Machine Learning Project

Trung Duy N Nguyen, ICT 2015

December 6, 2015

Content

I. Dataset description

Name of the dataset: **Wine Quality**

Created by: Paulo Cortez (Univ. Minho), Antonio Cerdeira, Fernando Almeida, Telmo Matos and Jose Reis (CVRVV) @ 2009

Source: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/

Description: The inputs include objective tests (e.g. PH values) and the output is based on sensory data (median of at least 3 evaluations made by wine experts). Each expert graded the wine quality between 0 (very bad) and 10 (very excellent).These datasets can be viewed as classification or regression tasks.

Several data mining methods were applied to model these datasets under a regression approach. The classes are ordered and not balanced (e.g. there are munch more normal wines than excellent or poor ones).

Outlier detection algorithms could be used to detect the few excellent or poor wines. Also, we are not sure if all input variables are relevant. So it could be interesting to test feature selection methods.

Samples size: 1599 observations

Description of attributes:

1. **Fixed acidity**: most acids involved with wine or fixed or nonvolatile (do not evaporate readily)
2. **Volatile acidity**: the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste
3. **Citric acid**: found in small quantities, citric acid can add 'freshness' and flavor to wines
4. **Residual sugar**: the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet
5. **Chlorides**: the amount of salt in the wine
6. **Free sulfur dioxide**: the free form of SO_2 exists in equilibrium between molecular SO_2 (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine
7. **Total sulfur dioxide**: amount of free and bound forms of S02; in low concentrations, SO2 is mostly undetectable in wine, but at free SO2 concentrations over 50 ppm, SO_2 becomes evident in the nose and taste of wine
8. **Density**: the density of water is close to that of water depending on the percent alcohol and sugar content
9. **pH**: describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale
10. **Sulphates**: a wine additive which can contribute to sulfur dioxide gas ($S0_2$) levels, wich acts as an antimicrobial and antioxidant
11. **Alcohol**: the percent alcohol content of the wine
12. **Quality**: output variable (based on sensory data, score between 0 and 10)

II. Objective

As the requirements of the project, I follow 5 steps of the KDD process which will be demonstrated range from choosing dataset to evaluation models. In this project, the problem that I chose to solve is classification; thus, the Red Wine dataset will be seperated into binary classes .Each step will be describe in details

1. **Data preprocessing**
    - Change the problem from regression to classification
    - Data Splitting
    - Correlation matrix (reduce dimensionals of features)
    - Normalize data (change the domain of features)
    - Resampling data
2. **Model Selection**
    - Naive Bayes
    - K-Nearest Neighbors
    - Random Forests
    - Tunning Parameters (only Random Forests)
3. **Evaluation**: Compare the accuracy of 3 chosen models
4. **Conclusion**: Draw the subjective conclusion about the characteristics of the data, and also verify what important features which affect the result of the models.

III. The experiment

1. Load required packages

In this project, "caret" package is used as a main library which means that many functions and Machine Learning models are invoked through all the steps. Beside, I also include some graphical libraries for the use of demonstration figures and plotting

```
In [1]: library(caret)          # classification and regression training
        library(corrplot)       # graphical display of the correlation matrix
        library(class)          # K-nearest neighbors
        library(klaR)           # naive bayes
        library(randomForest)   # Random Forests
        library(gridExtra)      # save dataframes as images
        library(pROC)
        library(reshape2)
        library(ggplot2)
        today <- as.character(Sys.Date())
```

2. Load dataset

```
In [2]: wine <- read.csv('winequality-red.csv',sep = ',')
        head(wine[1:11],5)
        wine$quality <- as.integer(wine$quality)
```

Out[2]:

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide
1	7.4	0.7	0	1.9	0.076	11	34
2	7.8	0.88	0	2.6	0.098	25	67
3	7.8	0.76	0.04	2.3	0.092	15	54
4	11.2	0.28	0.56	1.9	0.075	17	60
5	7.4	0.7	0	1.9	0.076	11	34

3. Data Preprocessing

a. Change the problem from regression to classification

As the dataset description, the quality attribute is marked by the experts on the scale of 10 which is mumerical; therefore, this attribute is appropriate for regression purposes. However, I changed the characteristic of the feature to nominal. The method is grouping all instances with the quality are greater than or equal 6 to the "good" class; the others, which means less than, "bad" class. Thus, the new modified dataset is ready for classification tasks with 2 classes "good" and "bad"

```
In [3]: #Change the problem from regression to classification
        good <- wine$quality >= 6
        bad <- wine$quality < 6
        wine[good, 'quality'] <- 'good'
        wine[bad, 'quality'] <- 'bad'
        wine$quality <- as.factor(wine$quality) # redefine the factor variable

        dummies <- dummyVars(quality ~ ., data = wine)
        wine_dummied <- data.frame(predict(dummies, newdata = wine))
        wine_dummied[, 'quality'] <- wine$quality

        head(wine$quality, 5) # now, the quality feature is nominal
```

Out[3]:

1. bad

2. bad
3. bad
4. good
5. bad

b. Data Splitting

I use the cross validation method to split the data into train set and test set with the proportion is 0.7 for train and 0.3 for test

```
In [4]: set.seed(1234)
        trainIndex <- createDataPartition(wine$quality, p = .7,list = FALSE,times = 1)
        wineTrain <- wine_dummied[ trainIndex,]
        wineTest  <- wine_dummied[-trainIndex,]
```

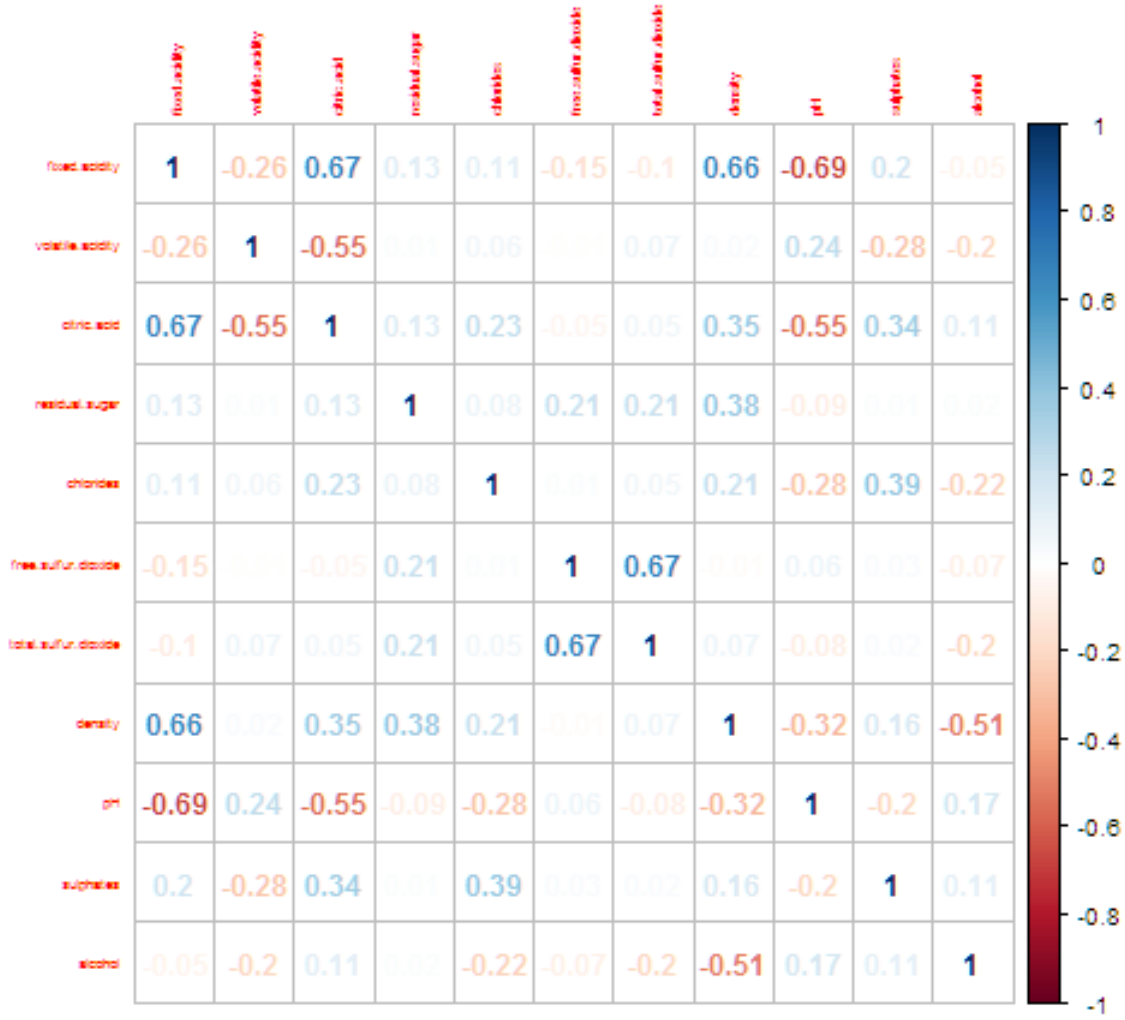
c. Correlation matrix

The purpose of finding the corellation matrix of the dataset is to get rid of the linearity in the data, which means reducing numbers of features are highly correlated, so that leads to the decreasing of the complexity when we apply machine learning models to the dataset.

```
In [5]: #Corelation matrix
        numericCol <- !colnames(wineTrain) %in% c('quality')
        correlMatrix <- cor(wineTrain[, numericCol])
        highlyCorrelated <- findCorrelation(correlMatrix, cutoff = 0.6)
        #highly correlated if threshold > 0.6
        colnames(correlMatrix)[highlyCorrelated]

        png(paste0(today, '-', 'correlation-matrix of 11 features.png'))
        corrplot(correlMatrix, method = 'number', tl.cex = 0.5)
        dev.off()
```

Out[5]:

1. "citric.acid"
2. "fixed.acidity"
3. "total.sulfur.dioxide"



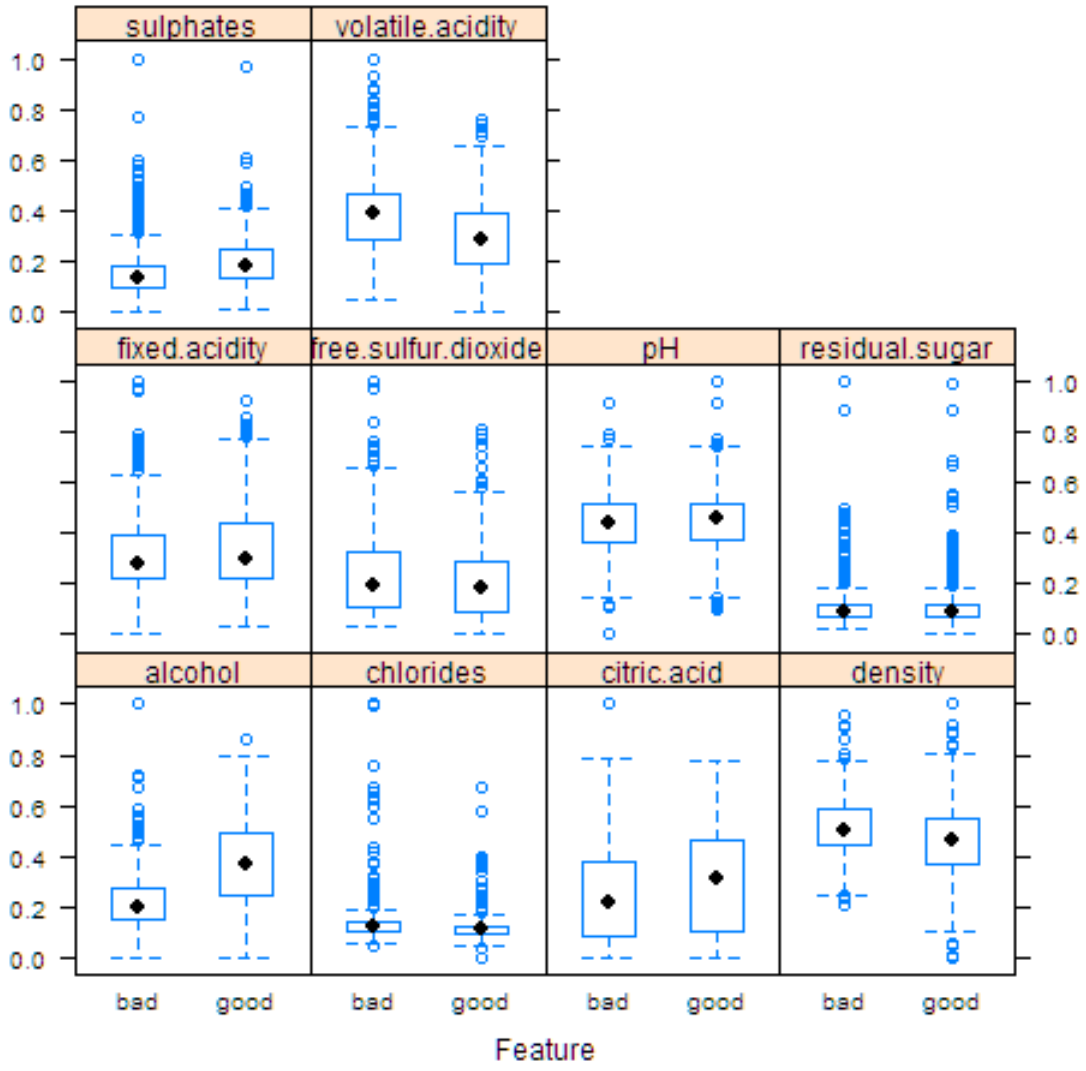
As the matrix has shown us, it is obvious to see that total.sulfur.dioxide has linearly correlated with other variables; therefore, I am not going not include this feature in any classifier.

```
In [6]: #Remove total.sulfur.dioxide from data.frame
        wineTrain <- wineTrain[ , -which(names(wineTrain) %in% c("total.sulfur.dioxide"))]
        wineTest <- wineTest[ , -which(names(wineTest) %in% c("total.sulfur.dioxide"))]
```

d. Normalize data

In the "caret" package, I use the function preProcess to normalize all the features in the dataset (from the 1^{st} column to the 10^{th} column) with the method "range"

```
In [7]: train_normalized <- preProcess(wineTrain[, 1:10], method = 'range')
        train_plot <- predict(train_normalized, wineTrain[, 1:10])
        png(paste0(today, '-', 'feature-plot.png'))
        featurePlot(train_plot, wineTrain$quality, 'box')
        dev.off()
```



From the figure, it looks like alcohol, citric.acid and density separate most with regard to good classification. Then these 3 features will be included in Machine Learning models.

e. Resampling data

The function trainControl in the caret package is a very comfortable way to set the resampling data we need. Among the resampling methods which accepts as argument, there are an extensive of choices such as bootstrap, cross validation, and repeated cross validation... In this project, I would like to demonstrate the resampling data's process by using cross validation method with the number of folds is 10.

```
In [8]: fitControl <- trainControl(method = 'cv', number = 10)
```

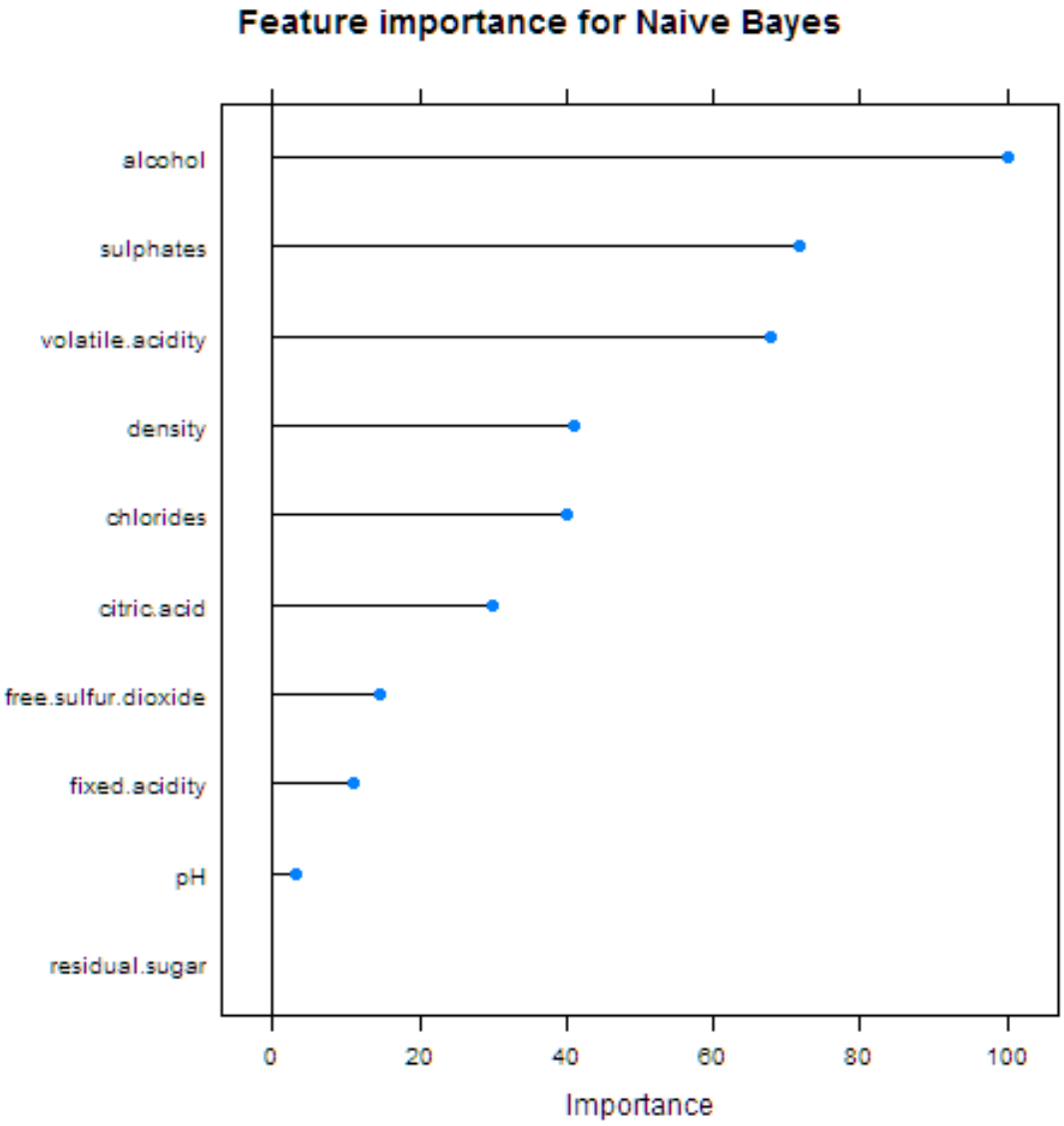
4. Model Selection

In this section, I would like to apply 3 supervised learning models which I researched. There are Naive Bayes, K-Nearest Neighbor and Random Forests. Through every models, I present the way how we input parameters of the train data into a model in "caret" library. Then, the next step is predicting on the test dataset, which based on the result of train data. Afterall, the confusion matrix is constructed to compute the accuracy of the chosen model. Furthermore, I discover that "caret" library has provided us a function to calculate the level of importance of the attributes; therefore, I also include this function in the script and visualize it for better evaluation what important features of the model.

a. Naive Bayes

In [9]:
```
##############################################
###############NAIVEBAYES####################
##############################################
fit_nb <- train(x = wineTrain[, 1:10], y = wineTrain$quality,
                method ='nb',trControl = fitControl)
predict_nb <- predict(fit_nb, newdata = wineTest[, 1:10])
confMat_nb <- confusionMatrix(predict_nb, wineTest$quality, positive = 'good')
importance_nb <- varImp(fit_nb, scale = TRUE)

confMat_nb

png(paste0(today, '-', 'importance-nb.png'))
plot(importance_nb, main = 'Feature importance for Naive Bayes')
dev.off()
```

Out[9]:
```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad  171   75
      good  52  181

               Accuracy : 0.7349
                 95% CI : (0.6929, 0.7739)
    No Information Rate : 0.5344
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.4707
 Mcnemar's Test P-Value : 0.05092

            Sensitivity : 0.7070
            Specificity : 0.7668
         Pos Pred Value : 0.7768
         Neg Pred Value : 0.6951
             Prevalence : 0.5344
         Detection Rate : 0.3779
   Detection Prevalence : 0.4864
      Balanced Accuracy : 0.7369

       'Positive' Class : good
```

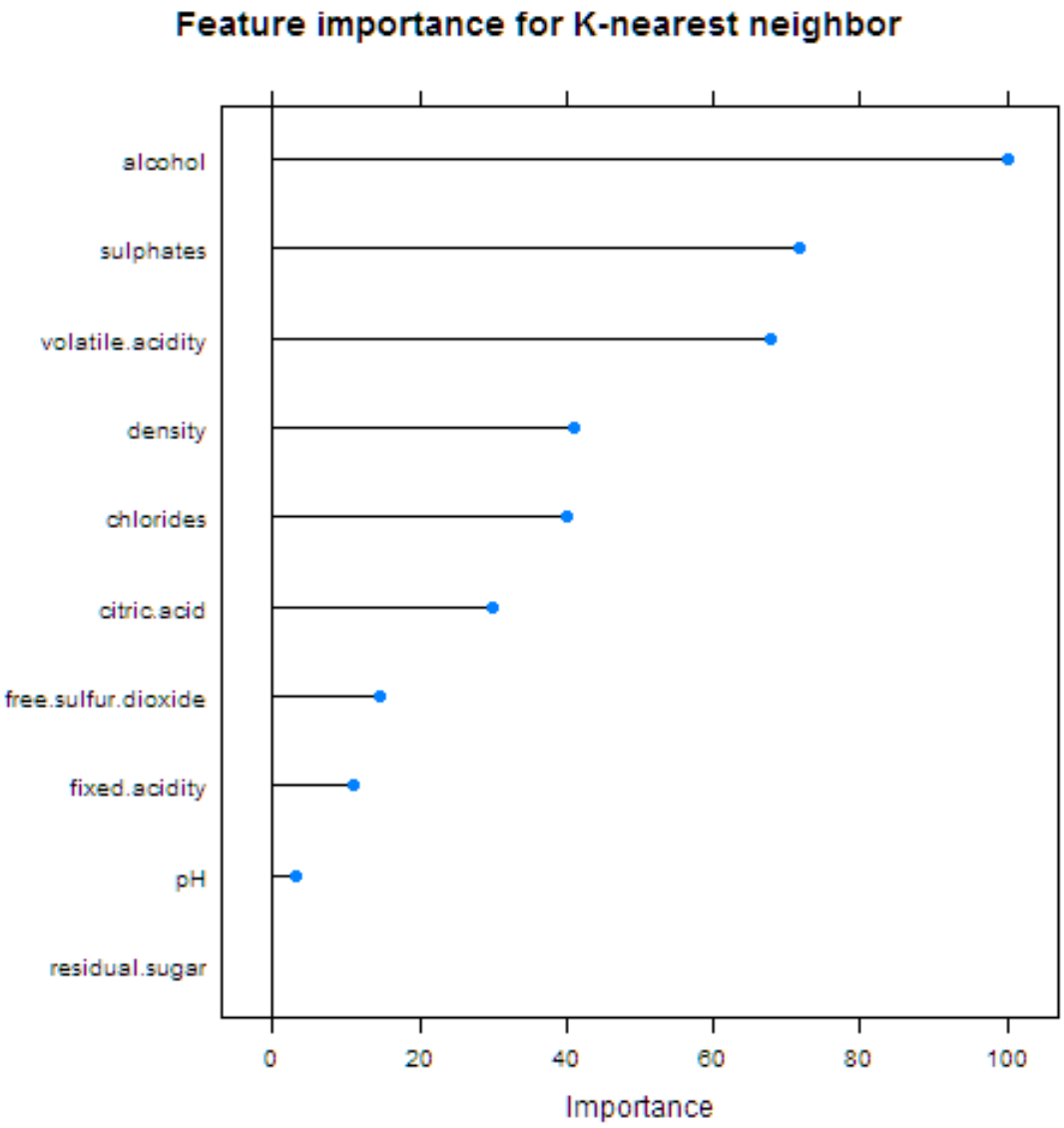


b. K-Nearest Neighbor

```
In [10]: ################################################
         ######################KNN#######################
         ################################################
         fit_knn <- train(x = wineTrain[, 1:10], y = wineTrain$quality,
                          method = 'knn',
                          preProcess = 'range',
                          trControl = fitControl,
                          tuneGrid = expand.grid(.k =
                                   c(3, 5, 7, 9, 11, 15, 21, 25, 31, 41, 51, 75, 101)))
         predict_knn <- predict(fit_knn, newdata = wineTest[, 1:10])
         confMat_knn <- confusionMatrix(predict_knn, wineTest$quality, positive = 'good')
         confMat_knn

         importance_knn <- varImp(fit_knn, scale = TRUE)

         png(paste0(today, '-', 'importance-knn.png'))
```

```
plot(importance_knn, main = 'Feature importance for K-nearest neighbor')
dev.off()
```

Out[10]:
```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad  158   71
      good  65  185

               Accuracy : 0.7161
                 95% CI : (0.6734, 0.7561)
    No Information Rate : 0.5344
    P-Value [Acc > NIR] : 3.124e-16

                  Kappa : 0.4304
 Mcnemar's Test P-Value : 0.6681

            Sensitivity : 0.7227
            Specificity : 0.7085
         Pos Pred Value : 0.7400
         Neg Pred Value : 0.6900
             Prevalence : 0.5344
         Detection Rate : 0.3862
   Detection Prevalence : 0.5219
      Balanced Accuracy : 0.7156

       'Positive' Class : good
```

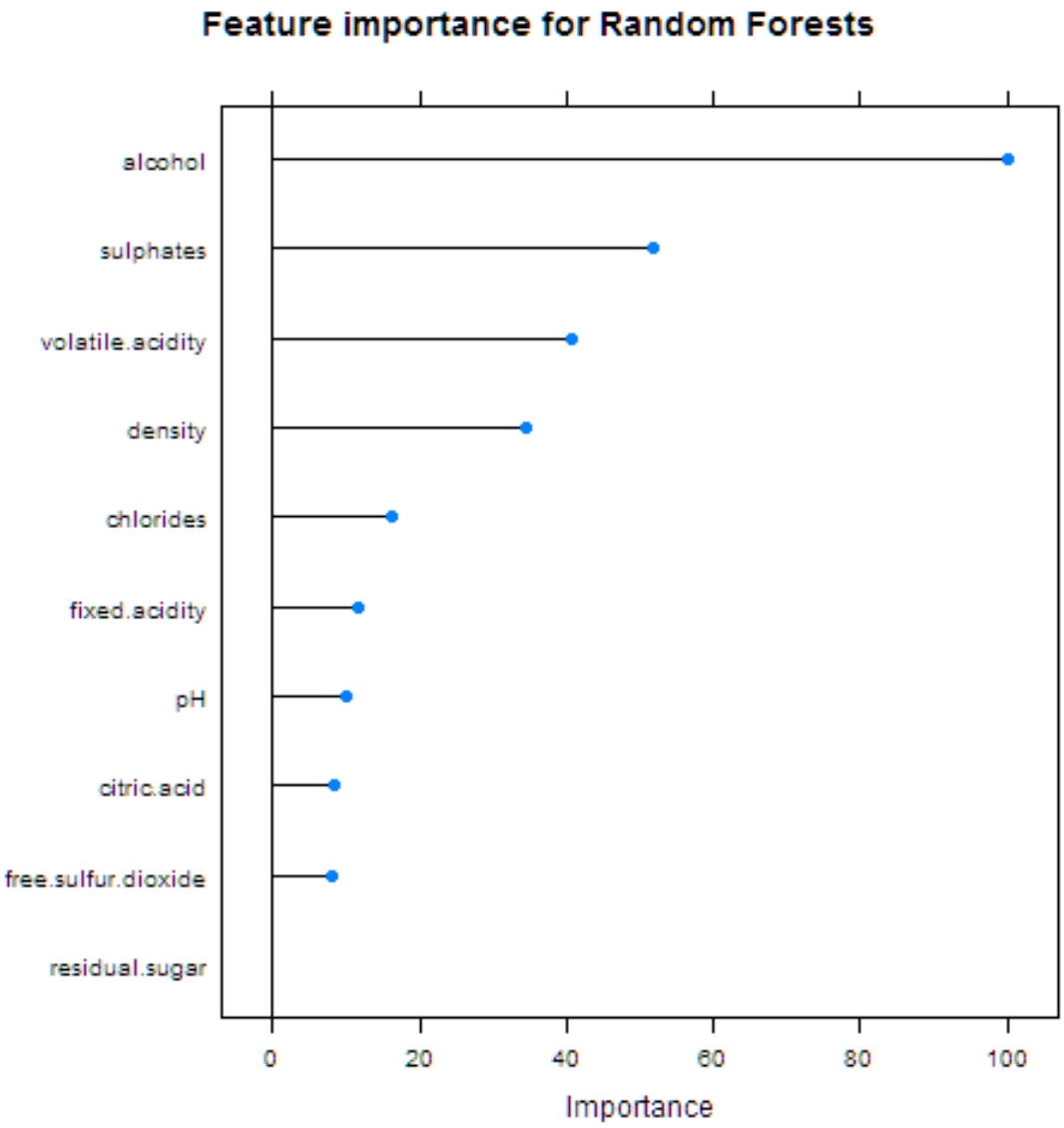


c. Random Forests

```
In [11]: ################################################
         ##############RANDOMFORESTS#####################
         ################################################

         fit_rf <- train(x = wineTrain[, 1:10], y = wineTrain$quality,
                         method = 'rf',
                         trControl = fitControl,
                         tuneGrid = expand.grid(.mtry = c(2:6)),
                         n.tree = 500)
         predict_rf <- predict(fit_rf, newdata = wineTest[, 1:10])
         confMat_rf <- confusionMatrix(predict_rf, wineTest$quality, positive = 'good')

         confMat_rf

         importance_rf <- varImp(fit_rf, scale = TRUE)
```

```
png(paste0(today, '-', 'importance-rf.png'))
plot(importance_rf, main = 'Feature importance for Random Forests')
dev.off()
```

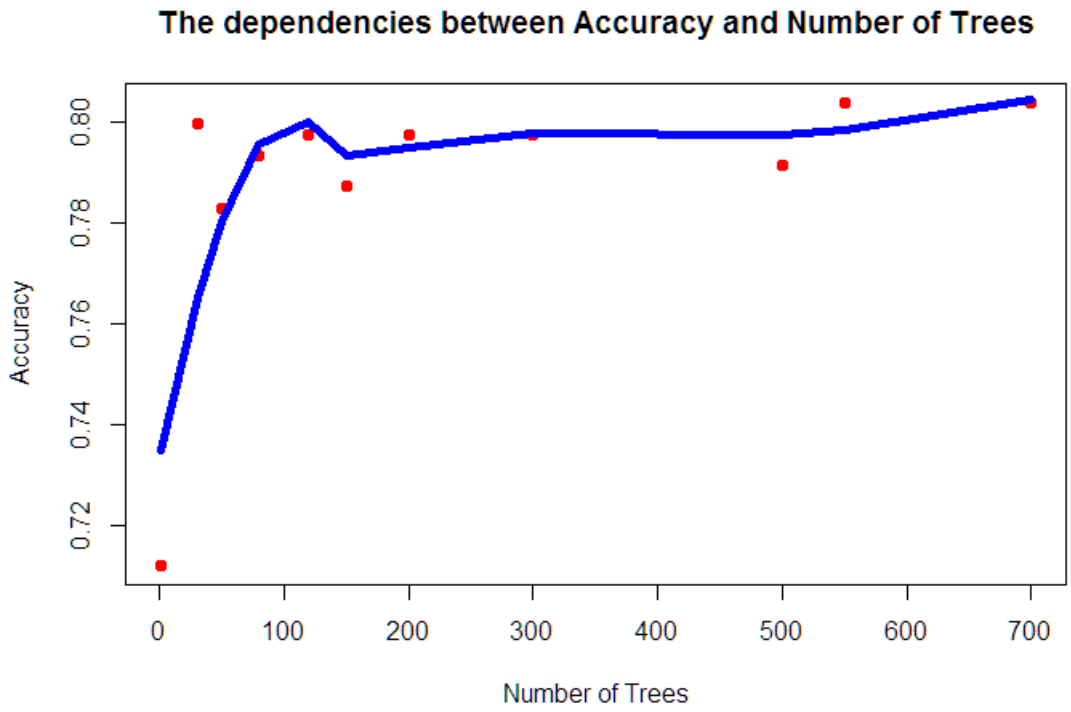
Out[11]:
```
Confusion Matrix and Statistics

          Reference
Prediction bad good
      bad  172   49
      good  51  207

               Accuracy : 0.7912
                 95% CI : (0.752, 0.8268)
    No Information Rate : 0.5344
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5802
 Mcnemar's Test P-Value : 0.9203

            Sensitivity : 0.8086
            Specificity : 0.7713
         Pos Pred Value : 0.8023
         Neg Pred Value : 0.7783
             Prevalence : 0.5344
         Detection Rate : 0.4322
   Detection Prevalence : 0.5386
      Balanced Accuracy : 0.7899

       'Positive' Class : good
```



d. Tunning parameters

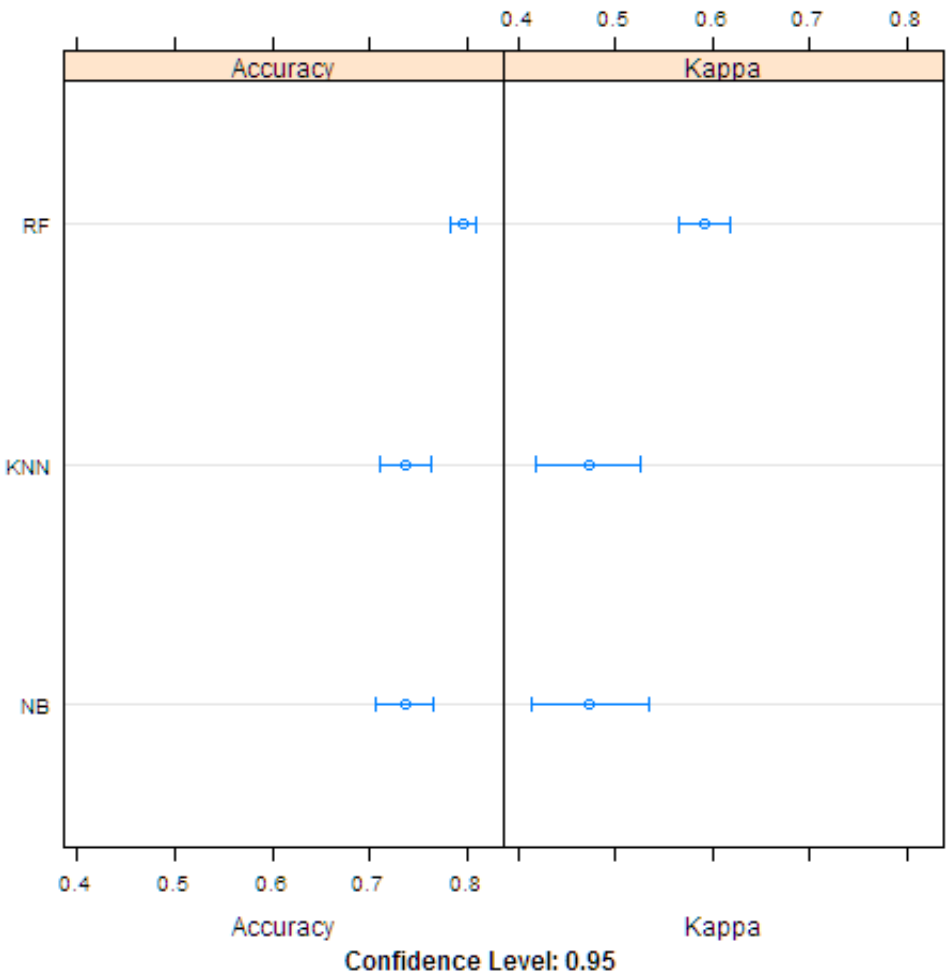
Due to the lack of resources, I just try to tunning parameters in Random Forests. Thus, I choose the number of trees in the algorithm for the purpose to evaluate what it affect the accuracy of the model.

```
In [12]: ntree <- c(1, 30, 50, 80, 120, 150, 200, 300, 500, 550, 700) #Vector number of trees
```



5. Evaluate the accuracy of the three chosen model

```
In [13]: models <- resamples(list(NB = fit_nb, KNN = fit_knn,
                                  RF = fit_rf))
         png(paste0(today, '-', 'models-comparison.png'))
         dotplot(models)
         dev.off()
```



In [14]:
```
results <- summary(models)
png(paste0(today, '-', 'models-accuracy.png'), width = 480, height = 180)
grid.table(results$statistics$Accuracy)
dev.off()
```

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
NB	0.6786	0.7165	0.7277	0.7357	0.7522	0.8125	0
KNN	0.6696	0.7277	0.75	0.7375	0.7646	0.7679	0
RF	0.7679	0.779	0.8	0.7964	0.8121	0.8214	0

IV. Conclusion

It does not look like wine quality is well supported by its chemical properties (We can easily see that in feature important of 3 chosen models). At each quality level variability of the predictors is high and the groups are not well separated.

The total.sulfur.dioxide is highly correlated and should be excluded from any classifier.

The acohol attribute strongly affects wines belonged to which class.

Between 3 chosen models, Random Forests gives us the best result.