CSC 633 Machine Learning Final Project

Project Title: Wine Quality Data Analysis

Project Description: This project will utilize UCI wine quality data which we will analyze using four methods including decision tree, Linear model, K-nearest neighbors, and support vector machines. These methods will be applied and compared to predict wine taste from chemical properties.

Linear Regression - by Xiaofang Yu

1. Introduction

This project intends to predict the quality of the red wines based on its physicochemical features using the linear regression methods. Use the red wine quality data set described in the web page below:
http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

This red wine data set (winequality-red.csv) contains 1599 observations of 11 attributes. The median score of the wine tasters is given in the last column. The delimiter used in this file is a semi colon.

In this section, first, an ordinary least squares linear model will be created by using the lm function on the first 75% observations as train data set. Then, check the model's performance on the rest 25% observations. After that, the trained/tested model will be applied to predict the results for the whole data set and measure how well the model worked.

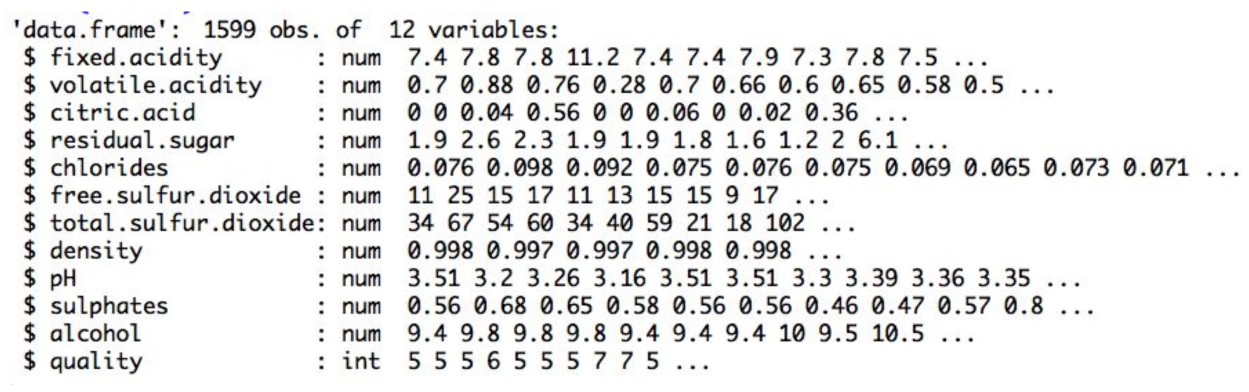
As a comparison to the ordinary linear regression, a ridge regression will be performed on the train data set (the first 75% observations). Then, Apply the ridge regression model to the test data set (the last 25% observations)

Compare the coefficients resulting from the ridge regression with the coefficients that were obtained using the ordinary least square model.

2. Preprocessing data

First, load the data set and display its info using the str function in R:

```
library(caret)
library(corrplot)
# set working directory
setwd("/Users/xiaofangyu/ITUCourses/CSC633ML/finalProject")
rm(list=ls())  # clean up predefined objects
# load data
redwine <- read.csv("winequality-red.csv", header=TRUE, sep=";")
str(redwine)
```



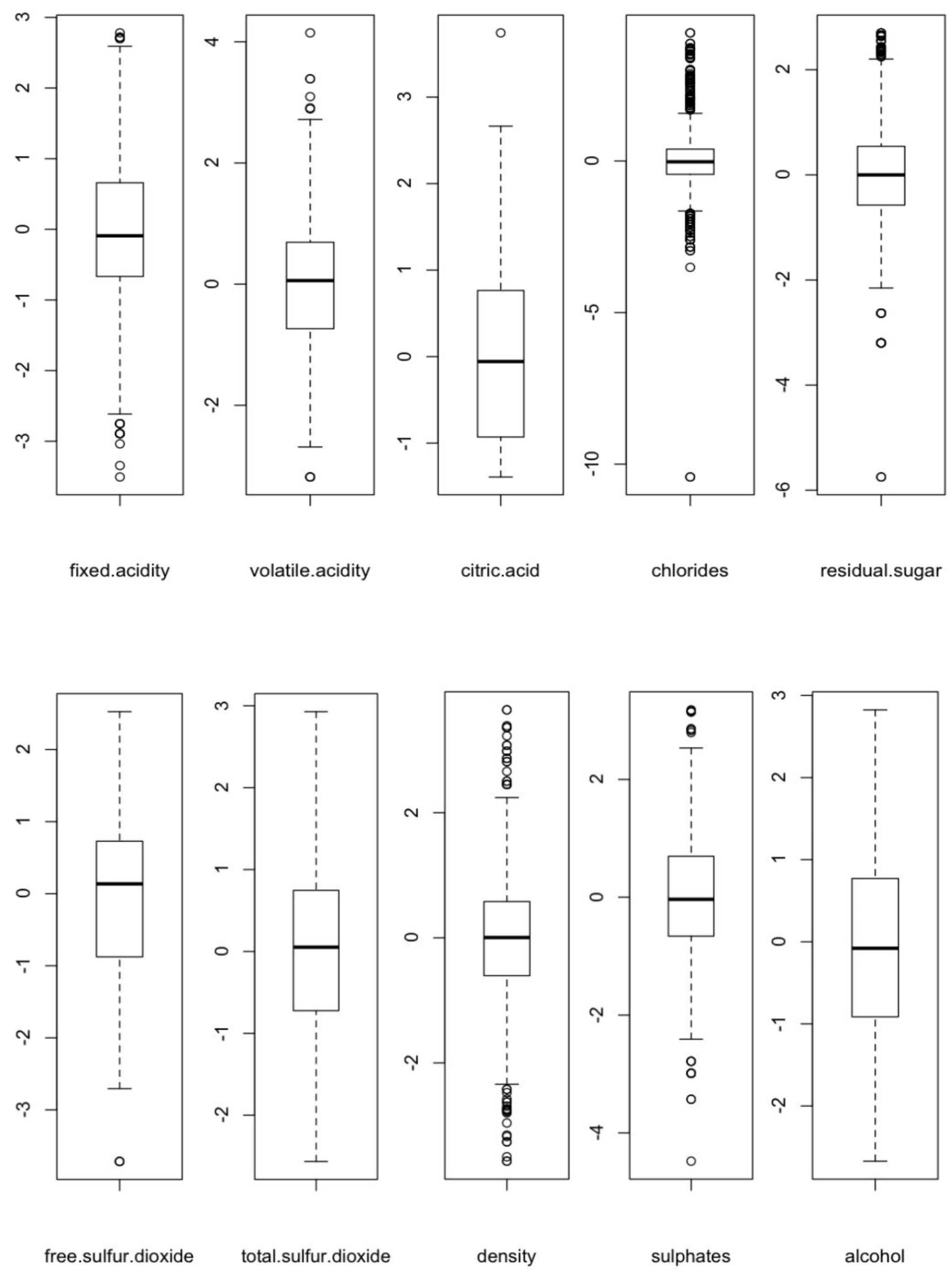
```
'data.frame':  1599 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

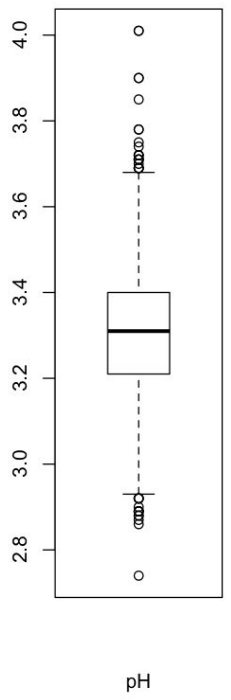
Second, preprocess the data set:

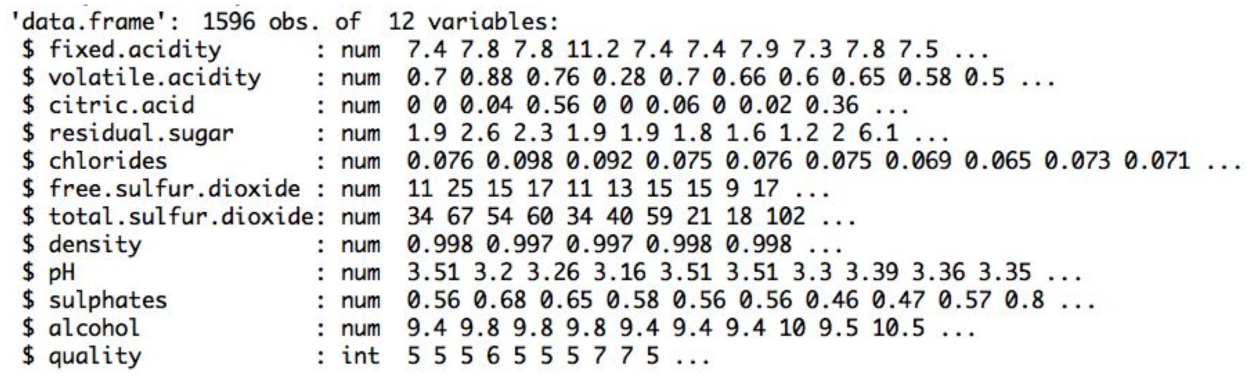
```
preprocess_red <- preProcess(redwine[,1:11], c("BoxCox", "center", "scale"))
redwine_new <- data.frame(trans = predict(preprocess_red, redwine))
```

Checking for outliers using boxplots and remove outliers:

```
boxplot(redwine_new$trans.fixed.acidity,xlab="fixed.acidity", show.names=TRUE)
boxplot(redwine_new$trans.volatile.acidity,xlab="volatile.acidity", show.names=TRUE)
boxplot(redwine_new$trans.citric.acid,xlab="citric.acid", show.names=TRUE)
boxplot(redwine_new$trans.residual.sugar,xlab="residual.sugar", show.names=TRUE)
boxplot(redwine_new$trans.chlorides,xlab="chlorides", show.names=TRUE)
boxplot(redwine_new$trans.free.sulfur.dioxide,xlab="free.sulfur.dioxide", show.names=TRUE)
boxplot(redwine_new$trans.total.sulfur.dioxide,xlab="total.sulfur.dioxide", show.names=TRUE)
boxplot(redwine_new$trans.density,xlab="density", show.names=TRUE)
boxplot(redwine_new$trans.PH,xlab="PH", show.names=TRUE)
boxplot(redwine_new$trans.sulphates,xlab="sulphates", show.names=TRUE)
boxplot(redwine_new$trans.alcohol,xlab="alcohol", show.names=TRUE)

#remove outliers after scaling, centering and transforming data
redwine_new <- redwine[!abs(redwine_new$trans.fixed.acidity) > 3,]

#display preprocessed data using str
str(redwine_new)
```


fixed.acidity
volatile.acidity
citric.acid
chlorides
residual.sugar
free.sulfur.dioxide
total.sulfur.dioxide
density
sulphates
alcohol



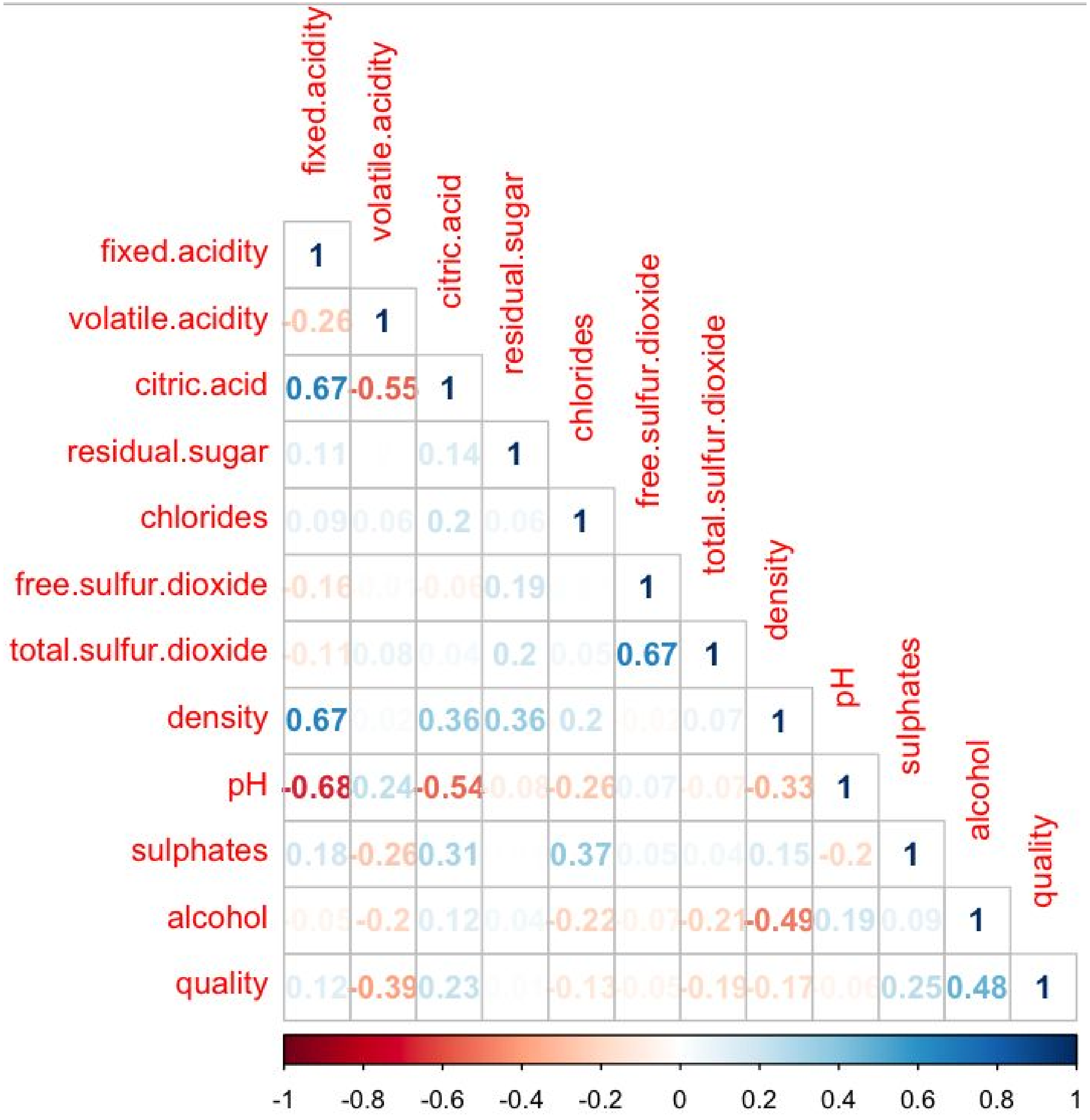


```
'data.frame':	1596 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
 $ volatile.acidity    : num  0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
 $ citric.acid         : num  0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
 $ residual.sugar      : num  1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
 $ chlorides           : num  0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
 $ free.sulfur.dioxide : num  11 25 15 17 11 13 15 15 9 17 ...
 $ total.sulfur.dioxide: num  34 67 54 60 34 40 59 21 18 102 ...
 $ density             : num  0.998 0.997 0.997 0.998 0.998 ...
 $ pH                  : num  3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
 $ sulphates           : num  0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
 $ alcohol             : num  9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 7 7 5 ...
```

3. **Check the correlation**

Calculate pairwise correlation matrix to see how different variables are related to quality. Plotting a heat map:

```
# Find correlations between predictors and quality
corrplot(cor(redwine_new), type = "lower", method = "number")
```



From the above chart, the top 5 predictors with highest correlations with quality are:

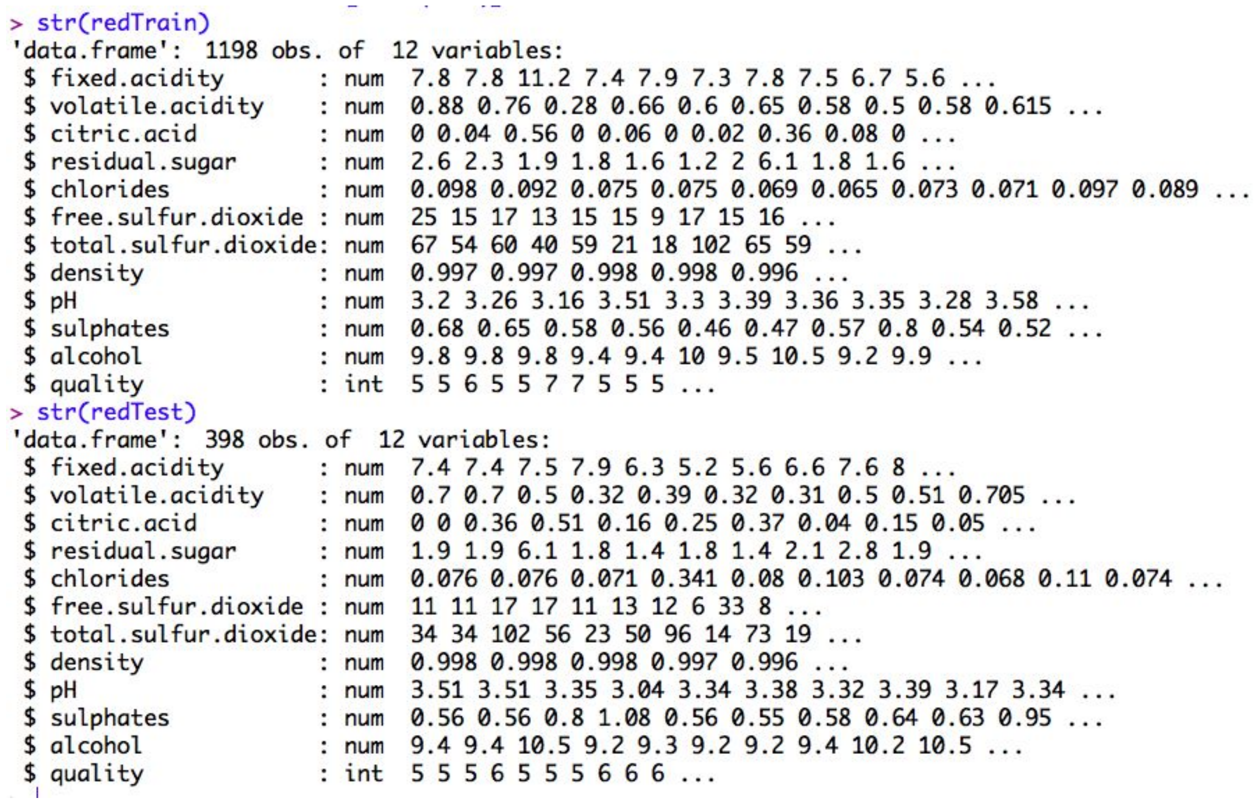
- Fixed acidity
- Citric acid
- Residual sugar
- sulphates
- alcohol

4. Create train and test sets

Split the data into train set and test set using the *createDataPartition* function (75/25 train/test split):

```
# 75/25 train/test split
set.seed(1)
redSplit <- createDataPartition(redwine_new$quality,
                    p = 0.75,
                    list = FALSE)
redTrain <- redwine_new[redSplit,]
redTest  <- redwine_new[-redSplit,]
```

Check redTrain and redTest using the *str* function:

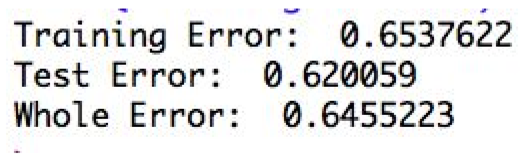


```
> str(redTrain)
'data.frame':	1198 obs. of  12 variables:
 $ fixed.acidity       : num  7.8 7.8 11.2 7.4 7.9 7.3 7.8 7.5 6.7 5.6 ...
 $ volatile.acidity    : num  0.88 0.76 0.28 0.66 0.6 0.65 0.58 0.5 0.58 0.615 ...
 $ citric.acid         : num  0 0.04 0.56 0 0.06 0 0.02 0.36 0.08 0 ...
 $ residual.sugar      : num  2.6 2.3 1.9 1.8 1.6 1.2 2 6.1 1.8 1.6 ...
 $ chlorides           : num  0.098 0.092 0.075 0.075 0.069 0.065 0.073 0.071 0.097 0.089 ...
 $ free.sulfur.dioxide : num  25 15 17 13 15 15 9 17 15 16 ...
 $ total.sulfur.dioxide: num  67 54 60 40 59 21 18 102 65 59 ...
 $ density             : num  0.997 0.997 0.998 0.998 0.996 ...
 $ pH                  : num  3.2 3.26 3.16 3.51 3.3 3.39 3.36 3.35 3.28 3.58 ...
 $ sulphates           : num  0.68 0.65 0.58 0.56 0.46 0.47 0.57 0.8 0.54 0.52 ...
 $ alcohol             : num  9.8 9.8 9.8 9.4 9.4 10 9.5 10.5 9.2 9.9 ...
 $ quality             : int  5 5 6 5 5 7 7 5 5 5 ...
> str(redTest)
'data.frame':	398 obs. of  12 variables:
 $ fixed.acidity       : num  7.4 7.4 7.5 7.9 6.3 5.2 5.6 6.6 7.6 8 ...
 $ volatile.acidity    : num  0.7 0.7 0.5 0.32 0.39 0.32 0.31 0.5 0.51 0.705 ...
 $ citric.acid         : num  0 0 0.36 0.51 0.16 0.25 0.37 0.04 0.15 0.05 ...
 $ residual.sugar      : num  1.9 1.9 6.1 1.8 1.4 1.8 1.4 2.1 2.8 1.9 ...
 $ chlorides           : num  0.076 0.076 0.071 0.341 0.08 0.103 0.074 0.068 0.11 0.074 ...
 $ free.sulfur.dioxide : num  11 11 17 17 11 13 12 6 33 8 ...
 $ total.sulfur.dioxide: num  34 34 102 56 23 50 96 14 73 19 ...
 $ density             : num  0.998 0.998 0.998 0.997 0.996 ...
 $ pH                  : num  3.51 3.51 3.35 3.04 3.34 3.38 3.32 3.39 3.17 3.34 ...
 $ sulphates           : num  0.56 0.56 0.8 1.08 0.56 0.55 0.58 0.64 0.63 0.95 ...
 $ alcohol             : num  9.4 9.4 10.5 9.2 9.3 9.2 9.2 9.4 10.2 10.5 ...
 $ quality             : int  5 5 5 6 5 5 5 6 6 6 ...
```

5. Ordinary least squares linear model

Create an ordinary least squares linear model and train it using the *lm* function on the train data set (the first 75% observations). Then, check the model's performance on the rest 25% observations. After that, apply the trained/tested model to predict the results for the whole data set.

```
yTrain=redTrain[,12]
rmse=function(X,Y){return( sqrt(sum((X-Y)^2)/length(Y)) )}
lmodel=lm(quality~.,data=redTrain)
wineTrainErr=rmse(yTrain,predict(lmodel,newdata=redTrain[,-12]))
yTest=redTest[,12]
wineTestErr=rmse(yTest,predict(lmodel,newdata=redTest[,-12]))

#apply the trained/tested model to predict the results for the whole data set
whole=yTrain
whole=append(whole,yTest)
wholeErr = rmse(whole, predict(lmodel,newdata = rbind(redTrain[,-12],redTest[,-12])))
cat("Training Error: ", wineTrainErr, "\nTest Error: ", wineTestErr, "\nWhole Error: ", wholeErr,
"\n")
```

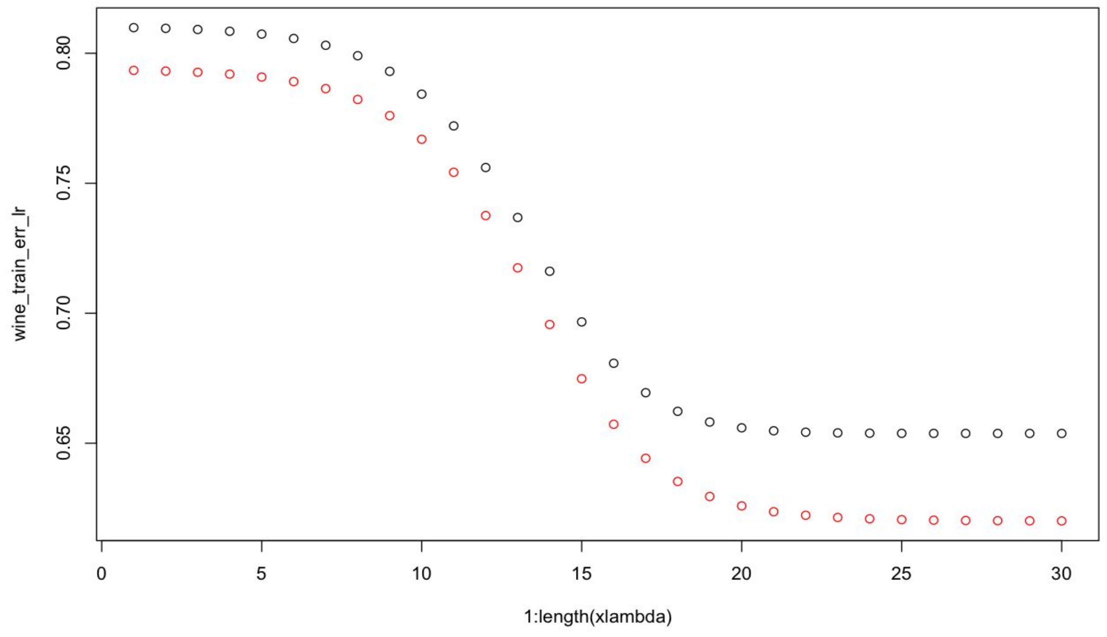


```
Training Error:  0.6537622
Test Error:  0.620059
Whole Error:  0.6455223
```

6. Ridge regression model

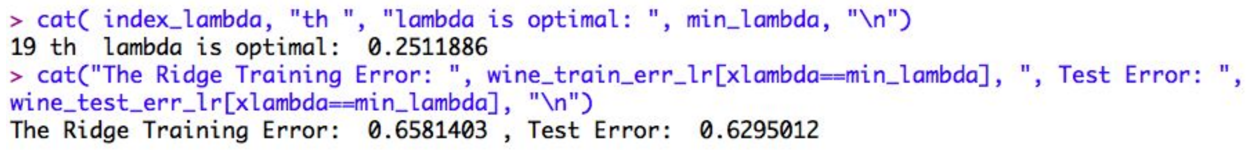
As a comparison to the ordinary linear regression, a ridge regression will be performed on the train data set (the first 75% observations). Then, Apply the ridge regression model to the test data set (the last 25% observations)

```
# Ridge regression
library("ridge")
xlambda=rep(0, times = 30)
for (i in seq(from = 0, to = 29)){
        exp <- (+3 -4*(i/20))
        xlambda [i+1] <- 10^exp
}

wine_train_err_lr = rep(0, times = length(xlambda))
wine_test_err_lr = rep(0, times = length(xlambda))
min_lambda = 10000
for( i in 1:length(xlambda)){
        lrmodel = linearRidge(quality~., data = redTrain, lambda = xlambda [i])
        wine_train_err_lr[i] = rmse(yTrain, predict(lrmodel, newdata = redTrain[,-12]))
        wine_test_err_lr[i] = rmse(yTest, predict(lrmodel, newdata = redTest[,-12]))
        if( i > 1 && wine_test_err_lr[i] < (wine_test_err_lr[i-1]-0.005)){
                min_lambda = xlambda [i]
                index_lambda=i
        }
}

plot(1:length(xlambda),wine_train_err_lr,
     ylim=c(min(wine_train_err_lr, wine_test_err_lr),
     max(wine_train_err_lr, wine_test_err_lr)))
points(1:length(xlambda),wine_test_err_lr, col='red')
```



Display the result and check how well this model does:

```
cat( index_lambda, "th ", "lambda is optimal: ", min_lambda, "\n")
cat("The Ridge Training Error: ", wine_train_err_lr[xlambda==min_lambda], ", Test Error: ",
wine_test_err_lr[xlambda==min_lambda], "\n")
```

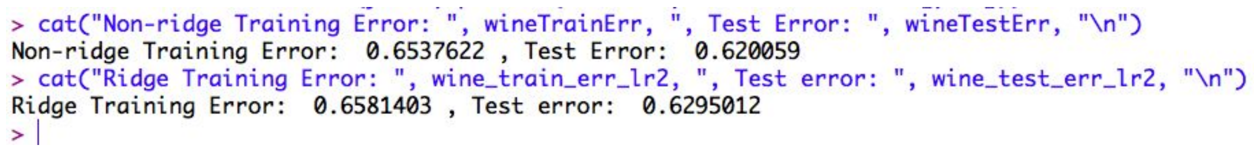


```
> cat( index_lambda, "th ", "lambda is optimal: ", min_lambda, "\n")
19 th  lambda is optimal:  0.2511886
> cat("The Ridge Training Error: ", wine_train_err_lr[xlambda==min_lambda], ", Test Error: ",
wine_test_err_lr[xlambda==min_lambda], "\n")
The Ridge Training Error:  0.6581403 , Test Error:  0.6295012
```

7. Compare two models

Compare the results and coefficients resulting from the ridge regression with the coefficients that were obtained using the ordinary least square model.

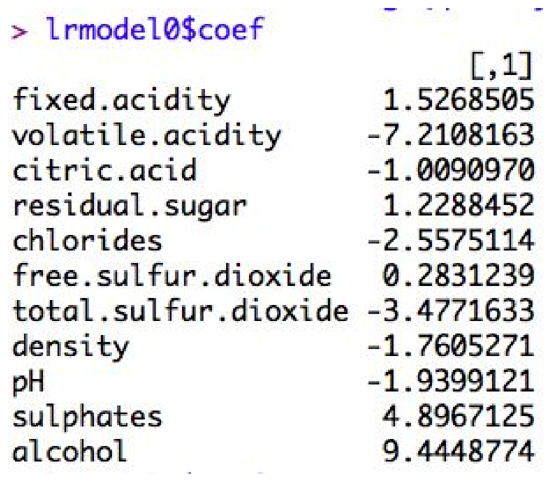
```
lrmodel2 = linearRidge(quality~., data = redTrain, lambda=min_lambda)
wine_train_err_lr2 = rmse(yTrain, predict(lrmodel2, newdata = redTrain[,-12]))
wine_test_err_lr2 = rmse(yTest, predict(lrmodel2, newdata = redTest[,-12]))
```



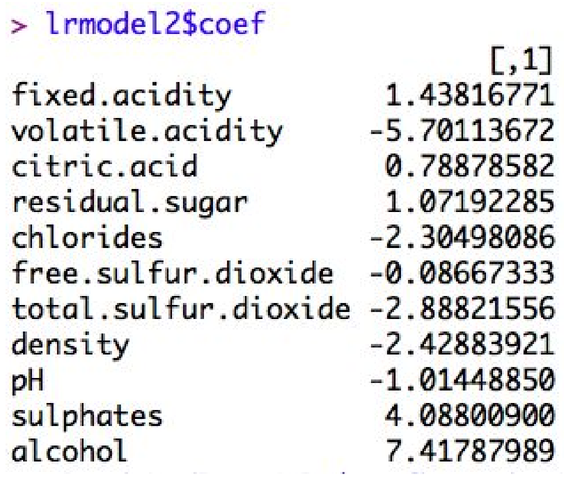
```
> cat("Non-ridge Training Error: ", wineTrainErr, ", Test Error: ", wineTestErr, "\n")
Non-ridge Training Error:  0.6537622 , Test Error:  0.620059
> cat("Ridge Training Error: ", wine_train_err_lr2, ", Test error: ", wine_test_err_lr2, "\n")
Ridge Training Error:  0.6581403 , Test error:  0.6295012
> |
```

From the above, two models generated similar results. The non-ridge regression model did slightly better, both are not very good though. Now let's take a look at the coefficients:

```
lrmodel0 = linearRidge(quality~., data = redTrain, lambda=0)
lrmodel0$coef
lrmodel2$coef
```



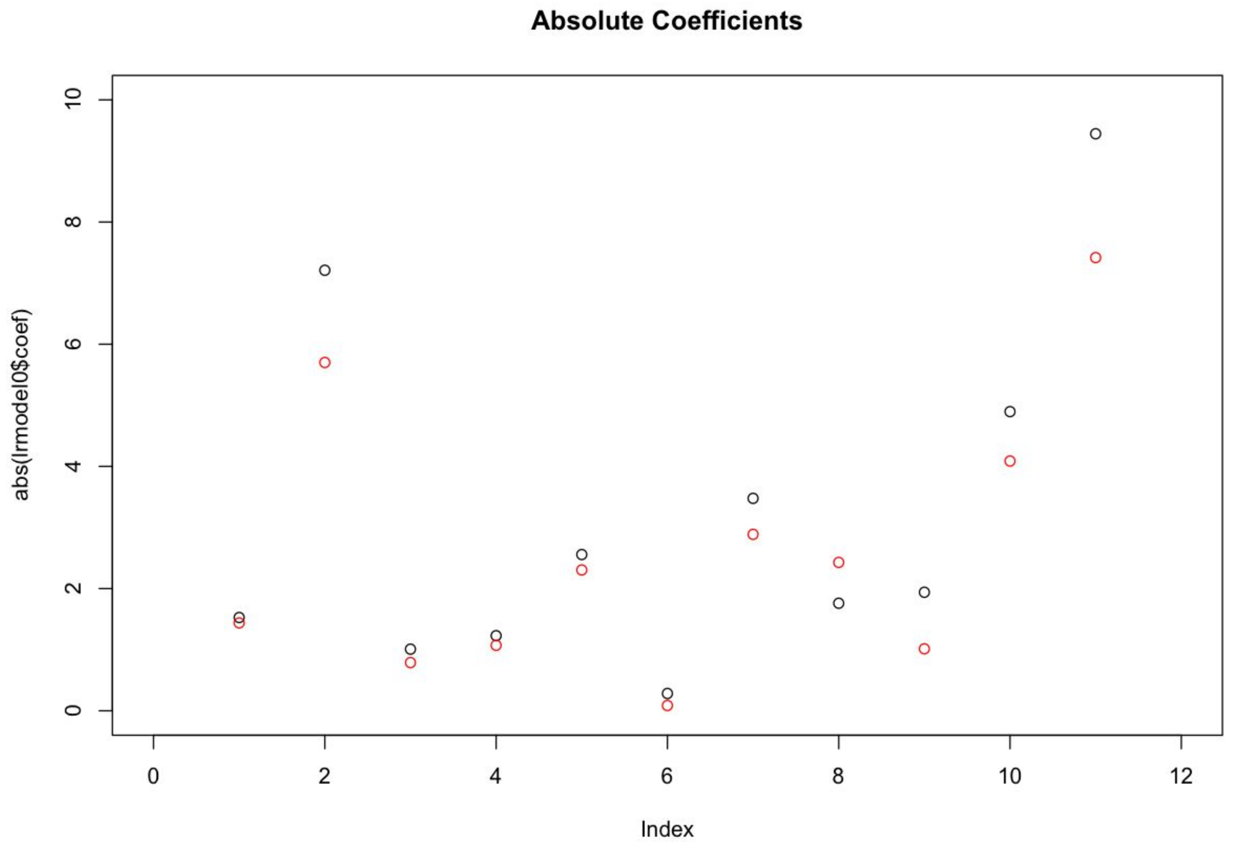
```
> lrmodel0$coef
                          [,1]
fixed.acidity        1.5268505
volatile.acidity    -7.2108163
citric.acid         -1.0090970
residual.sugar       1.2288452
chlorides           -2.5575114
free.sulfur.dioxide  0.2831239
total.sulfur.dioxide -3.4771633
density             -1.7605271
pH                  -1.9399121
sulphates            4.8967125
alcohol              9.4448774
```



```
> lrmodel2$coef
                           [,1]
fixed.acidity        1.43816771
volatile.acidity    -5.70113672
citric.acid          0.78878582
residual.sugar       1.07192285
chlorides           -2.30498086
free.sulfur.dioxide -0.08667333
total.sulfur.dioxide -2.88821556
density             -2.42883921
pH                  -1.01448850
sulphates            4.08800900
alcohol              7.41787989
```

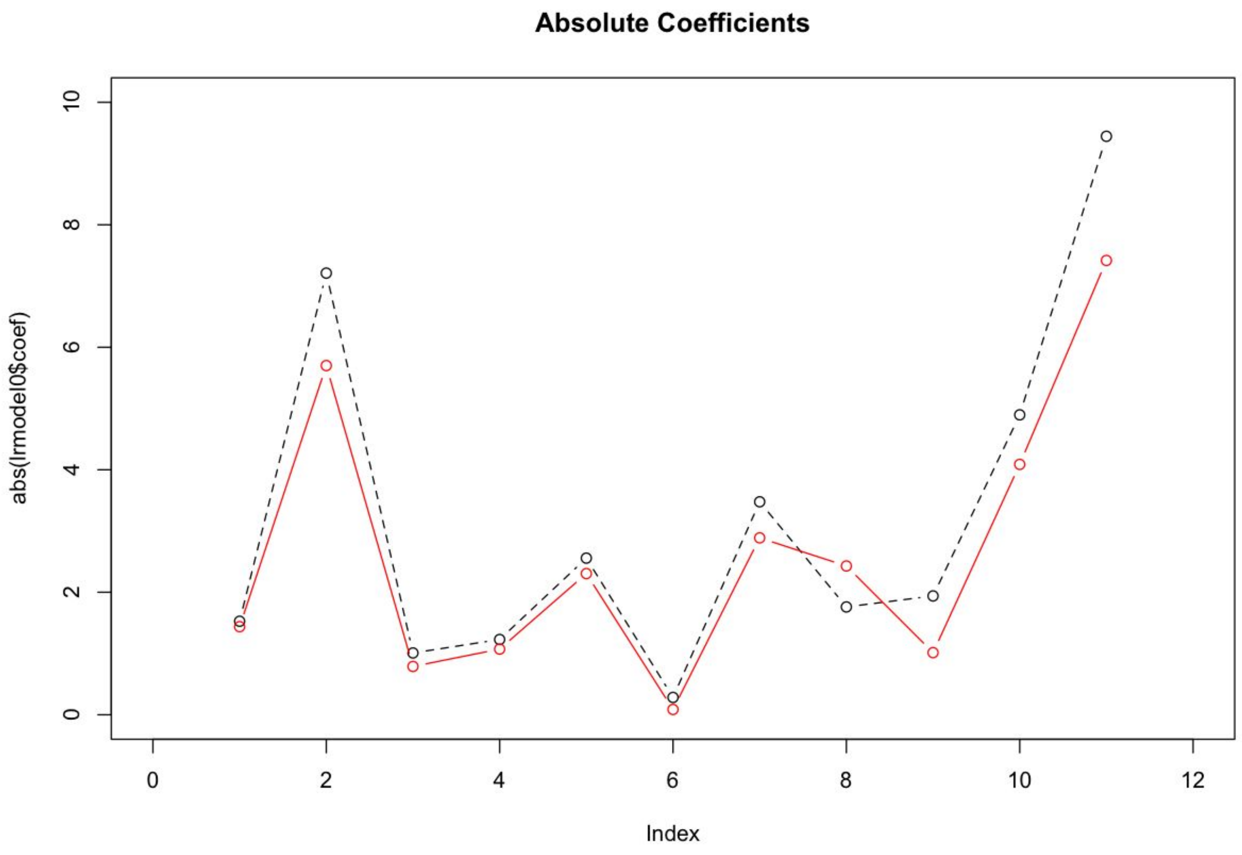
Use plot to generate a chart for the above absolute coefficients:

```
plot(abs(lrmodel0$coef), main="Absolute Coefficients", xlim=c(0,12),ylim=c(0,10))
points(abs(lrmodel2$coef),col="red")
```



Adding lines to make is clearer:

```
#use lines to make it clearer
plot(abs(lrmodel0$coef), main="Absolute Coefficients", xlim=c(0,12),ylim=c(0,10),
type="b", lty=2, )
lines(abs(lrmodel2$coef),col="red", type = "b", lty=1)
```



8. Conclusions

(1) The ordinary least squares linear model performed a little bit better with lower error result.

(2) Performance of model types (Ordinary least squares linear model vs. Ridge regression) were relatively similar overall. The results of both methods are not very good. We should try other models.

9. References

1. http://patriciahoffmanphd.com/machinelearning.php
2. https://towardsdatascience.com/machine-learning-linear-models-part-1-312757aab7bc
3. https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial
4. Cortez. P,Cerdeira.P,Almeida. F, Matos.F,& Reis. J. (2008), Modeling wine preferences by data mining from physicochemical properties, Decision Support Systems, v.47 n.4, p.547-553.
5. UCI Machine Learning Repository (2017). Wine quality data set, Retrieved Sep 10, 2017, from http://archive.ics.uci.edu/ml/datasets/Wine+Quality
6. https://onlinecourses.science.psu.edu/stat857/node/223
7. https://www.kaggle.com/shivamnijhawan96/wine-quality-using-linear-regression
8. http://rstudio-pubs-static.s3.amazonaws.com/175762_83cf2d7b322c4c63bf9ba2487b79e77e.html
9. https://rstudio-pubs-static.s3.amazonaws.com/98369_7d87780667b74901af21ff93c1c1e1db.html
10. http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/