White Wine Quality Evaluation and Prediction

December 8, 2014
Columbia University in the City of New York

Group Members:

Yuying Chen	yc2912
Qing Dong	qd2123
Yi Jiang	yj2306
Wenxin Liang	wl2455
Mengtian Song	ms4784

Content

Abstract

This article deals with the design and creation of models for white wine quality evaluation and prediction. A large dataset is considered, with white wine *vinho verde* sample, which from Portugal. Regression models are built to predict wine quality based on physicochemical tests, and classification models are used to classify wine quality into several categories according to its score. The Random Forest model achieves the most promising result. Apart from these models, other kinds of models may be used to give new relationships between wine quality and the predictors.

Key words: Regression, Random forests, Classification models, Ridge, LASSO

1.Introduction

As one kind of luxury goods, white wine is increasingly enjoyed by a wide variety of consumers. To safeguard human health and to develop the technologies of wine making, white wine quality assessments and prediction become more and more important.

Wine certification is generally assessed by physicochemical and sensory test (Appalasamy *et al*., 2012). Based on physicochemical tests, the predictors for white wine include its density, alcohol or pH values. Meanwhile, human experts perform sensory tests such as taste preference (Cortez *et al*., 2009). There are three regression techniques, which are the Support Vector Machine (SVM), multiple regression, and neural networks from machine learning using by the Cortez and his teammates.

We analyze the regression relationships between eleven predictors based on physicochemical tests and wine quality. The physicochemical test use to characterize white wine include *fixed acidity, volatile acidity, citric acid*, *residual sugar*, *chlorides*, *free sulfur dioxide*, *total sulfur dioxide*, *density*, *pH*, *sulphates* and *alcohol.* Each sample in this dataset is first evaluated by a minimum of three sensory assessors (using blind tastes) who grade the wine according to a scale that ranges from 0 (very bad) to 10 (excellent) (Appalasamy *et al*., 2012).

For the data we use to assess the quality of white wine, the data is continuous, and it is from Wine Quality Data Set on the UCI Machine Learning Repository website. Moreover, it is collected from May, 2004 to February, 2007 using only protected designation of origin samples that were tested at the official certification entity.

Our main focus in analyzing the white wine data set is to find out which predictors have more influence on determines the quality of white wine and to find out the capable models of prediction quality of white wine.

2.Materials

For creation the models, the dataset we use included 11 input variables (based on physicochemical tests) and 1 output variable *Quality* (based on sensory data). The total number of observations is 4898. The *Quality* of white wines is evaluated using a scale from 0 to 10 by integers and all the input variables are numeric, which showed in Table1 in appendix.

2.1 Input Data Summary

We summarize the twelve variables in Table 2. Based on Table 2, we conclude that the minimum value of the output variable, *Quality*, is 3 and the maximum value of *Quality* is 9 instead of 0(very bad) and 10(excellent). The mean is greater than the median for all the input variables and Range is much larger compared to the IQR. This phenomenon indicates that there are outliers existing in the data set.

In addition, Figure 1 shows that *alcohol* and *volatile acidity* have a linear relationship with *Quality*, and the values of wine *Quality* is almost a constant with respect to *free sulfur dioxide*, *pH*, and *residual sugar*, which means that the *Quality* has nothing to do with these factors.

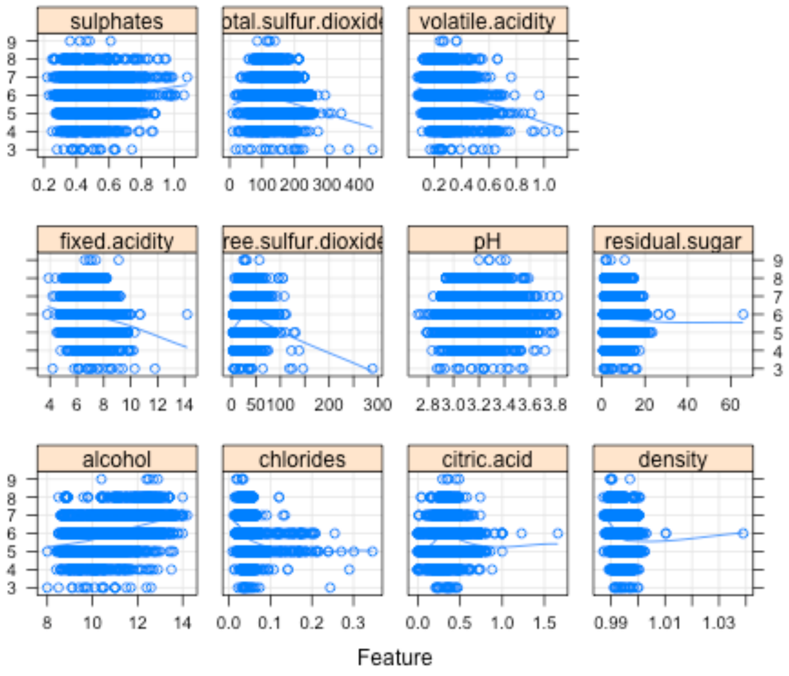


Figure 1 Scatter plots of each predictor versus the wine *Quality*

2.2 Exploratory Data Analysis and Data Pre-processing

For missing value checking, based on Table3 in appendix, which describes the twelve variables in summary, there are no missing values for all the variables.

For outlier checking, the box plots, showed in figure 2, indicates that all of the inputs variables except *alcohol* have outliers. *Residual sugar* has a positively skewed distribution and even after eliminating the outliers distribution will still remain skewed. Also we notice that mostly outliers are on the larger side.

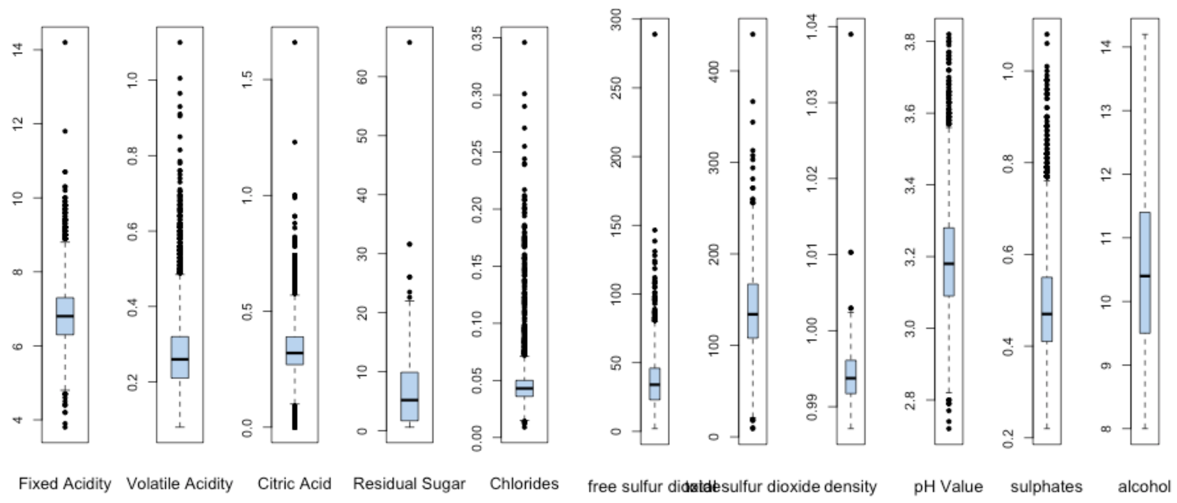


Figure 2. The boxplots of all input variables

For correlation checking, since the variables have non-normal distribution, we consider person correlations, which in Appendix Table 4, spearman rank correlations which in Appendix Table 5 and the correlation plot showed in figure 3 below. There are high correlations, which is greater than or equal to 40% in absolute value, existing. In addition based on figure 3, we observe that *density* and *residual sugar* has a strong positive relationship, *alcohol* and *density* has strong negative relationship.

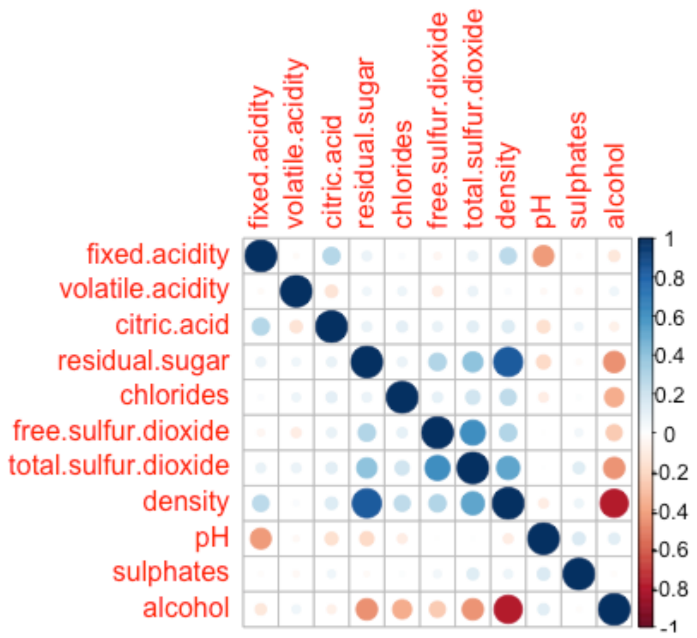


Figure 3. Correlation plot between predictors

2.3 Data Preparation

Since most of the input variables have large amounts of outliers and there are correlation existing, data in predictors need to be centered, scaled and transformed to prepare for model building. After transformation, new box plots shows that our data has a better distribution. Then, we split our data to training set and test set with proportion of 3:1. The training set is used to create the model and do the10-fold cross-validation to produce reasonable estimates for model building, and the test set is used to qualify performance.

3. Methodology and Models

To evaluate white wine quality based on our data set, a series of models are created and evaluated. The purpose of testing data in a variety of models is to select the best model to predict wine quality. In every model, tuning parameters (if had) are first estimated, and then are used to fit our dataset which after transformation. 10-fold cross-validation method also uses for resampling. Then, predictions are made. Final we obtain our result displayed by R^2 for prediction and error rate to test the fitness of the models.

3.1 Regressions Models

First, we use linear regression models, including multiple linear regression, Ridge, LASSO and polynomial regression. R^2 and error rate are generated for checking whether the model is appropriate for our white wine data set. R^2 for prediction and error rate value are summarized in Table 9 below.

Since multiple linear regression is the common model built to understand the association between the input variables and the output variable in the first place and it is easier to interpret so we use multiple linear regression as our first model to study the white wine quality. The result displays in the appendix Table 6. We find that the input variables *citric acid*, *chlorides* and *total sulfur dioxide* are not significant. The R^2 for the predictions is nearly 27%, which means the model does not match the condition of the data so well. The error rate for the predictions is nearly 48%, which means there is a lot of error happened for prediction of *Quality*.

Since the result of multiple regression shows that there are some predictors not significant, Ridge and LASSO are taken into consideration to do regularization. The result of Ridge displays in the appendix Table 7 and the result of LASSO marks in the appendix Table 8. From the result of Ridge, none of the predictor is dropped, and there is no improvement from multiple regressions. The R^2 for the predictions is nearly 27%, and the error rate for the predictions is nearly 48%.

In the result of Lasso, the input predictors *fixed acidity, citric acid, total sulfur dioxide* and *density* are dropped. These parameters dropped are the correlated predictors shown in Table 4, Table 5 and figure 3. The R^2 for the predictions is nearly 26%, and the error rate for the predictions is nearly 48%, which are still not good.

In order to investigate whether a polynomial relationship fits the model better, a polynomial regression model with squared terms of the significant variables is tried, which improves R^2 value to nearly 31%, and the error rate for the predictions is nearly 48%. Although polynomial regression model can explain more about our data set, it still cannot be the good model we expect.

Since the result of Multiple Regression is not good in prediction, Ridge and Lasso regression methods do not make much improvement and polynomial regression cannot explain more than 50% of the data then we can conclude that linear regression is not that appropriate for this data set.

	Multiple Linear Regression	Ridge	LASSO	Polynomial Regression
Error rate	0.4818449	0.4789009	0.484789	0.4759568
R^2	0.2737022	0.2784749	0.2625427	0.3066771

Table 9. R^2 and error rate for Linear Regression Models

Second, we used non-linear regression models, including Random Forest, neural networks (NN) and Support Vector Machines (SVM). R^2 for prediction and error rate are summarized in Table 10 below.

Since the Random Forest is one of the most accurate learning algorithms available, we try this model as our first non-linear regression model. Based on decision trees, the algorithm nature makes Random Forest suitable to regression and classification. We estimate the turning parameter with 10-fold cross-validation method, it turns out that the best turning parameter is 2. The error rate for random forest with regression prediction is almost 31.7% and R^2 for prediction we obtain is nearly 52%.

The Random Forest model also provides a method to measure the importance of predictors in prediction. There are two measures to measure the variable importance, mean decrease in accuracy and mean decrease in node impurity. For each tree, the prediction accuracy on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor variable. The difference between the two accuracies are then averaged over all trees, and normalized by the standard error. For regression, the MSE is computed on the out-of-bag data for each tree, and then the same compute after permuting a variable. The differences are averaged and normalized by the standard error. The second measure is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares. According to figure 4, in the mean decrease in accuracy measure, the input predictors *volatile acidity*, *alcohol* and *free sulfur dioxide* are the most important while *alcohol*, *density* and *free sulfur dioxide* are the most important predictors in mean decrease in node impurity measure. Therefore, we notice that the input predictors *alcohol*, *free sulfur dioxide* and *volatile acidity* have more influence to the white wine *Quality*.

Since Neural Networks are the powerful nonlinear regression techniques inspired by theories about how the brain works, we also use this model for white wine *Quality* analysis. The outcome is modeled by an intermediary set of unobserved variables called hidden units. These hidden units are linear combinations of the original predictors. However, neural networks have a tendency to over-fit the relationship between the predictors and the response due to the large number of regression coefficients. In this model, the final values used for the model are size = 3 and decay

= 0.1. For prediction, the values we used are size=3, decay=0.1, and 5 hidden units. In addition, the error rate for NN we obtain is nearly 50% and the R^2 value for prediction is 20%.

For support vector machine (SVM), since it is less sensitive to outliers, it is better than multiple linear regression. In addition, SVM is characterized by usage of kernels, absence of local minima, and capacity control obtained by acting on the margin, or on number of support vectors. The variables are transformed into a high m-dimensional feature space, by using a nonlinear mapping that does not need to be explicitly known but that depends of a kernel function. The result we got includes nearly 47% error rate and approximately 30% R^2 value for prediction.

	Random Forest	NN	SVM
Error rate	0.3169935	0.4996101	0.4698320
R^2	0.5221166	0.200876	0.2956326

Table 10. R^2 and error rate for Non-linear Regression Models

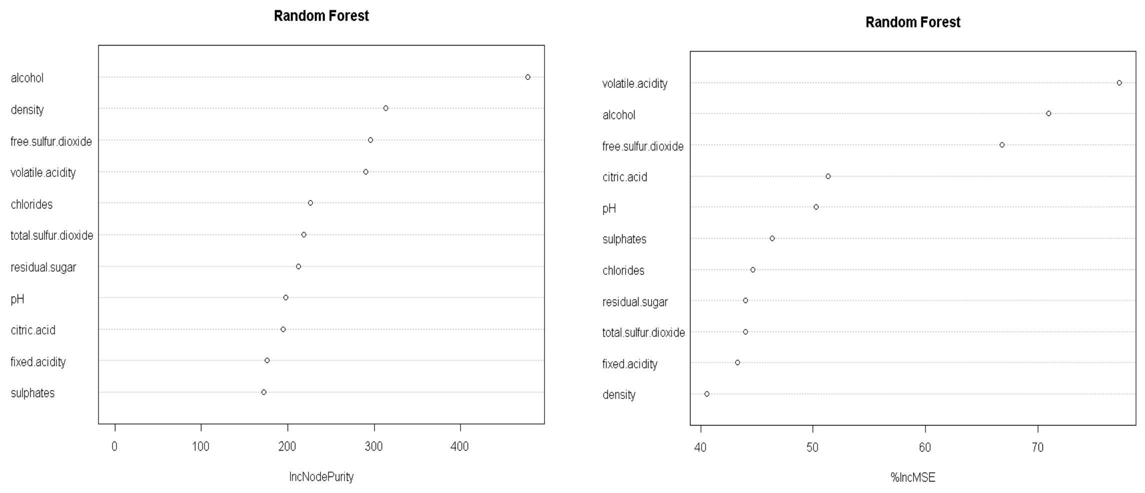


Figure 4. The importance of predictors in prediction based on Random Forest

3.2 Classification Models

Moreover, we built classification models to classify white wine *Quality* into 7 categories, from 3 to 9. We use Decision Tree, Random Forest, Support Vector machine (SVM) and Adaboost to classify. In each model, an error rate is generated. The error rate for each model summarize in Table 11 below.

Since the problem can be viewed as a multi-class classification problem, it is very natural to use tree-based method. The very first model using is just the decision tree. Decision tree is a supervised machine learning method. Although it is a greedy algorithm, it can still sometimes show off a very good result. A key advantage of tree-based method is that it is very easy to understand and it is very easy to interpret the result. In every step, the tree will choose both the feature and the cut point that could increase most the purity of notes. And the result is given by the most majority of predicts in this leaf. With this method, the tree will be trained to put the similar wines with similar wine score together and therefore useful in classification. For the white wine evaluation, we get the error rate of 48.65%. It is not so good so we used Random Forest.

The decision tree method only train one tree, but the ensemble method such as Random Forest which combined a lot of trees may get a better result. Random Forest is an algorithm, which is unexcelled in accuracy among current algorithms. Compare to the other classification methods we used for our white wine evaluation dataset, Random Forest performs very well. The error rate is just 31% for this multi-class classification problem, which is the best one among all the algorithms we have tried.

For Adaboost model, it is also an effective machine-learning algorithm that can be used to classification. The algorithm gives every observation a weight to measure how important it is in the classification. The higher the error rate is, the higher the weight of this observation is, so that the classifier will pay more attention to classify the observations that are easy to get wrong result. The result of Adaboost in this white wine quality dataset, which indicated by the error rate, is nearly 48%.

Basically support vector machine (SVM) is typically an algorithm to solve 0-1 classification problems. However, with the method of “one-against-one” or “one-against-all” method, we can also apply SVM into the multi-class classification problem. Here we take the radial bas function (RBF) kernel to solve the problem and the result is showed below.

	Decision Tree	Random Forest	Adaboost	SVM
Error rate	0.4865702	0.3109504	0.4690873	0.3167331

Table 11. Error rate for Classification Models

4. Results

Compare Table 9 and Table 10, for regression models we can conclude that except Random Forest model, the other regression models are not good fits because the values of R^2 for prediction are all below 0.5, which means all the regression models cannot explain half of the white wine data.

Based on Table 11, for classification models, Random Forest have the best performance compare to other classification models. Support Vector Machine also has a good performance.

Compare the result showed in Table 9, Table 10 and Table 11, the regression models and the classification models we observe that Random Forest model do the best job compare to all the other models. It works well in variance reduction and noise robustness. Also Random Forest has several advantages. Random Forest is one of the most accurate learning algorithms available. Random Forest gets its samples from the Bootstrap sample, which the variance between different trees is very large, and it is very good for the ensemble algorithms. Also, Random Forest runs efficiently on large databases and gives estimates of what variables are important in the classification. It can handle thousands of input variables without variable deletion. Moreover, Random Forest is able to generate forests, which can be saved for future use. Therefore, random forest model provides more stable predictions and model accuracy.

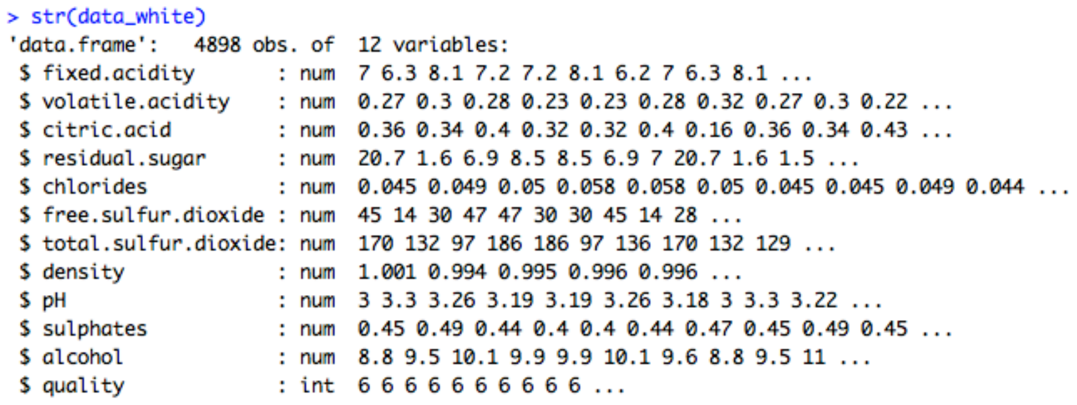
5. Conclusions and Implications

Based on figure 4, *alcohol*, *free sulfur dioxide* and *volatile acidity* have more influence on white wine *Quality* than other predictors.

Compare all the regression models built, the values of R^2 for prediction for the total 7 regression models are not big enough, even Random Forest can only explain 52% of the data set, then regression models cannot fit the data very well. Compare all the classification models built, the SVM model performs much better than the other model except Random Forest. Random Forest is best model we have to cover all the conditions of our data. Finally compare all the models used, the error rate for random forest is the lowest, Random Forest can be a more capable model for predicting *Quality* of the white wine.

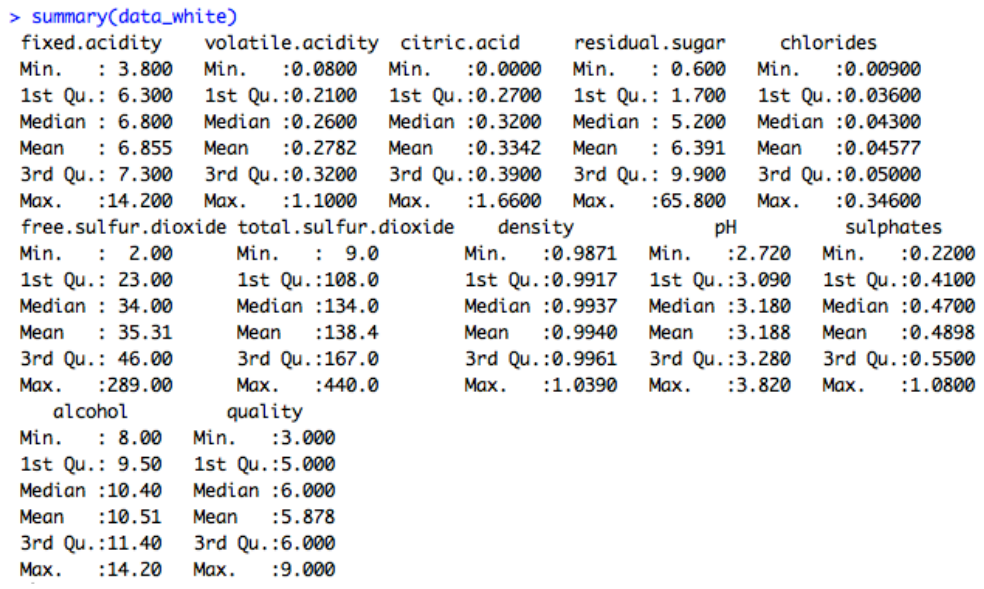
In this data set, the values of the response variable are integers mainly from 3 to 9, instead of all the real numbers in this range, so the regression lines between the outcome and predictors are not perfect. In this sense, more adjustments should be applied to get better fits, and other models should be used to fit the data to gain a relatively perfect prediction.

Appendix



```
> str(data_white)
'data.frame':   4898 obs. of  12 variables:
 $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
 $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.22 ...
 $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.43 ...
 $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
 $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045 0.049 0.044 ...
 $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
 $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
 $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
 $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
 $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.45 ...
 $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
 $ quality             : int  6 6 6 6 6 6 6 6 6 6 ...
```

Table1. The structure of white wine data set



```
> summary(data_white)
 fixed.acidity    volatile.acidity  citric.acid     residual.sugar     chlorides      
 Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600   Min.   :0.00900  
 1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700   1st Qu.:0.03600  
 Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200   Median :0.04300  
 Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391   Mean   :0.04577  
 3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900   3rd Qu.:0.05000  
 Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800   Max.   :0.34600  
 free.sulfur.dioxide total.sulfur.dioxide    density             pH          sulphates     
 Min.   :  2.00      Min.   :  9.0        Min.   :0.9871   Min.   :2.720   Min.   :0.2200  
 1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917   1st Qu.:3.090   1st Qu.:0.4100  
 Median : 34.00      Median :134.0        Median :0.9937   Median :3.180   Median :0.4700  
 Mean   : 35.31      Mean   :138.4        Mean   :0.9940   Mean   :3.188   Mean   :0.4898  
 3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961   3rd Qu.:3.280   3rd Qu.:0.5500  
 Max.   :289.00      Max.   :440.0        Max.   :1.0390   Max.   :3.820   Max.   :1.0800  
    alcohol          quality     
 Min.   : 8.00   Min.   :3.000  
 1st Qu.: 9.50   1st Qu.:5.000  
 Median :10.40   Median :6.000  
 Mean   :10.51   Mean   :5.878  
 3rd Qu.:11.40   3rd Qu.:6.000  
 Max.   :14.20   Max.   :9.000  
```

Table 2. The summary of white wine data set

```
> library(Hmisc)
> describe(data_white,digit=5)
data_white

 12  Variables      4898  Observations
-------------------------------------------------------------------------------
fixed.acidity
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0      68  6.8548     5.6     5.9     6.3     6.8     7.3     7.9     8.3

lowest :  3.8  3.9  4.2  4.4  4.5, highest: 10.2 10.3 10.7 11.8 14.2
-------------------------------------------------------------------------------
volatile.acidity
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     125 0.27824    0.15    0.17    0.21    0.26    0.32    0.40    0.46

lowest : 0.080 0.085 0.090 0.100 0.105, highest: 0.910 0.930 0.965 1.005 1.100
-------------------------------------------------------------------------------
citric.acid
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0      87 0.33419    0.17    0.22    0.27    0.32    0.39    0.49    0.54

lowest : 0.00 0.01 0.02 0.03 0.04, highest: 0.91 0.99 1.00 1.23 1.66
-------------------------------------------------------------------------------
residual.sugar
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     310  6.3914     1.1     1.2     1.7     5.2     9.9    14.0    15.7

lowest :  0.60  0.70  0.80  0.90  0.95, highest: 22.60 23.50 26.05 31.60 65.80
-------------------------------------------------------------------------------

chlorides
       n  missing   unique     Mean      .05      .10      .25      .50      .75      .90
    4898        0      160 0.045772    0.027    0.030    0.036    0.043    0.050    0.058
     .95
   0.067

lowest : 0.009 0.012 0.013 0.014 0.015, highest: 0.255 0.271 0.290 0.301 0.346
-------------------------------------------------------------------------------
free.sulfur.dioxide
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     132  35.308      11      15      23      34      46      57      63

lowest :   2.0   3.0   4.0   5.0   6.0, highest: 128.0 131.0 138.5 146.5 289.0
-------------------------------------------------------------------------------
total.sulfur.dioxide
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     251  138.36      75      87     108     134     167     195     212

lowest :   9.0  10.0  18.0  19.0  21.0, highest: 307.5 313.0 344.0 366.5 440.0
-------------------------------------------------------------------------------
density
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     890 0.99403 0.98964 0.99033 0.99172 0.99374 0.99610 0.99815 0.99900

lowest : 0.98711 0.98713 0.98722 0.98740 0.98742
highest: 1.00240 1.00241 1.00295 1.01030 1.03898
-------------------------------------------------------------------------------
pH
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     103  3.1883    2.96    3.00    3.09    3.18    3.28    3.38    3.46
```

```
lowest : 2.72 2.74 2.77 2.79 2.80, highest: 3.77 3.79 3.80 3.81 3.82
--------------------------------------------------------------------------------
sulphates
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0      79 0.48985    0.34    0.36    0.41    0.47    0.55    0.64    0.71

lowest : 0.22 0.23 0.25 0.26 0.27, highest: 0.99 1.00 1.01 1.06 1.08
--------------------------------------------------------------------------------
alcohol
      n missing  unique    Mean     .05     .10     .25     .50     .75     .90     .95
   4898       0     103  10.514     8.9     9.0     9.5    10.4    11.4    12.4    12.7

lowest :  8.00  8.40  8.50  8.60  8.70, highest: 13.80 13.90 14.00 14.05 14.20
--------------------------------------------------------------------------------
quality
      n missing  unique    Mean
   4898       0       7  5.8779

           3   4    5    6   7   8 9
Frequency 20 163 1457 2198 880 175 5
%          0   3   30   45  18   4 0
--------------------------------------------------------------------------------
```

Table 3. Describe the white wine data set

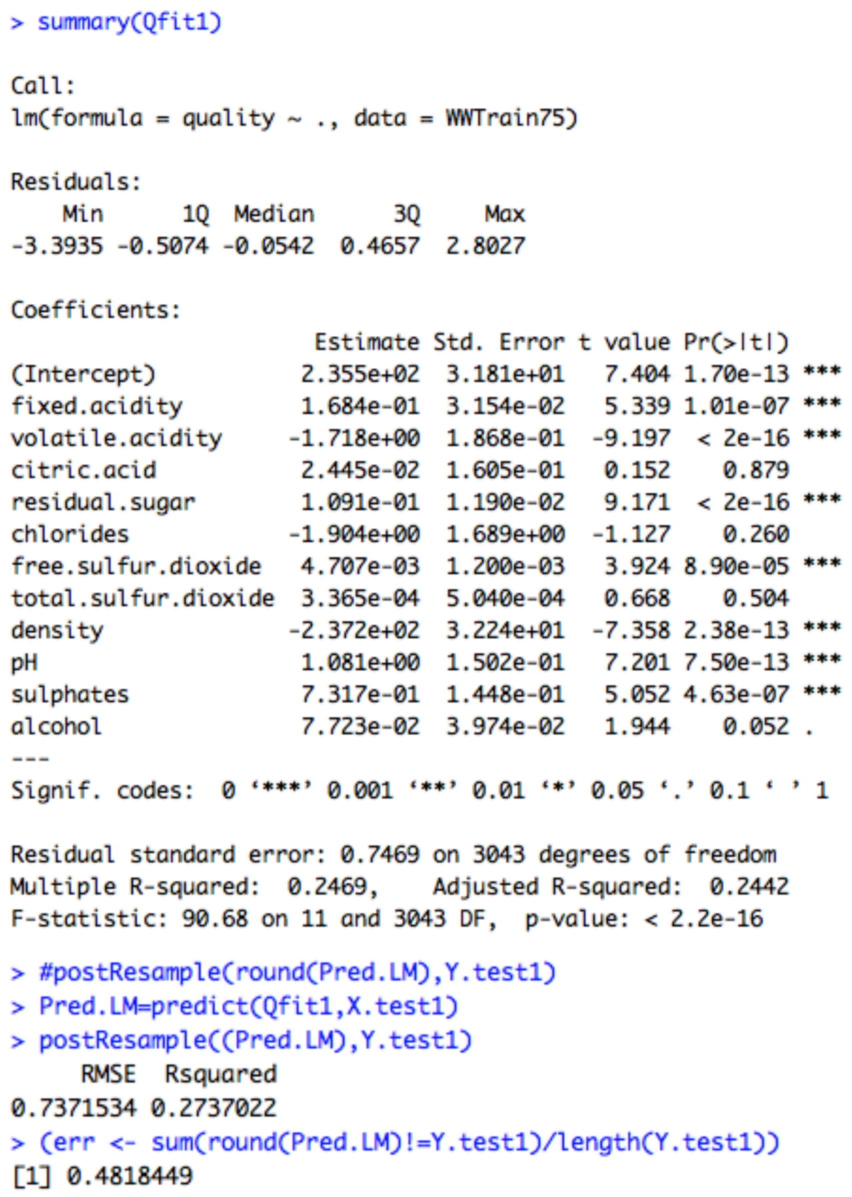
```
                     fixed.acidity volatile.acidity citric.acid residual.sugar
fixed.acidity           1.00000000      -0.02269729  0.28918070     0.08902070
volatile.acidity       -0.02269729       1.00000000 -0.14947181     0.06428606
citric.acid             0.28918070      -0.14947181  1.00000000     0.09421162
residual.sugar          0.08902070       0.06428606  0.09421162     1.00000000
chlorides               0.02308564       0.07051157  0.11436445     0.08868454
free.sulfur.dioxide    -0.04939586      -0.09701194  0.09407722     0.29909835
total.sulfur.dioxide    0.09106976       0.08926050  0.12113080     0.40143931
density                 0.26533101       0.02711385  0.14950257     0.83896645
pH                     -0.42585829      -0.03191537 -0.16374821    -0.19413345
sulphates              -0.01714299      -0.03572815  0.06233094    -0.02666437
alcohol                -0.12088112       0.06771794 -0.07572873    -0.45063122

                       chlorides free.sulfur.dioxide total.sulfur.dioxide     density
fixed.acidity         0.02308564       -0.0493958591          0.091069756  0.26533101
volatile.acidity      0.07051157       -0.0970119393          0.089260504  0.02711385
citric.acid           0.11436445        0.0940772210          0.121130798  0.14950257
residual.sugar        0.08868454        0.2990983537          0.401439311  0.83896645
chlorides             1.00000000        0.1013923521          0.198910300  0.25721132
free.sulfur.dioxide   0.10139235        1.0000000000          0.615500965  0.29421041
total.sulfur.dioxide  0.19891030        0.6155009650          1.000000000  0.52988132
density               0.25721132        0.2942104109          0.529881324  1.00000000
pH                   -0.09043946       -0.0006177961          0.002320972 -0.09359149
sulphates             0.01676288        0.0592172458          0.134562367  0.07449315
alcohol              -0.36018871       -0.2501039415         -0.448892102 -0.78013762
```

```
                               pH   sulphates     alcohol
fixed.acidity        -0.4258582910 -0.01714299 -0.12088112
volatile.acidity     -0.0319153683 -0.03572815  0.06771794
citric.acid          -0.1637482114  0.06233094 -0.07572873
residual.sugar       -0.1941334540 -0.02666437 -0.45063122
chlorides            -0.0904394560  0.01676288 -0.36018871
free.sulfur.dioxide  -0.0006177961  0.05921725 -0.25010394
total.sulfur.dioxide  0.0023209718  0.13456237 -0.44889210
density              -0.0935914935  0.07449315 -0.78013762
pH                    1.0000000000  0.15595150  0.12143210
sulphates             0.1559514973  1.00000000 -0.01743277
alcohol               0.1214320987 -0.01743277  1.00000000
```

Table 4. Pearson’s Correlation

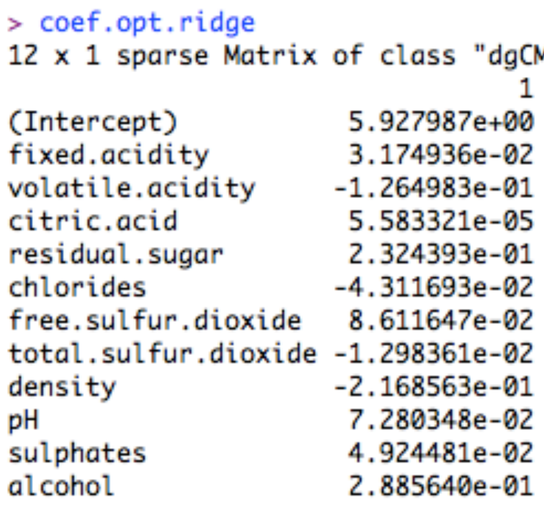
```
                     fixed.acidity volatile.acidity citric.acid residual.sugar
fixed.acidity           1.00000000     -0.042865228  0.29787793     0.10672494
volatile.acidity       -0.04286523      1.000000000 -0.15040998     0.10862744
citric.acid             0.29787793     -0.150409981  1.00000000     0.02462098
residual.sugar          0.10672494      0.108627441  0.02462098     1.00000000
chlorides               0.09469118     -0.004934156  0.03265950     0.22784390
free.sulfur.dioxide    -0.02454223     -0.081212903  0.08831406     0.34610674
total.sulfur.dioxide    0.11264866      0.117613959  0.09321867     0.43125248
density                 0.27003091      0.010124341  0.09142519     0.78036485
pH                     -0.41834116     -0.045203569 -0.14619267    -0.18002822
sulphates              -0.01323781     -0.016902303  0.07976628    -0.00384398
alcohol                -0.10682740      0.033966554 -0.02916996    -0.44525743
                       chlorides free.sulfur.dioxide total.sulfur.dioxide    density
fixed.acidity         0.094691176        -0.024542230           0.11264866 0.27003091
volatile.acidity     -0.004934156        -0.081212903           0.11761396 0.01012434
citric.acid           0.032659495         0.088314056           0.09321867 0.09142519
residual.sugar        0.227843904         0.346106737           0.43125248 0.78036485
chlorides             1.000000000         0.167045505           0.37524367 0.50830177
free.sulfur.dioxide   0.167045505         1.000000000           0.61861634 0.32782180
total.sulfur.dioxide  0.375243666         0.618616339           1.00000000 0.56382409
density               0.508301765         0.327821798           0.56382409 1.00000000
pH                   -0.054006467        -0.006273578          -0.01182872 -0.11006085
sulphates             0.093930696         0.052251683           0.15782480 0.09507867
alcohol              -0.570806407        -0.272569338          -0.47661933 -0.82185508
                              pH   sulphates     alcohol
fixed.acidity        -0.418341158 -0.01323781 -0.10682740
volatile.acidity     -0.045203569 -0.01690230  0.03396655
citric.acid          -0.146192675  0.07976628 -0.02916996
residual.sugar       -0.180028223 -0.00384398 -0.44525743
chlorides            -0.054006467  0.09393070 -0.57080641
free.sulfur.dioxide  -0.006273578  0.05225168 -0.27256934
total.sulfur.dioxide -0.011828718  0.15782480 -0.47661933
density              -0.110060852  0.09507867 -0.82185508
pH                    1.000000000  0.14024331  0.14885725
sulphates             0.140243305  1.00000000 -0.04486799
alcohol               0.148857249 -0.04486799  1.00000000
```

Table 5. Spearman Rank Correlation

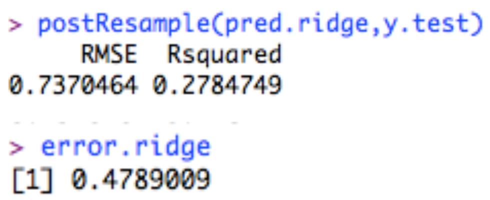


```
> summary(Qfit1)

Call:
lm(formula = quality ~ ., data = WWTrain75)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3935 -0.5074 -0.0542  0.4657  2.8027 

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)    
(Intercept)           2.355e+02  3.181e+01   7.404 1.70e-13 ***
fixed.acidity         1.684e-01  3.154e-02   5.339 1.01e-07 ***
volatile.acidity     -1.718e+00  1.868e-01  -9.197  < 2e-16 ***
citric.acid           2.445e-02  1.605e-01   0.152    0.879    
residual.sugar        1.091e-01  1.190e-02   9.171  < 2e-16 ***
chlorides            -1.904e+00  1.689e+00  -1.127    0.260    
free.sulfur.dioxide   4.707e-03  1.200e-03   3.924 8.90e-05 ***
total.sulfur.dioxide  3.365e-04  5.040e-04   0.668    0.504    
density              -2.372e+02  3.224e+01  -7.358 2.38e-13 ***
pH                    1.081e+00  1.502e-01   7.201 7.50e-13 ***
sulphates             7.317e-01  1.448e-01   5.052 4.63e-07 ***
alcohol               7.723e-02  3.974e-02   1.944    0.052 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7469 on 3043 degrees of freedom
Multiple R-squared:  0.2469,	Adjusted R-squared:  0.2442 
F-statistic: 90.68 on 11 and 3043 DF,  p-value: < 2.2e-16

> #postResample(round(Pred.LM),Y.test1)
> Pred.LM=predict(Qfit1,X.test1)
> postResample((Pred.LM),Y.test1)
     RMSE  Rsquared 
0.7371534 0.2737022 
> (err <- sum(round(Pred.LM)!=Y.test1)/length(Y.test1))
[1] 0.4818449
```

Table 6. The result of multiple linear regression

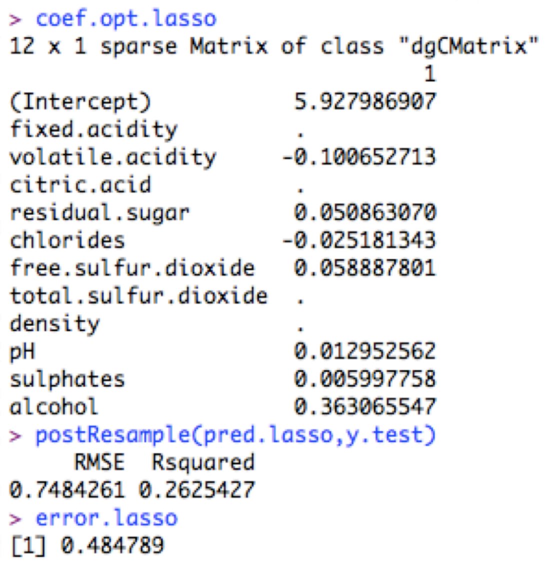


```
> coef.opt.ridge
12 x 1 sparse Matrix of class "dgCMatrix"
                                 1
(Intercept)           5.927987e+00
fixed.acidity         3.174936e-02
volatile.acidity     -1.264983e-01
citric.acid           5.583321e-05
residual.sugar        2.324393e-01
chlorides            -4.311693e-02
free.sulfur.dioxide   8.611647e-02
total.sulfur.dioxide -1.298361e-02
density              -2.168563e-01
pH                    7.280348e-02
sulphates             4.924481e-02
alcohol               2.885640e-01
```



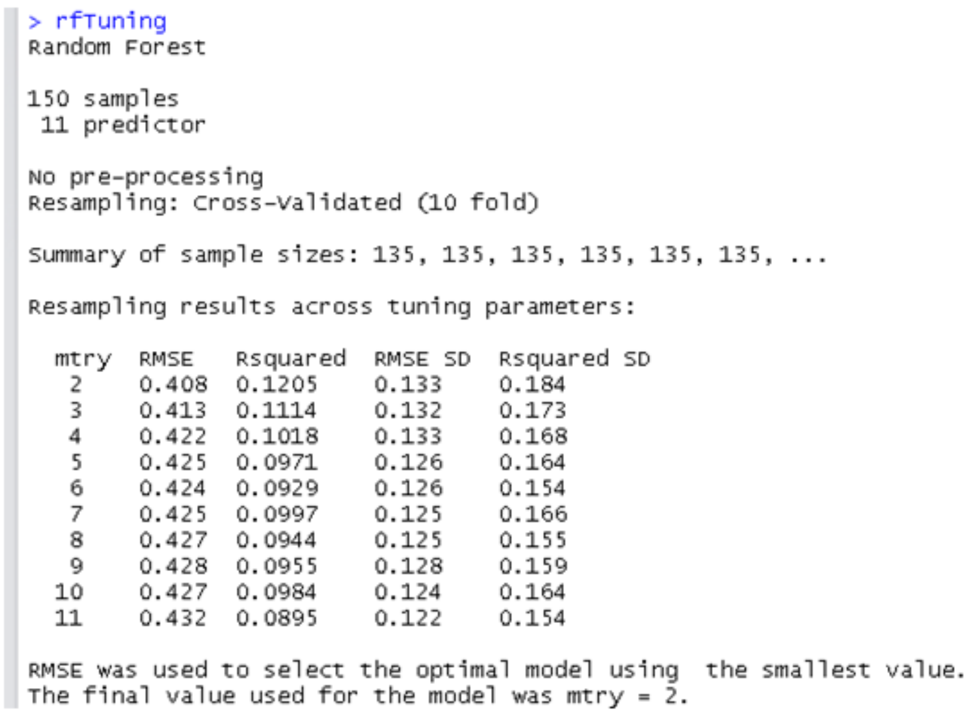
```
> postResample(pred.ridge,y.test)
     RMSE  Rsquared
0.7370464 0.2784749

> error.ridge
[1] 0.4789009
```

Table 7. Result of Ridge



```
> coef.opt.lasso
12 x 1 sparse Matrix of class "dgCMatrix"
                               1
(Intercept)           5.927986907
fixed.acidity         .
volatile.acidity     -0.100652713
citric.acid           .
residual.sugar        0.050863070
chlorides            -0.025181343
free.sulfur.dioxide   0.058887801
total.sulfur.dioxide  .
density               .
pH                    0.012952562
sulphates             0.005997758
alcohol               0.363065547
> postResample(pred.lasso,y.test)
     RMSE  Rsquared
0.7484261 0.2625427
> error.lasso
[1] 0.484789
```

Table 8. Result of LASSO



```
> rfTuning
Random Forest

150 samples
 11 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)

Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...

Resampling results across tuning parameters:

  mtry  RMSE   Rsquared  RMSE SD  Rsquared SD
   2    0.408  0.1205    0.133    0.184
   3    0.413  0.1114    0.132    0.173
   4    0.422  0.1018    0.133    0.168
   5    0.425  0.0971    0.126    0.164
   6    0.424  0.0929    0.126    0.154
   7    0.425  0.0997    0.125    0.166
   8    0.427  0.0944    0.125    0.155
   9    0.428  0.0955    0.128    0.159
  10    0.427  0.0984    0.124    0.164
  11    0.432  0.0895    0.122    0.154

RMSE was used to select the optimal model using  the smallest value.
The final value used for the model was mtry = 2.
```

Table 9. Random Forest 10 fold cross validation

R codes for the whole process,
data_white <-

```
read.table("/Users/Wenxin_AN/Documents/Master/ada/project/winequality-white.csv
",sep=";",header=TRUE)
head(data_white)
data_white_x <- data_white[,-12]
# Data Source
attach(data_white)
Class_variables<- sapply(data_white,is.numeric)
str(data_white)
# Sample R code for EDA/Preparation
library(Hmisc)
describe(data_white,digit=5)
library(caret)
featurePlot(x=data_white[,-12],y=data_white[,12],between=list(x=1,y=1),type=c("g",
"p","smooth"))

# Plot the boxplot for outlier checking
par(mfrow=c(1,5), oma = c(1,1,0,0) + 0.1,    mar = c(3,3,1,1) + 0.1)
attach(data_white)
boxplot(fixed.acidity, col="slategray2", pch=19)
mtext("Fixed Acidity", cex=0.8, side=1, line=2)
boxplot(volatile.acidity, col="slategray2", pch=19)
mtext("Volatile Acidity", cex=0.8, side=1, line=2)
boxplot(citric.acid, col="slategray2", pch=19)
mtext("Citric Acid", cex=0.8, side=1, line=2)
boxplot(residual.sugar, col="slategray2", pch=19)
mtext("Residual Sugar", cex=0.8, side=1, line=2)
boxplot(chlorides, col="slategray2", pch=19)
mtext("Chlorides", cex=0.8, side=1, line=2)

par(mfrow=c(1,6), oma = c(1,1,0,0) + 0.1,    mar = c(3,3,1,1) + 0.1)
boxplot(free.sulfur.dioxide, col="slategray2", pch=19)
mtext("free sulfur dioxide", cex=0.8, side=1, line=2)
boxplot(total.sulfur.dioxide, col="slategray2", pch=19)
mtext("total sulfur dioxide", cex=0.8, side=1, line=2)
boxplot(density, col="slategray2", pch=19)
mtext("density", cex=0.8, side=1, line=2)
boxplot(pH, col="slategray2", pch=19)
mtext("pH Value", cex=0.8, side=1, line=2)
boxplot(sulphates, col="slategray2", pch=19)
mtext("sulphates", cex=0.8, side=1, line=2)
boxplot(alcohol, col="slategray2", pch=19)
mtext("alcohol", cex=0.8, side=1, line=2)

# Correlation checking
```

```
Correlation<- cor(data_white[,-12])
Correlation
cor(data_white[,-12], method="spearman")
par(mfrow=c(1,1))
library(corrplot)
corrplot(Correlation)

# Sample R code for Preparing Data
limout <- rep(0,11)
for (i in 1:11){
   t1 <- quantile(data_white[,i], 0.75)
   t2 <- IQR(data_white[,i], 0.75)
   limout[i] <- t1 + 1.5*t2
}
WhiteWineIndex <- matrix(0, 4898, 11)
for (i in 1:4898)
   for (j in 1:11){
      if (data_white[i,j] > limout[j]) WhiteWineIndex[i,j] <- 1
   }
WWInd <- apply(WhiteWineIndex, 1, sum)
WhiteWineTemp <- cbind(WWInd, data_white)
Indexes <- rep(0, 208)
j <- 1
for (i in 1:4898){
   if (WWInd[i] > 0) {Indexes[j]<- i
                               j <- j + 1}
   else j <- j
}
WhiteWineLib <-data_white[-Indexes,]
set.seed(1)
indexes = sample(1:nrow(WhiteWineLib), size=0.75*nrow(WhiteWineLib))
WWTrain75 <- WhiteWineLib[indexes,]
WWTest25 <- WhiteWineLib[-indexes,]
X.train1 <- WWTrain75[,-12]
X.test1    <- WWTest25[,-12]
Y.train1 <- WWTrain75[,12]
Y.test1    <- WWTest25[,12]

# Sample R code for Multiple Linear Regression
options(digits=7)
library(car)
Qfit1 <- lm(quality ~ ., data=WWTrain75)
summary(Qfit1)
#vif(Qfit1)
```

```
#postResample(round(Pred.LM),Y.test1)
Pred.LM=predict(Qfit1,X.test1)
postResample((Pred.LM),Y.test1)
(err <- sum(round(Pred.LM)!=Y.test1)/length(Y.test1))

par(mfrow=c(1,1))
truehist(residuals(Qfit1), h = 0.25, col="slategray3")
#qqPlot(residuals(Qfit1), pch=19, col="darkblue", cex=0.6)
mtext("Distribution of Residuals for Qfit1", outer=T, side=1, line = 2)

# Polynomial Regression
Qfit6 <- lm(quality ~ poly(alcohol,2) + poly(volatile.acidity,2) + residual.sugar +
poly(free.sulfur.dioxide,2) + chlorides + sulphates + poly(pH,2), data=WWTrain75)
summary(Qfit6)
Pred.LM_6=predict(Qfit6,X.test1)
postResample((Pred.LM_6),Y.test1)
(err <- sum(round(Pred.LM_6)!=Y.test1)/length(Y.test1))
residualPlots(Qfit6, pch=19, col="blue", cex=0.6)

# Ridge and Lasso
set.seed(1)
indexes = sample(1:nrow(WhiteWineLib), size=0.75*nrow(WhiteWineLib))
x.train <- WhiteWineLib[indexes,]
x.test <- WhiteWineLib[-indexes,]
y.train=x.train$quality
y.test=x.test$quality
x.train=x.train[,-12]
x.test=x.test[,-12]
x.train=as.data.frame(x.train)
x.test=as.data.frame(x.test)

mean.train <- apply(x.train, 2, mean)
sd.train <- apply(x.train, 2, sd)

x.train <- scale(x.train)
x.train[,sd.train==0] <- 0

x.test <- scale(x.test, center = mean.train, scale=sd.train)
x.test[,sd.train==0] <- 0

library("glmnet")
set.seed(2)
fit.ridge <- cv.glmnet(x.train, y.train, alpha = 0)
#plot(fit.ridge)
```

```
coef.opt.ridge <- coef(fit.ridge)
#test
pred.ridge <- predict(fit.ridge,x.test)
pred.ridge.final=round(pred.ridge)
error.ridge=sum(pred.ridge.final!=y.test)/length(y.test)
coef.opt.ridge
postResample(pred.ridge,y.test)
error.ridge

#lasso
set.seed(2)
fit.lasso <- cv.glmnet(x.train, y.train, alpha = 1, standardize=F,family="gaussian")
#plot(fit.lasso)
coef.opt.lasso <- coef(fit.lasso)
#test
pred.lasso <- predict(fit.lasso,x.test)
postResample(pred.lasso,y.test)
pred.lasso.final=round(pred.lasso)
error.lasso=sum(pred.lasso.final!=y.test)/length(y.test)
coef.opt.lasso
postResample(pred.lasso,y.test)
error.lasso

#Nonlinear models
#Random Forest
wine<-read.csv(file="winequality-white.csv",sep=";",header=TRUE)
library(caret)
X<-wine[,-12]
quality<-wine[,12]
pre.x=predict(P,wine[,-12])
Y=wine[,12]
Y=data.frame(Y)
trainset<-createDataPartition(wine[,12],p=3/4,list=F)
train.x<-pre.x[trainset,]
test.x<-pre.x[-trainset,]
train.y<-Y[trainset,]
test.y<-Y[-trainset,]
library(randomForest)
trainx=train.x[1:150,]
trainy=train.y[1:150]
rfTuning=train(trainx,trainy,method="rf",tuneLength=10,trControl=trainControl(meth
od="cv"))
rfModel<-randomForest(train.x,train.y,importance=TRUE,ntrees=1000,mtry=2)
rfcv(train.x,train.y,cv.fold=10,scale="log",step=0.5,mtry=function(p)2,recursive=FA
LSE)
varImpPlot(rfModel,main="Random Forest")
predict.y<-predict(rfModel,test.x)
postResample(pred=predict.y,obs=test.y)
err<-sum(round(predict.y)!=test.y)/length(test.y)

# NN
library(nnet)
nnetFit=nnet(trainwine,trainqty,size=5,decat=0.01,linout=TRUE,trace=FALSE,maxit
=500,MAXNWts=5*(ncol(trainwine)+1)+5+1)
nnetAvg=avNNet(trainwine,trainqty,size=5,decat=0.01,repeats=5,linout=TRUE,trace
=FALSE,maxit=500,MAXNWts=5*(ncol(trainwine)+1)+5+1)
predfit=predict(nnetFit,testwine)
predavg=predict(nnetAvg,testwine)
nnvaluesfit=data.frame(obs=testqty,pred=predfit)
nnvaluesavg=data.frame(obs=testqty,pred=predavg)
defaultSummary(nnvaluesfit)
defaultSummary(nnvaluesavg)



toohigh<-findCorrelation(cor(trainwine),cutoff=0.7)
trainnnet<-trainwine[,-toohigh]
testnnet<-testwine[,-toohigh]
nnetgrid<-expand.grid(.decay=c(0,0.01,.1),.size=c(1:10))
set.seed(100)
nnettune<-train(trainwine,trainqty,method="nnet",tuneGrid=nnetgrid,trControl=train
Control(method="cv"),preProc=c("center","scale"),linout=TRUE,trace=FALSE,Max
NWts=5*(ncol(trainnnet)+1)+5+1,maxit=500)
nnettune
nnetFit=nnet(trainwine,trainqty,size=5,decat=0.1,linout=TRUE,trace=FALSE,maxit=
500,MAXNWts=5*(ncol(trainwine)+1)+5+1)
> predqtyFIT=predict(nnetFit,trainwine)
lmvaluesFIT=data.frame(obs=trainqty,pred=predqtyFIT)
defaultSummary(lmvaluesFIT)
predqtyFIT2=predict(nnetFit,testwine)
lmvaluesFIT2=data.frame(obs=testqty,pred=predqtyFIT2)
defaultSummary(lmvaluesFIT2)
error

# Classification Models
data$quality <- as.factor(data$quality)
limout.up <- rep(0,11)
limout.low <- rep(0,11)
```

```
for (i in 1:11){
  t1 <- quantile(data[,i], 0.75)
  t2 <- IQR(data[,i], 0.75)
  t3 <- quantile(data[,i],0.25)
  limout.up[i] <- t1 + 1.5*t2
  limout.low[i]<- t3 - 1.5*t2
}
dataIndex <- matrix(0, 4898, 11)
for (i in 1:4898)
  for (j in 1:11){
    if (data[i,j] > limout.up[j]||data[i,j]< limout.low[j]) dataIndex[i,j] <- 1
  }
WWInd <- apply(dataIndex, 1, sum)
Indexes <- rep(0, 883)
j <- 1
for (i in 1:4898){
  if (WWInd[i] > 0) {Indexes[j]<- i
                     j <- j + 1}
  else j <- j
}
dataLib <-data[-Indexes,]
indexes = sample(1:nrow(dataLib), size=0.75*nrow(dataLib))
data <- dataLib
train.data <- data[indexes,]
test.data <- data[-indexes,]

require(rpart)
p<-ncol(data)
fit.tree <- rpart(quality~.,data=train.data,method="class")
predict.tree <- predict(fit.tree,test.data[,-p],type="class")
err.tree<-sum(predict.tree!=test.data$quality)/nrow(test.data)

require(randomForest)
set.seed(123)
fit.rf<-randomForest(quality~.,data=train.data,importance=TRUE,proximity=TRUE,n
tree=1000)
round(importance(fit.rf),2)
predict.rf <- predict(fit.rf,test.data[,-p],type="class")
predict.rf$confusion
err.rf <-sum(predict.rf!=test.data$quality)/nrow(test.data)

require(ada)
require(adabag)
fit.ada <- boosting.cv(quality~.,data=train.data)
```

```
predict.ada <- predict(fit.ada,newdata=test.data)
err.ada <-sum(predict.ada$class!=test.data$quality)/nrow(test.data)



require(e1071)
fit.svm <- svm(quality~.,data=train.data,gamma=1,type="C",cross=10)
predict.svm <-predict(fit.svm,test.data[,-p])
# table(pred=predict.svm,true=test.data$quality)
err.svm <-sum(predict.svm!=test.data$quality)/nrow(test.data)

err.svm
err.tree
err.rf
err.ada
```

Reference

[1] Cortez.P., (2009). UCI Machine Learning Repository. Retrieved from, http://archive.ics.uci.edu/ml/datasets/Wine+Quality

[2] Cortez.P. Cerdeira.A Almeida.F, Matos.T. & Reis.J., (2009). Modeling wine preferences by data mining from physicochemical properties, Decision Support Systems.

[3] Appalasamy.P., Mustapha.A., Rizal. N.D., Johari.F. & Mansor,A.F., (2012). Classification-based Data Mining Approach for Quality Control in Wine Production. Journal of Applied Sciences, 12: 598-601. Retrieved from, http://scialert.net/fulltext/?doi=jas.2012.598.601&org=11#89121_b

[4] Feilhauer P., (2012). Prediction of wine quality. Retrieved from, http://www.feec.vutbr.cz/EEICT/2012/sbornik/03doktorskeprojekty/03kybernetikaaautomatizace/05-xfeilh00.pdf

[5] The Pennsylvania State University., (2014). Analysis of wine quality data. Retrieved from, https://onlinecourses.science.psu.edu/stat857/node/223