}
```

Linear regression (response = numeric)

```
lr_model_full <- train(xTrain,yTrain , method = "lm",
                trControl = train.control)

print(lr_model_full)
```

```
## Linear Regression
##
## 1279 samples
```

```
##   11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1152, 1151, 1150, 1152, 1151, 1151, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.6715411  0.3220998  0.5244047
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
summary(lr_model_full)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.71574 -0.38634 -0.05389  0.47160  2.10488
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           3.057e+01  2.400e+01   1.273 0.203115
## fixed.acidity         4.641e-02  2.909e-02   1.596 0.110833
## volatile.acidity     -1.130e+00  1.408e-01  -8.020 2.39e-15 ***
## citric.acid          -1.543e-01  1.706e-01  -0.905 0.365901
## residual.sugar        2.462e-02  1.703e-02   1.446 0.148493
## chlorides            -1.966e+00  4.724e-01  -4.161 3.38e-05 ***
## free.sulfur.dioxide   3.529e-03  2.535e-03   1.392 0.164177
## total.sulfur.dioxide -3.246e-03  8.471e-04  -3.832 0.000133 ***
## density              -2.683e+01  2.450e+01  -1.095 0.273574
## pH                   -2.685e-01  2.147e-01  -1.250 0.211353
## sulphates             8.657e-01  1.281e-01   6.757 2.14e-11 ***
## alcohol               2.486e-01  3.004e-02   8.273 3.26e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6674 on 1267 degrees of freedom
## Multiple R-squared:  0.3343, Adjusted R-squared:  0.3285
## F-statistic: 57.84 on 11 and 1267 DF,  p-value: < 2.2e-16
```

```
pred_lr_full = round(predict(lr_model_full,xTest))

res <- table( pred_lr_full,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))
res
```

```
##                   yTest
## pred_lr_full       true result = 3 true result = 4 true result = 5
##   prediction = 4                 1               1               1
```

```
##   prediction = 5                0               7             102
##   prediction = 6                0               1              37
##   prediction = 7                0               0               0
##                 yTest
## pred_lr_full     true result = 6 true result = 7 true result = 8
##   prediction = 4                0               0               0
##   prediction = 5               31               0               0
##   prediction = 6               96              23               3
##   prediction = 7                5              10               2
```

```
lr_model_full.accuracy = mean(pred_lr_full==yTest)

lr_model_full.MAE = cal_MAE(pred_lr_full, yTest)
```

Feature selection by using AIC in MLR (response = numeric)

```
set.seed(1)
slm <- step(lm(quality ~ .,data = data[index,]),direction="backward")
```

```
## Start:  AIC=-1022.55
## quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS      AIC
## - citric.acid           1    0.3644 564.66 -1023.73
## - density               1    0.5344 564.83 -1023.34
## - pH                    1    0.6964 564.99 -1022.98
## - free.sulfur.dioxide   1    0.8630 565.16 -1022.60
## <none>                              564.29 -1022.55
## - residual.sugar        1    0.9309 565.22 -1022.45
## - fixed.acidity         1    1.1339 565.43 -1021.99
## - total.sulfur.dioxide  1    6.5407 570.83 -1009.81
## - chlorides             1    7.7112 572.00 -1007.19
## - sulphates             1   20.3369 584.63  -979.27
## - volatile.acidity      1   28.6469 592.94  -961.22
## - alcohol               1   30.4860 594.78  -957.26
##
## Step:  AIC=-1023.73
## quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     density + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS      AIC
## - density               1     0.554 565.21 -1024.47
## - pH                    1     0.640 565.30 -1024.28
## - fixed.acidity         1     0.841 565.50 -1023.82
## - residual.sugar        1     0.882 565.54 -1023.73
## <none>                              564.66 -1023.73
## - free.sulfur.dioxide   1     1.083 565.74 -1023.28
```

```
## - total.sulfur.dioxide  1     8.017 572.67 -1007.70
## - chlorides             1     9.225 573.88 -1005.00
## - sulphates             1    20.294 584.95  -980.57
## - alcohol               1    30.217 594.87  -959.05
## - volatile.acidity      1    35.449 600.11  -947.85
##
## Step:  AIC=-1024.47
## quality ~ fixed.acidity + volatile.acidity + residual.sugar +
##     chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
##     pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS      AIC
## - fixed.acidity         1     0.289 565.50 -1025.82
## - residual.sugar        1     0.386 565.60 -1025.60
## <none>                              565.21 -1024.47
## - free.sulfur.dioxide   1     1.252 566.46 -1023.64
## - pH                    1     2.211 567.42 -1021.48
## - total.sulfur.dioxide  1     8.423 573.63 -1007.56
## - chlorides             1     9.857 575.07 -1004.36
## - sulphates             1    19.965 585.18  -982.08
## - volatile.acidity      1    37.784 603.00  -943.71
## - alcohol               1    85.487 650.70  -846.33
##
## Step:  AIC=-1025.82
## quality ~ volatile.acidity + residual.sugar + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS      AIC
## - residual.sugar        1     0.478 565.98 -1026.74
## <none>                              565.50 -1025.82
## - free.sulfur.dioxide   1     1.225 566.73 -1025.05
## - pH                    1     5.956 571.46 -1014.42
## - total.sulfur.dioxide  1     9.228 574.73 -1007.12
## - chlorides             1    10.429 575.93 -1004.45
## - sulphates             1    20.582 586.08  -982.10
## - volatile.acidity      1    38.442 603.94  -943.71
## - alcohol               1    85.456 650.96  -847.83
##
## Step:  AIC=-1026.74
## quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol
##
##                        Df Sum of Sq    RSS      AIC
## <none>                              565.98 -1026.74
## - free.sulfur.dioxide   1     1.247 567.22 -1025.93
## - pH                    1     6.317 572.29 -1014.55
## - total.sulfur.dioxide  1     8.813 574.79 -1008.98
## - chlorides             1    10.273 576.25 -1005.74
## - sulphates             1    20.321 586.30  -983.63
## - volatile.acidity      1    38.302 604.28  -944.99
## - alcohol               1    88.896 654.87  -842.15
```

```
summary(slm)
```

```
##
## Call:
## lm(formula = quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol, data = data[index,
##     ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6706 -0.3880 -0.0494  0.4713  2.0990
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.7707424  0.4531540  10.528  < 2e-16 ***
## volatile.acidity     -1.0866562  0.1171674  -9.274  < 2e-16 ***
## chlorides            -2.1596732  0.4496519  -4.803 1.75e-06 ***
## free.sulfur.dioxide   0.0041645  0.0024887   1.673 0.094501 .
## total.sulfur.dioxide -0.0035211  0.0007915  -4.449 9.39e-06 ***
## pH                   -0.4985847  0.1323787  -3.766 0.000173 ***
## sulphates             0.8294653  0.1227877   6.755 2.16e-11 ***
## alcohol               0.2717625  0.0192342  14.129  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6673 on 1271 degrees of freedom
## Multiple R-squared:  0.3323, Adjusted R-squared:  0.3286
## F-statistic: 90.37 on 7 and 1271 DF,  p-value: < 2.2e-16
```

```
set.seed(1)
lr_model_selected <- train(quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + pH + sulphates + alcohol,data = data[index,],method = "lm", trControl = trai

summary(lr_model_selected)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6706 -0.3880 -0.0494  0.4713  2.0990
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.7707424  0.4531540  10.528  < 2e-16 ***
## volatile.acidity     -1.0866562  0.1171674  -9.274  < 2e-16 ***
## chlorides            -2.1596732  0.4496519  -4.803 1.75e-06 ***
## free.sulfur.dioxide   0.0041645  0.0024887   1.673 0.094501 .
## total.sulfur.dioxide -0.0035211  0.0007915  -4.449 9.39e-06 ***
## pH                   -0.4985847  0.1323787  -3.766 0.000173 ***
## sulphates             0.8294653  0.1227877   6.755 2.16e-11 ***
## alcohol               0.2717625  0.0192342  14.129  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.6673 on 1271 degrees of freedom
## Multiple R-squared:  0.3323, Adjusted R-squared:  0.3286
## F-statistic: 90.37 on 7 and 1271 DF,  p-value: < 2.2e-16
```

```r
print(lr_model_selected)
```

```
## Linear Regression
##
## 1279 samples
##    7 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1151, 1150, 1151, 1151, 1151, 1152, ...
## Resampling results:
##
##   RMSE       Rsquared  MAE
##   0.6684154  0.329266  0.5243709
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
pred_lr_selected = round(predict(lr_model_selected,xTest))

#The accuracy table:
res <- table(pred_lr_selected ,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))

res
```

```
##                 yTest
## pred_lr_selected true result = 3 true result = 4 true result = 5
##   prediction = 4               1               1               1
##   prediction = 5               0               7             102
##   prediction = 6               0               1              37
##   prediction = 7               0               0               0
##                 yTest
## pred_lr_selected true result = 6 true result = 7 true result = 8
##   prediction = 4               0               0               0
##   prediction = 5              31               0               0
##   prediction = 6              95              23               3
##   prediction = 7               6              10               2
```

```r
lr_model_selected.accuracy = mean(pred_lr_selected==yTest)
lr_model_selected.MAE = cal_MAE(round(predict(lr_model_selected,xTest)), yTest)
```

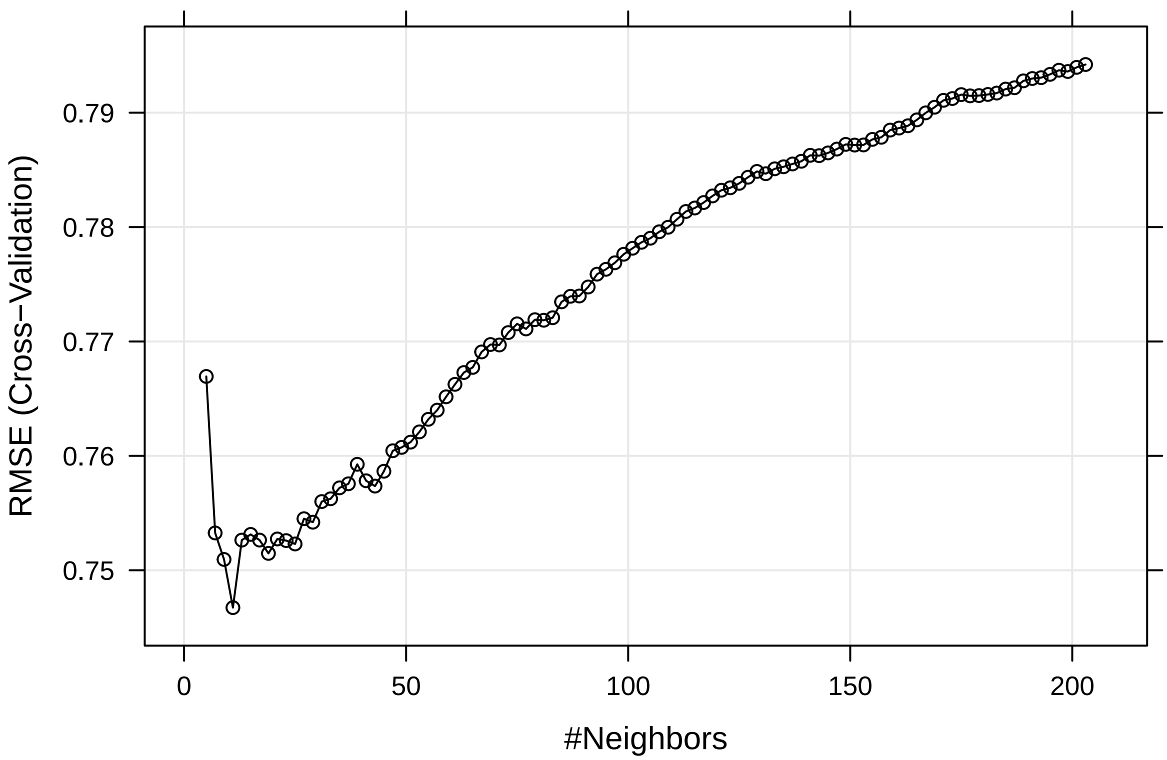
After feature selection, the model lefts feature volatile.acidity + chlorides + free.sulfur.dioxide + total.sulfur.dioxide + pH + sulphates + alcohol

K Nearest Neighbors (response = numeric)

```
set.seed(1)
knn_model <- train(xTrain,yTrain ,  method = "knn",
               trControl = train.control, tuneLength = 100)
summary(knn_model)
```

```
##              Length Class      Mode
## learn         2     -none-     list
## k             1     -none-     numeric
## theDots       0     -none-     list
## xNames       11     -none-     character
## problemType   1     -none-     character
## tuneValue     1     data.frame list
## obsLevels     1     -none-     logical
## param         0     -none-     list
```

```
trellis.par.set(caretTheme())
plot(knn_model)
```




```
pred_knn <- round(predict(knn_model,xTest))

#The accuracy table:
```

```r
res <- table(pred_knn ,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))
res
```

```
##                  yTest
## pred_knn          true result = 3 true result = 4 true result = 5
##   prediction = 5               0               5              85
##   prediction = 6               1               4              54
##   prediction = 7               0               0               1
##                  yTest
## pred_knn          true result = 6 true result = 7 true result = 8
##   prediction = 5              43               2               1
##   prediction = 6              86              28               4
##   prediction = 7               3               3               0
```

```r
knn_model.accuracy = mean(pred_knn==yTest)
knn_model.MAE <- cal_MAE(pred_knn, yTest)
```

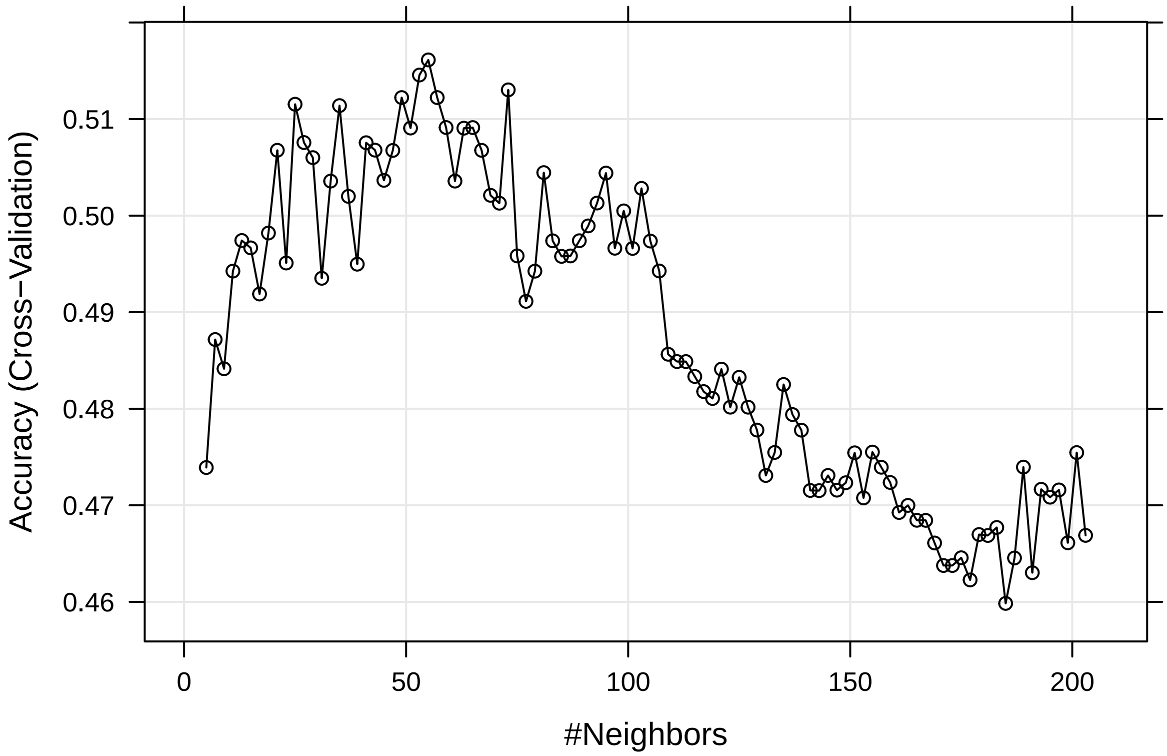
Since the Knn model has a parameter k (use how many neighborhood to predict the response), the train model will choose the k with high accuracy. (each time may be difference base on the combination of testing and training data)

K Nearest Neighbors (response = Category)

```r
set.seed(1)
knn_model_cat <- train(xTrain,yTrain_str ,  method = "knn",
               trControl = train.control, tuneLength = 100)

summary(knn_model_cat)
```

```
##             Length Class      Mode
## learn        2     -none-     list
## k            1     -none-     numeric
## theDots      0     -none-     list
## xNames      11     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    6     -none-     character
## param        0     -none-     list
```

```r
trellis.par.set(caretTheme())
plot(knn_model_cat)
```




```r
pred_knn_cat <-predict(knn_model_cat,xTest)

#The accuracy table:
res <- table(pred_knn_cat,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))
res
```

```
##                  yTest
## pred_knn_cat      true result = 3 true result = 4 true result = 5
##   prediction = 3               0               0               0
##   prediction = 4               0               0               0
##   prediction = 5               1               6              95
##   prediction = 6               0               3              45
##   prediction = 7               0               0               0
##   prediction = 8               0               0               0
##                  yTest
## pred_knn_cat      true result = 6 true result = 7 true result = 8
##   prediction = 3               0               0               0
##   prediction = 4               0               0               0
##   prediction = 5              51              10               2
##   prediction = 6              80              22               3
##   prediction = 7               1               1               0
##   prediction = 8               0               0               0
```

```r
knn_model_cat.accuracy = mean(pred_knn_cat == yTest)
knn_model_cat.MAE = cal_MAE(pred_knn_cat, yTest)
```

Since the Knn model has a parameter k (use how many neighborhood to predict the response), the train model will choose the k with high accuracy. (each time may be difference base on the combination of testing and training data)

Random Forest (response = numeric)

```r
set.seed(1)
rf_model <- train(xTrain,yTrain , method = "rf",
               trControl = train.control)

summary(rf_model)
```

```
##                 Length Class      Mode
## call               4   -none-     call
## type               1   -none-     character
## predicted       1279   -none-     numeric
## mse              500   -none-     numeric
## rsq              500   -none-     numeric
## oob.times       1279   -none-     numeric
## importance        11   -none-     numeric
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
## mtry               1   -none-     numeric
## forest            11   -none-     list
## coefs              0   -none-     NULL
## y               1279   -none-     numeric
## test               0   -none-     NULL
## inbag              0   -none-     NULL
## xNames            11   -none-     character
## problemType        1   -none-     character
## tuneValue          1   data.frame list
## obsLevels          1   -none-     logical
## param              0   -none-     list
```

```r
print(rf_model)
```

```
## Random Forest
##
## 1279 samples
##   11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1151, 1150, 1151, 1151, 1151, 1152, ...
```

```
## Resampling results across tuning parameters:
##
##   mtry  RMSE       Rsquared   MAE
##    2    0.5903607  0.4866615  0.4393690
##    6    0.5890098  0.4826466  0.4314920
##   11    0.5929309  0.4744164  0.4324376
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 6.
```

```
pred_rf <- round(predict(rf_model,xTest))

#The accuracy table:
res <- table( pred_rf,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))
res
```

```
##                  yTest
## pred_rf            true result = 3 true result = 4 true result = 5
##   prediction = 5                 1               9             109
##   prediction = 6                 0               0              29
##   prediction = 7                 0               0               2
##                  yTest
## pred_rf            true result = 6 true result = 7 true result = 8
##   prediction = 5                26               0               0
##   prediction = 6                94              17               3
##   prediction = 7                12              16               2
```

```
rf_model.accuracy = mean( pred_rf ==yTest)
rf_model.MAE = cal_MAE(pred_rf, yTest)
```

Random Forest (response = Category)

```
set.seed(1)
rf_model_cat <- train(xTrain,yTrain_str , method = "rf",
               trControl = train.control)

summary(rf_model_cat)
```

```
##                 Length Class      Mode
## call               4   -none-     call
## type               1   -none-     character
## predicted       1279   factor     numeric
## err.rate        3500   -none-     numeric
## confusion         42   -none-     numeric
## votes           7674   matrix     numeric
## oob.times       1279   -none-     numeric
## classes            6   -none-     character
## importance        11   -none-     numeric
```

```
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
## mtry               1   -none-     numeric
## forest            14   -none-     list
## y               1279   factor     numeric
## test               0   -none-     NULL
## inbag              0   -none-     NULL
## xNames            11   -none-     character
## problemType        1   -none-     character
## tuneValue          1   data.frame list
## obsLevels          6   -none-     character
## param              0   -none-     list
```

```
print(rf_model_cat)
```

```
## Random Forest
##
## 1279 samples
##   11 predictor
##    6 classes: '3', '4', '5', '6', '7', '8'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1153, 1151, 1150, 1152, 1151, 1151, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.6897283  0.4991499
##    6    0.6858276  0.4955747
##   11    0.6811454  0.4888324
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

```
pred_rf_cat <- predict(rf_model_cat,xTest)

res <- table( pred_rf_cat ,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))

res
```

```
##                  yTest
## pred_rf_cat        true result = 3 true result = 4 true result = 5
##   prediction = 3                 0               0               0
##   prediction = 4                 0               0               0
##   prediction = 5                 1               9             115
##   prediction = 6                 0               0              23
##   prediction = 7                 0               0               2
##   prediction = 8                 0               0               0
##                  yTest
```

```
## pred_rf_cat      true result = 6 true result = 7 true result = 8
##   prediction = 3                0               0               0
##   prediction = 4                0               0               0
##   prediction = 5               25               1               0
##   prediction = 6               99              14               2
##   prediction = 7                8              18               3
##   prediction = 8                0               0               0
```

```
rf_model_cat.accuracy = mean(pred_rf_cat ==yTest)
rf_model_cat.accuracy
```

```
## [1] 0.725
```

```
rf_model_cat.MAE <- cal_MAE(pred_rf_cat, yTest)
```

Naive Bayes Classification (response = Category)

```
set.seed(1)
nb_model <- train(xTrain,yTrain_str , method = "nb",
               trControl = train.control)

summary(nb_model)
print(nb_model)

pred_nb <- predict(nb_model,xTest)


res <- table(pred_nb ,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))

res

nb_model.accuracy = mean(pred_nb ==yTest)
nb_model.accuracy

nb_model.MAE <- cal_MAE(pred_nb, yTest)
```

Linear discriminant analysis (response = Category)

```
lda_model <- train(xTrain,yTrain_str , method = "lda",
               trControl = train.control)

summary(lda_model)
```

```
##              Length Class      Mode
```

```
## prior        6     -none-     numeric
## counts       6     -none-     numeric
## means       66     -none-     numeric
## scaling     55     -none-     numeric
## lev          6     -none-     character
## svd          5     -none-     numeric
## N            1     -none-     numeric
## call         3     -none-     call
## xNames      11     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    6     -none-     character
## param        0     -none-     list
```

```
print(lda_model)
```

```
## Linear Discriminant Analysis
##
## 1279 samples
##   11 predictor
##    6 classes: '3', '4', '5', '6', '7', '8'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 1152, 1152, 1151, 1150, 1152, 1151, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5777895  0.3264941
```

```
pred_lda <- predict(lda_model,xTest)

res <- table( pred_lda ,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))

res
```

```
##                  yTest
## pred_lda          true result = 3 true result = 4 true result = 5
##   prediction = 3               1               1               0
##   prediction = 4               0               0               0
##   prediction = 5               0               7             108
##   prediction = 6               0               1              29
##   prediction = 7               0               0               3
##   prediction = 8               0               0               0
##                  yTest
## pred_lda          true result = 6 true result = 7 true result = 8
##   prediction = 3               0               0               0
##   prediction = 4               2               0               0
##   prediction = 5              31               1               0
##   prediction = 6              81              12               2
##   prediction = 7              18              20               3
##   prediction = 8               0               0               0
```

```r
lda_model.accuracy = mean(pred_lda == yTest)
lda_model.accuracy
```

```
## [1] 0.65625
```

```r
lda_model.MAE = cal_MAE(pred_lda, yTest)
```

Support vector machine (response = numeric)

```r
set.seed(1)
svm_model <- train(xTrain,yTrain , method = "svmPoly",
                trControl = train.control )

pred_svm <- round(predict(svm_model,xTest))

res <- table(pred_svm  ,yTest)
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))
res
```

```
##                   yTest
## pred_svm          true result = 3 true result = 4 true result = 5
##   prediction = 4               0               1               0
##   prediction = 5               1               8             104
##   prediction = 6               0               0              36
##   prediction = 7               0               0               0
##                   yTest
## pred_svm          true result = 6 true result = 7 true result = 8
##   prediction = 4               0               0               0
##   prediction = 5              35               1               0
##   prediction = 6              86              17               2
##   prediction = 7              11              15               3
```

```r
svm_model.accuracy = mean(pred_svm  ==yTest)
svm_model.MAE <- cal_MAE(pred_svm, yTest)
```

Support vector machine (response = Category)

```r
set.seed(1)
svm_model_cat <- train(xTrain,yTrain_str , method = "svmPoly",
                trControl = train.control)

pred_svm_cat <- predict(svm_model_cat,xTest)


res <- table(pred_svm_cat,yTest)
```

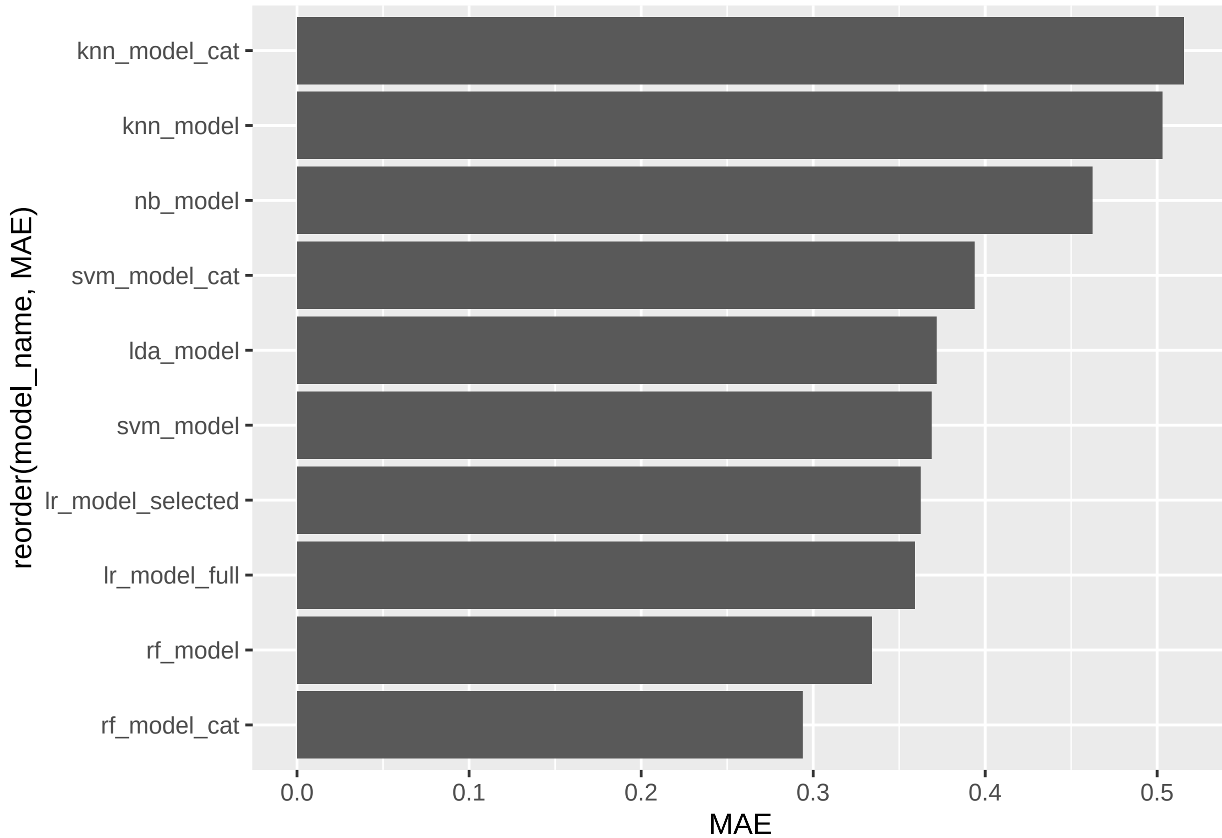
```r
rownames(res) <-paste0('prediction = ', rownames(res))
colnames(res) <- paste0('true result = ', colnames(res))
res
```

```
##                yTest
## pred_svm_cat     true result = 3 true result = 4 true result = 5
##   prediction = 3               0               0               1
##   prediction = 4               0               0               1
##   prediction = 5               1               8             107
##   prediction = 6               0               1              29
##   prediction = 7               0               0               2
##   prediction = 8               0               0               0
##                yTest
## pred_svm_cat     true result = 6 true result = 7 true result = 8
##   prediction = 3               0               0               0
##   prediction = 4               0               1               0
##   prediction = 5              40               2               0
##   prediction = 6              82              13               3
##   prediction = 7              10              17               2
##   prediction = 8               0               0               0
```

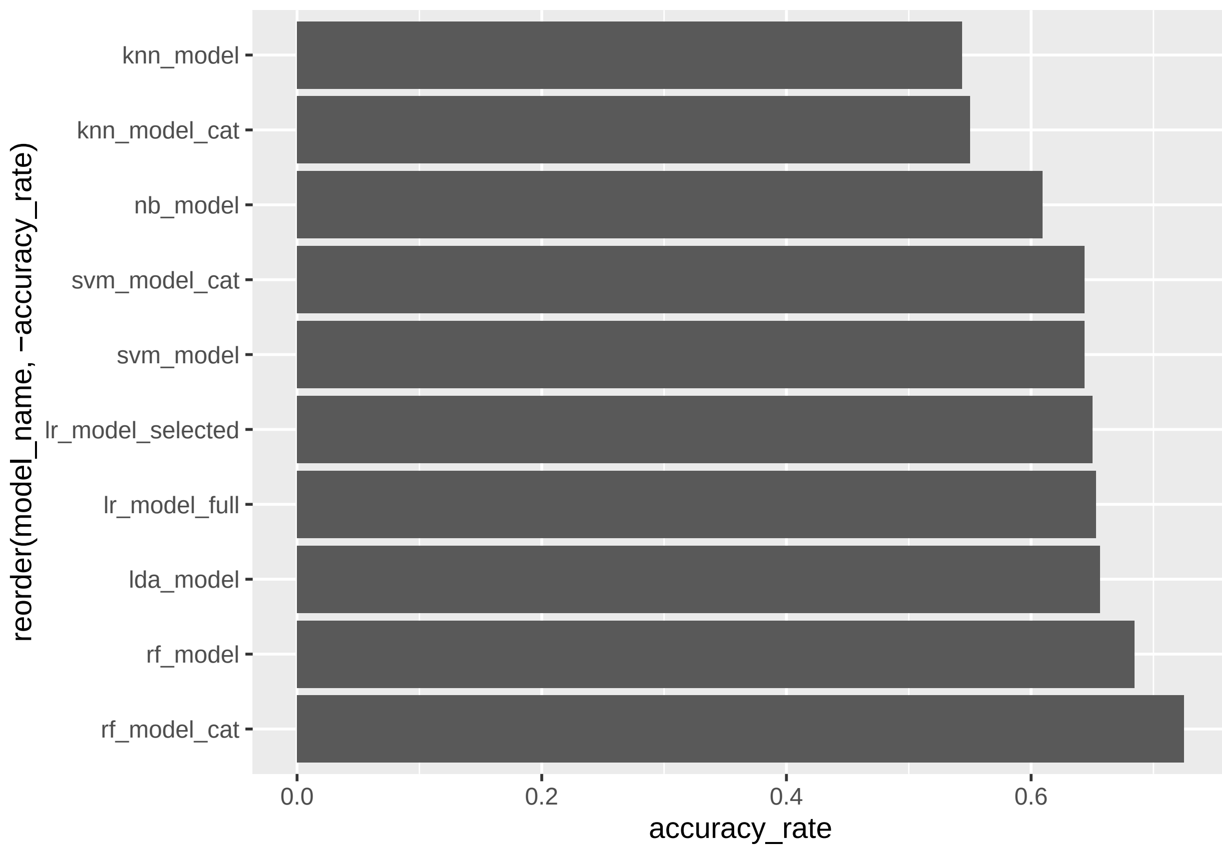
```r
svm_model_cat.accuracy = mean(pred_svm_cat ==yTest)
svm_model_cat.MAE <- cal_MAE(pred_svm_cat, yTest)
```

Comparison between different models

```r
model_name <- c('lr_model_full' , 'lr_model_selected' , 'knn_model' , 'knn_model_cat' , 'rf_model' , 'r

MAE = c(lr_model_full.MAE , lr_model_selected.MAE , knn_model.MAE , knn_model_cat.MAE , rf_model.MAE , r

accuracy_rate = c(lr_model_full.accuracy, lr_model_selected.accuracy, knn_model.accuracy, knn_model_cat

accuracy_df <- data.frame (
                  model_name = model_name,
                  MAE  = MAE,
                  accuracy_rate =accuracy_rate
                  )
par(mfrow = c(1,2))
ggplot(data = accuracy_df, aes(x = reorder(model_name,MAE), y = MAE)) +  geom_bar(stat="identity") + coo
```




```
ggplot(data = accuracy_df, aes(x = reorder(model_name,-accuracy_rate), y = accuracy_rate)) +  geom_bar(
```



The Random forest model using the Category type response perform the best, it have the highest accuracy rate and the lowest MAE.

We can observe that the performance measured by MAE and accuracy rate are difference.

MAE is more sensitive with the variation of the error.(the distance between the prediction and the true value)

Although the accuracy rate of lda model is higher than svm model (numeric response) . But if we consider the variation of the error, the svm model perform better. (the error of Ida is more serious)

For rf model and knn model, using category response to train the model perform better than using numeric response. But for svm model , using numeric response to train the model perform better than using category response.