Wine Color Prediction

Amir Dehkordi

Introduction

In this task we have a dataset related to red and white variants of Portuguese "Vinho Verde" wine and our objective is to predict the color of wine based on the 11 given variables/predictors in the data set. It would also be worth mentioning that we have another variable in addition to our 11 variables, "quality", but we do not consider it for training our models.

As mentioned before, we have 11 input variables and 1 output variable. The following defines the meaning and the type of input and output variables briefly.

Input variables:

1. **fixed acidity**: It is the amount of predominant fixed acids found in wines.(numeric)
2. **volatile acidity**:It refers to the steam distillable acids present in wine. (numeric)
3. **citric acid**: It can be added to finished wines to increase acidity and give a "fresh" flavor. (numeric)
4. **residual sugar**: It is from natural grape sugars leftover in a wine after the alcoholic fermentation finishes. (numeric)
5. **chlorides**: It is the major contributor to saltiness. (numeric)
6. **free sulfur dioxide**: They are available to react and thus exhibit both germicidal and antioxidant properties.(numeric)
7. **total sulfur dioxide**: The TSO2 level is also regulated by the U.S. Alcohol and Tobacco Tax and Trade Bureau. (numeric)
8. **density**: It is the mass per unit volume of wine or must at 20°C. (numeric)
9. **pH**: It expresses the acidity or alkalinity. (numeric)
10. **sulphates**: It acts as both a wine's preservative and enhancer. (numeric)
11. **alcohol**: It is the level of alcohol in the wine. (numeric)

Output variable:

1. **wine**: It is the color of the wine. (binary: "Red,"White")

As it can be observed, we have a Classification problem and we have to classify the color of the wines into "Red" and "White". Therefore, we are going to deploy five classification approaches to predict the class of our "wine". We use Logistic Regression, Linear Discriminant Analysis (LDA), Decision Tree Classifier, Bagging, and Random Forest models to train our data; and we will discuss each model further at the point it is employed.

Methods and Analysis

1.

The first step is to Import our data and check if we have any duplicated rows in our dataset.

```
# 1
# Importing data
library(dplyr)
df <- read.csv('winedata.csv')
sum(duplicated(df))
```

```
## [1] 1177
```

```
df <- distinct(df)
```

As it can be observed, we have 1177 duplicated rows in our dataset. Thus, we remove these duplicated rows with distinct() function. Finally, we come up with 5320 rows of data.

Now, it is the time to inspect through our data and execute a short Exploratory Data Analysis (EDA).

```
# Exploratory Data Analysis
head(df)
```

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.0             0.27        0.36           20.7     0.045
## 2           6.3             0.30        0.34            1.6     0.049
## 3           8.1             0.28        0.40            6.9     0.050
## 4           7.2             0.23        0.32            8.5     0.058
## 5           6.2             0.32        0.16            7.0     0.045
## 6           8.1             0.22        0.43            1.5     0.044
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  45                  170  1.0010 3.00      0.45     8.8
## 2                  14                  132  0.9940 3.30      0.49     9.5
## 3                  30                   97  0.9951 3.26      0.44    10.1
## 4                  47                  186  0.9956 3.19      0.40     9.9
## 5                  30                  136  0.9949 3.18      0.47     9.6
## 6                  28                  129  0.9938 3.22      0.45    11.0
##   quality  wine
## 1       6 White
## 2       6 White
## 3       6 White
## 4       6 White
## 5       6 White
## 6       6 White
```

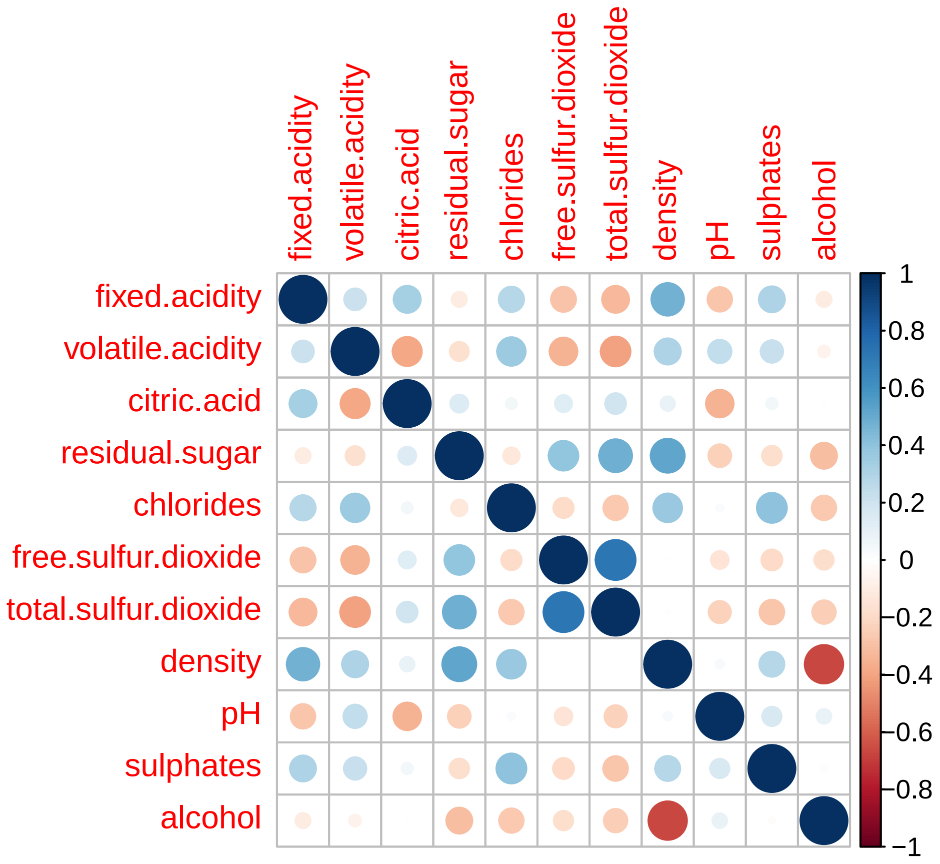
```
summary(df)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.400   1st Qu.:0.2300   1st Qu.:0.2400   1st Qu.: 1.800
##  Median : 7.000   Median :0.3000   Median :0.3100   Median : 2.700
##  Mean   : 7.215   Mean   :0.3441   Mean   :0.3185   Mean   : 5.048
##  3rd Qu.: 7.700   3rd Qu.:0.4100   3rd Qu.:0.4000   3rd Qu.: 7.500
##  Max.   :15.900   Max.   :1.5800   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  1.00      Min.   :  6.0        Min.   :0.9871
##  1st Qu.:0.03800   1st Qu.: 16.00      1st Qu.: 74.0        1st Qu.:0.9922
##  Median :0.04700   Median : 28.00      Median :116.0        Median :0.9947
##  Mean   :0.05669   Mean   : 30.04      Mean   :114.1        Mean   :0.9945
##  3rd Qu.:0.06600   3rd Qu.: 41.00      3rd Qu.:153.2        3rd Qu.:0.9968
##  Max.   :0.61100   Max.   :289.00      Max.   :440.0        Max.   :1.0390
##        pH           sulphates         alcohol          quality
##  Min.   :2.720   Min.   :0.2200   Min.   : 8.00   Min.   :3.000
##  1st Qu.:3.110   1st Qu.:0.4300   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.210   Median :0.5100   Median :10.40   Median :6.000
##  Mean   :3.225   Mean   :0.5334   Mean   :10.55   Mean   :5.796
##  3rd Qu.:3.330   3rd Qu.:0.6000   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :4.010   Max.   :2.0000   Max.   :14.90   Max.   :9.000
##      wine
##  Length:5320
##  Class :character
##  Mode  :character
##
##
##
```

From the head() function we can observe a common sense about our input and output variables. From the summary() function we can observe the Quartiles, Max, and Min of each numeric variable. We can also observe that the majority of our numeric variables are distributed commonly and do not have extreme outliers. However, "residual.sugar" and "free.sulfur.dioxide" variables have much bigger Max values that their 3rd Quartile, but because the dataset is given and is asked to be used, we ignore removing the outliers of such variables.

Here, let us check the correlation between variables.

```r
# Correlation Heatmap
library(corrplot)
dfcorr <- cor(df[,-c(12,13)])
corrplot(dfcorr, method = "circle")
```



The Correlation Heatmap is illustrated above and we can observe that there is no significant correlation between our variables, which is ideal. However, there is a considerable correlation between "density" and "alcohol", but because the dataset is given and is asked to be used, we ignore dropping any of the variables.

The next step is to compute the Mean and Standard Deviation for each of our variables.

```
# Computing the mean and standard deviation of the variables
submean <- aggregate(. ~ wine, data = df, FUN = mean)[,-13]
subsd <- aggregate(. ~ wine, data = df, FUN = sd)[,-13]
```

Next, for checking if the mean of our variables are significantly different for the two types of wine, "Red" and "White", we need to perform the t-test. For this purpose, we can easily conduct the test for all of our columns (variables) by using a 'for loop' and t.test() function. Please note that, we assume equal variance for each variable and it is performed for 95% interval.

```
# Conducting the t-test for variables in a for loop
df_split <- split(df, df$wine)
out <- c()
for (i in 1:11){
  cat("\nt-test between wine and ", colnames(df[i]))
  a = t.test(df_split$Red[,i], df_split$White[,i], var.equal = TRUE)
  b = t.test(df_split$Red[,i], df_split$White[,i], var.equal = TRUE)$p.value
  print(a)
  out <- c(out, b)
}
```

```
##
```

```
## t-test between wine and  fixed.acidity
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = 40.58, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1.400175 1.542325
## sample estimates:
## mean of x mean of y
##  8.310596  6.839346
##
##
## t-test between wine and  volatile.acidity
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = 61.606, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2410181 0.2568615
## sample estimates:
## mean of x mean of y
## 0.5294776 0.2805377
##
##
## t-test between wine and  citric.acid
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = -13.633, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.07091532 -0.05308397
## sample estimates:
## mean of x mean of y
## 0.2723326 0.3343322
##
##
## t-test between wine and  residual.sugar
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = -25.38, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -3.653379 -3.129461
## sample estimates:
## mean of x mean of y
##  2.523400  5.914819
##
##
## t-test between wine and  chlorides
##  Two Sample t-test
```

```
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = 42.049, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.04025022 0.04418687
## sample estimates:
##  mean of x  mean of y
## 0.08812362 0.04590507
##
##
## t-test between wine and  free.sulfur.dioxide
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = -38.337, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -19.96724 -18.02449
## sample estimates:
## mean of x mean of y
##  15.89330  34.88917
##
##
## t-test between wine and  total.sulfur.dioxide
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = -70.338, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -92.88618 -87.84889
## sample estimates:
## mean of x mean of y
##  46.82597 137.19351
##
##
## t-test between wine and  density
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = 34.671, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.002754344 0.003084491
## sample estimates:
## mean of x mean of y
## 0.9967089 0.9937895
##
##
## t-test between wine and  pH
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = 23.856, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1049333 0.1237235
## sample estimates:
## mean of x mean of y
##  3.309787  3.195458
##
##
## t-test between wine and  sulphates
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = 41.031, df = 5318, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.1603103 0.1763977
## sample estimates:
## mean of x mean of y
## 0.6587049 0.4903509
##
##
## t-test between wine and  alcohol
##  Two Sample t-test
##
## data:  df_split$Red[, i] and df_split$White[, i]
## t = -4.2189, df = 5318, p-value = 2.496e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.23001612 -0.08406884
## sample estimates:
## mean of x mean of y
##  10.43232  10.58936
```

This is the output of our t-test for all of our variables. All of our variables have p-value nearly equal to 0. But, we discuss the findings after following table.

In this step, we create table of Mean, Standard Deviation, and p-value of the t-test for each variable.

```
# Creating the table of Means, STDs, and P-Values
out <- c(NA,out)
subdf <- rbind(submean, subsd, out)
rownames(subdf) <- c("Red Mean", "White Mean", "Red STD", "White STD", "P-Value")
round(subdf[,-1],4)
```

```
##            fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## Red Mean          8.3106           0.5295      0.2723         2.5234    0.0881
## White Mean        6.8393           0.2805      0.3343         5.9148    0.0459
## Red STD           1.7370           0.1830      0.1955         1.3523    0.0494
## White STD         0.8669           0.1034      0.1224         4.8616    0.0231
## P-Value           0.0000           0.0000      0.0000         0.0000    0.0000
##            free.sulfur.dioxide total.sulfur.dioxide density     pH sulphates
## Red Mean               15.8933              46.8260  0.9967 3.3098    0.6587
## White Mean             34.8892             137.1935  0.9938 3.1955    0.4904
## Red STD                  10.4473              33.4089  0.0019 0.1550    0.1707
## White STD                17.2100              43.1291  0.0029 0.1515    0.1135
## P-Value                   0.0000               0.0000  0.0000 0.0000    0.0000
##            alcohol
## Red Mean   10.4323
## White Mean 10.5894
## Red STD     1.0821
## White STD   1.2171
## P-Value     0.0000
```

The first row represents the Mean of each variable when the "wine" is "Red". The second row represents the Mean of each variable when the "wine" is "White". The third row represents the Standard Deviation of each variable when the "wine" is "Red". The fourth row represents the Standard Deviation of each variable when the "wine" is "White". The last row represents the p-value obtained from the performed t-test for each variable. As it can be observed, all p-values after rounding up to 4 digits are 0, thus, we can reject our null-hypothesis that states: "true difference in means is equal to 0" and we can conclude that mean of all predictors are significantly different for the two types of wine.

2.

In this part we set the seed equal to "1998", so that we can reproduce the experiment and results. Then, we devide the dataset into Training and Test sets randomly with proportion of 80% in training set and 20% in test set. Afterwards, for our Classification purposes, we convert the "White" wine to 1 and "Red Wine" to 0.

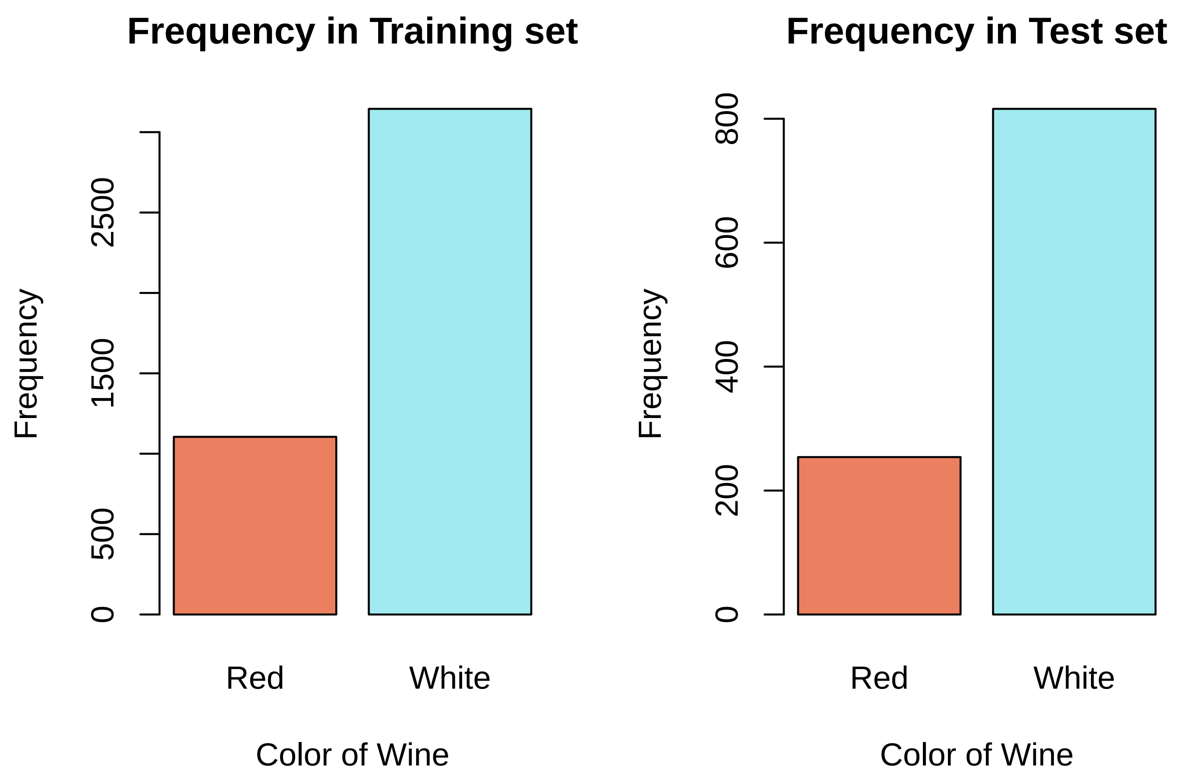
```r
# 2
# Splitting data into train and test data
set.seed(1986)
training <- sample(c(TRUE, FALSE), size = nrow(df), replace = TRUE, prob = c(0.8, 0.2))
df.train <- df[training,]
df.test <- df[!training,]

# If the color is white it is 1, otherwise 0
df.train$wine[df.train$wine == "White"] <- as.numeric(1)
df.test$wine[df.test$wine == "White"] <- as.numeric(1)
df.train$wine[df.train$wine == "Red"] <- as.numeric(0)
df.test$wine[df.test$wine == "Red"] <- as.numeric(0)
df.train$wine <- as.factor(df.train$wine)
df.test$wine <- as.factor(df.test$wine)
```

Now, we are supposed to summarise the number of red and white wine samples in training and test set. For this purpose, we create plots based on the Frequency and Proportion of the "Red" and "White".

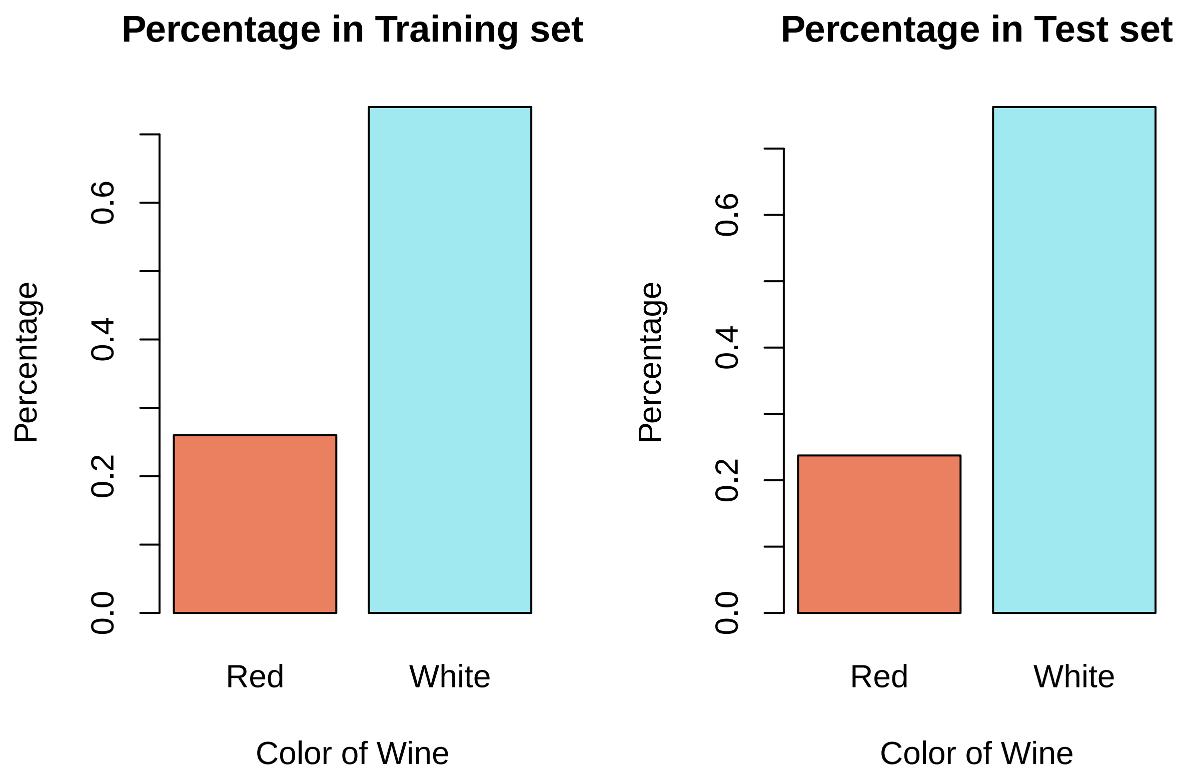
```r
# Creating plots for the training and test data
par(mfrow=c(1,2))
gptrain <- table(df.train$wine)
barplot(gptrain, xlab = "Color of Wine", ylab = "Frequency",
        main = "Frequency in Training set",
        col = c("#eb8060", "#a1e9f0"), names = c("Red", "White"))

gptest <- table(df.test$wine)
barplot(gptest, xlab = "Color of Wine", ylab = "Frequency",
        main = "Frequency in Test set",
        col =  c("#eb8060", "#a1e9f0"), names = c("Red", "White"))
```



```
gptrain.prop <- prop.table(gptrain)
barplot(gptrain.prop, xlab = "Color of Wine", ylab = "Percentage",
        main = "Percentage in Training set",
        col =  c("#eb8060", "#a1e9f0"), names = c("Red", "White"))

gptest.prop <- prop.table(gptest)
barplot(gptest.prop, xlab = "Color of Wine", ylab = "Percentage",
        main = "Percentage in Test set",
        col =  c("#eb8060", "#a1e9f0"), names = c("Red", "White"))
```



```r
par(mfrow=c(1,1))

# Table of the Frequency and Proportion
gpdf <- rbind(gptrain, prop.table(gptrain), gptest, prop.table(gptest))
round(gpdf,4)
```

```
##                 0         1
## gptrain 1105.0000 3145.0000
##            0.2600    0.7400
## gptest   254.0000  816.0000
##            0.2374    0.7626
```

From the table above, we can observe that 1105 out of 4225 observations in the train set are “Red” and the others are “White”,additionally, 254 out of 1095 observations in the test set are “Red” and the others are “White”. If we look at the proportions, we can observe that both train and test data have nearly the same proportion, around 25%.

3.

In this part we start training our different models and approaches with train data and then we test the models with test data. Finally inspect the important variables with respect to each approach.

Logistic Regression

We start classifiying by deploying the Logistic Regression approach as the following

```
# 3
# Logistic Regression
# Fit a model to the training data
logistic.model <- glm(wine ~ . - quality, data = df.train, family="binomial")
summary(logistic.model)
```

```
##
## Call:
## glm(formula = wine ~ . - quality, family = "binomial", data = df.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.6308  -0.0013   0.0200   0.0468   4.2206
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.472e+03  2.594e+02   9.533  < 2e-16 ***
## fixed.acidity         8.451e-01  3.022e-01   2.797  0.00516 **
## volatile.acidity     -5.200e+00  1.290e+00  -4.032 5.54e-05 ***
## citric.acid           2.577e+00  1.548e+00   1.664  0.09606 .
## residual.sugar        8.834e-01  1.103e-01   8.012 1.13e-15 ***
## chlorides            -2.574e+01  4.985e+00  -5.162 2.44e-07 ***
## free.sulfur.dioxide  -6.205e-02  1.524e-02  -4.073 4.65e-05 ***
## total.sulfur.dioxide  5.405e-02  5.922e-03   9.127  < 2e-16 ***
## density              -2.472e+03  2.631e+02  -9.395  < 2e-16 ***
## pH                    1.832e+00  1.819e+00   1.007  0.31388
## sulphates            -2.496e+00  1.643e+00  -1.519  0.12876
## alcohol              -2.617e+00  3.906e-01  -6.699 2.10e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4870.98  on 4249  degrees of freedom
## Residual deviance:  244.59  on 4238  degrees of freedom
## AIC: 268.59
##
## Number of Fisher Scoring iterations: 9
```

As it can be observed in the summary of our model, the important variables of our model, which had the p-values near 0, are "fixed.acidity", "volatile.acidity", "residual.sugar", "chlorides", "free.sulfur.dioxide", "total.sulfur.dioxide", "density", and "alcohol".

Now, Let us measure the misclassification of our Logistic Regression on test data.

```
# Test the model on the test data
prob <- predict(logistic.model, df.test, type = "response")
pred.test.glm <- ifelse(prob>0.5, 1, 0)
logistic.error <- mean(df.test$wine != pred.test.glm)
logistic.error
```

```
## [1] 0.008411215
```

Our Logistic Regression has test error of 0.8411%, which means that it has predicted 0.8411% of classes of the test data incorrectly.

Linear Discriminant Analysis (LDA)

Our second approach is the Linear Discriminant Analysis.

```
# Linear Discriminant Analysis
# Fit a model to the training data
library(MASS)
lda.model <- lda(wine ~ . - quality, data = df.train)
lda.model
```

```
## Call:
## lda(wine ~ . - quality, data = df.train)
##
## Prior probabilities of groups:
##    0    1
## 0.26 0.74
##
## Group means:
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 0      8.282353        0.5313891   0.2698100       2.534977 0.08765973
## 1      6.842734        0.2800525   0.3346264       5.938935 0.04575167
##   free.sulfur.dioxide total.sulfur.dioxide   density       pH sulphates
## 0            16.08824              47.4448 0.9967428 3.314027 0.6569502
## 1            35.00684             137.8526 0.9937973 3.193164 0.4897615
##    alcohol
## 0 10.42143
## 1 10.57967
##
## Coefficients of linear discriminants:
##                                LD1
## fixed.acidity         4.739163e-01
## volatile.acidity     -2.936711e+00
## citric.acid           7.228158e-01
## residual.sugar        3.881504e-01
## chlorides            -4.325947e+00
## free.sulfur.dioxide  -1.891394e-02
## total.sulfur.dioxide  1.981391e-02
## density              -1.059671e+03
## pH                    1.503007e+00
## sulphates            -6.584189e-01
## alcohol              -1.010802e+00
```

Unfortunately, one of the disadvantages of the LDA approach is that it does not have a good interpretability, thus, we are not able to determine the important variables in the LDA approach.

Now, Let us measure the misclassification of our Logistic Regression on test data.

```
# Test the model on the test data
pred.test.lda <- predict(lda.model, df.test)$class
lda.error <- mean(df.test$wine != pred.test.lda)
lda.error
```

```
## [1] 0.007476636
```

Our LDA model has test error of 0.7476%, which means that it has predicted 0.7476% of classes of the test data incorrectly.

Decision Tree Classifier

The third approach is the Decision Tree Classifier. In this approach, first, we construct a tree based on the "tree" function. Then, in the next step, we consider the size of the Decision Tree (number of Terminal Nodes) as a hyperparameter, and we try to tune and prune the tree for the best and optimal number of terminal nodes. Then, finally, we plot the pruned tree and we measure the Decision Tree misclassification on test data.

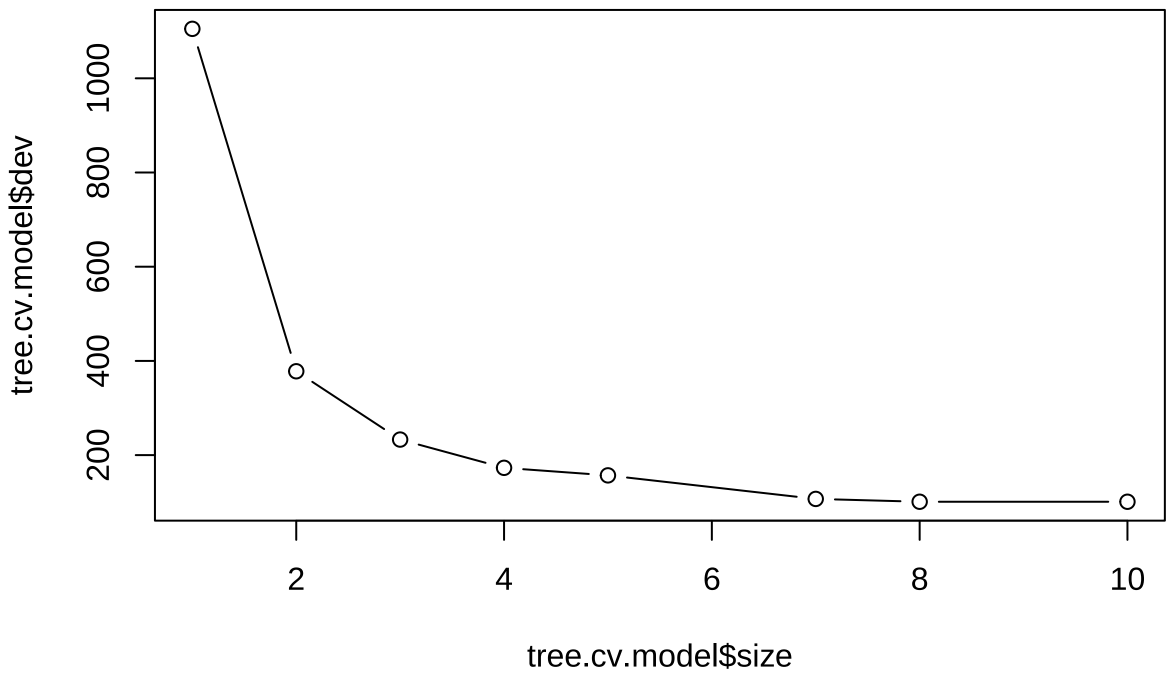
```
# Decision Tree Classifier
library(tree)
tree.model <- tree(wine ~ . - quality, data = df.train)
summary(tree.model)
```

```
##
## Classification tree:
## tree(formula = wine ~ . - quality, data = df.train)
## Variables actually used in tree construction:
## [1] "chlorides"            "total.sulfur.dioxide" "pH"
## [4] "volatile.acidity"     "density"
## Number of terminal nodes:  10
## Residual mean deviance:  0.1191 = 504.9 / 4240
## Misclassification error rate: 0.01788 = 76 / 4250
```

```
# Prune the tree using cross validation
tree.cv.model <- cv.tree(tree.model, FUN = prune.misclass, K = 10)
tree.cv.model
```

```
## $size
## [1] 10  8  7  5  4  3  2  1
##
## $dev
## [1]  101  101  107  157  173  233  378 1105
##
## $k
## [1]  -Inf   0.0   7.0  24.5  31.0  58.0 157.0 727.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(tree.cv.model$size, tree.cv.model$dev, type='b')
```



At above, we determined the optimal size of the tree; we have used K-fold Cross Validation, K=10, which means that we performed the CV with respect to different sizes of the tree. K=10 means that we have equally splitted the train data into 10 parts, and used one part each time as test data and the other 9 parts as train data, then, repeated this for 10 times, and finally, calculating the mean deviance of these 10 parts as deviance of the Cross Validation.

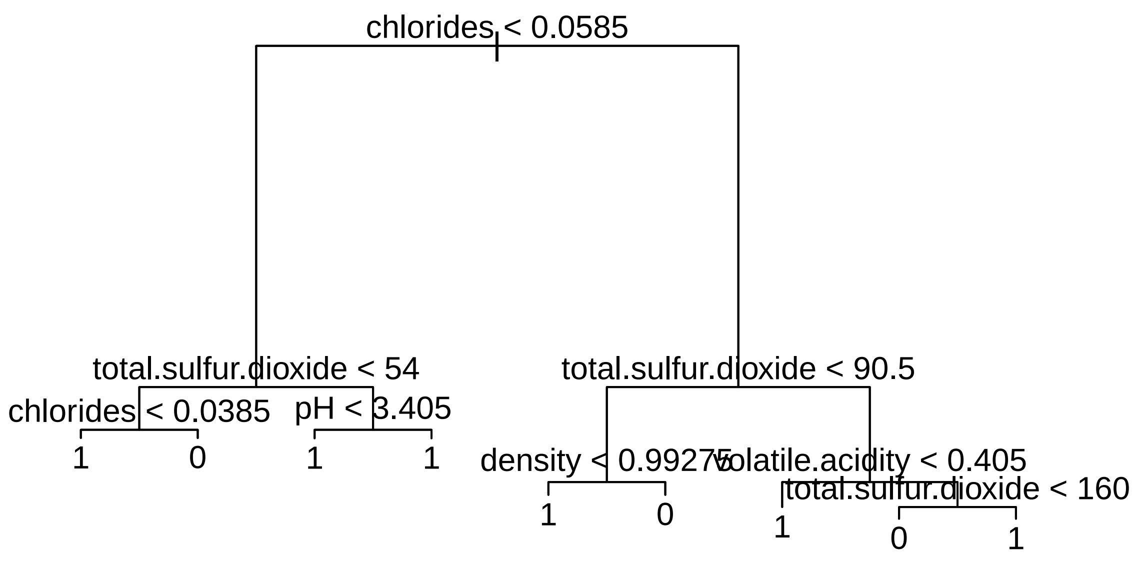
The output shows us that the Decision Tree has the lowest Cross Validation deviance when the size of the tree is equal to 10.

Therefore, we construct a new tree by using size=10.

```
# Prunned Decision Tree
tree.prune.model <- prune.tree(tree.model, best = 9)
summary(tree.prune.model)
```

```
##
## Classification tree:
## snip.tree(tree = tree.model, nodes = 11L)
## Variables actually used in tree construction:
## [1] "chlorides"            "total.sulfur.dioxide" "pH"
## [4] "density"              "volatile.acidity"
## Number of terminal nodes:  9
## Residual mean deviance:  0.1343 = 569.4 / 4241
## Misclassification error rate: 0.01788 = 76 / 4250
```

```r
# Plot the pruned tree
plot(tree.prune.model)
text(tree.prune.model, pretty = 0)
```



Now, Let us measure the misclassification of our Decision Tree on test data.

```r
# Test the model on the test data
pred.test.tree <- predict(tree.prune.model, df.test, type = "class")
tree.error <- mean(pred.test.tree != df.test$wine)
tree.error
```

```
## [1] 0.02616822
```

Our Decision Tree Classifier has test error of 2.6168%, which means that it has predicted 2.6168% of classes of the test data incorrectly. It is obvious that it does not have a superb classification power, but it is great for interpreting and presenting. We can also state that the most important variable in our Decision Tree is at our Root node, which is “chlorides”.

Bagging

The fourth approach is Bagging. Bagging approach is the special case of Random Forest, where we use all of the variables as the number of mtry.

```
# Bagging
library(rpart)
library(ipred)
bag.model <- bagging(wine ~ . - quality, data = df.train, coob = F , nbagg = 100)
bag.model
```

```
##
## Bagging classification trees with 100 bootstrap replications
##
## Call: bagging.data.frame(formula = wine ~ . - quality, data = df.train,
##     coob = F, nbagg = 100)
```

Now, Let us measure the misclassification of our Bagging approach on test data.

```
# Test the model on the test data
pred.test.bag <- predict(bag.model, df.test, type = "class")
bag.error <- mean(pred.test.bag != df.test$wine)
bag.error
```

```
## [1] 0.008411215
```

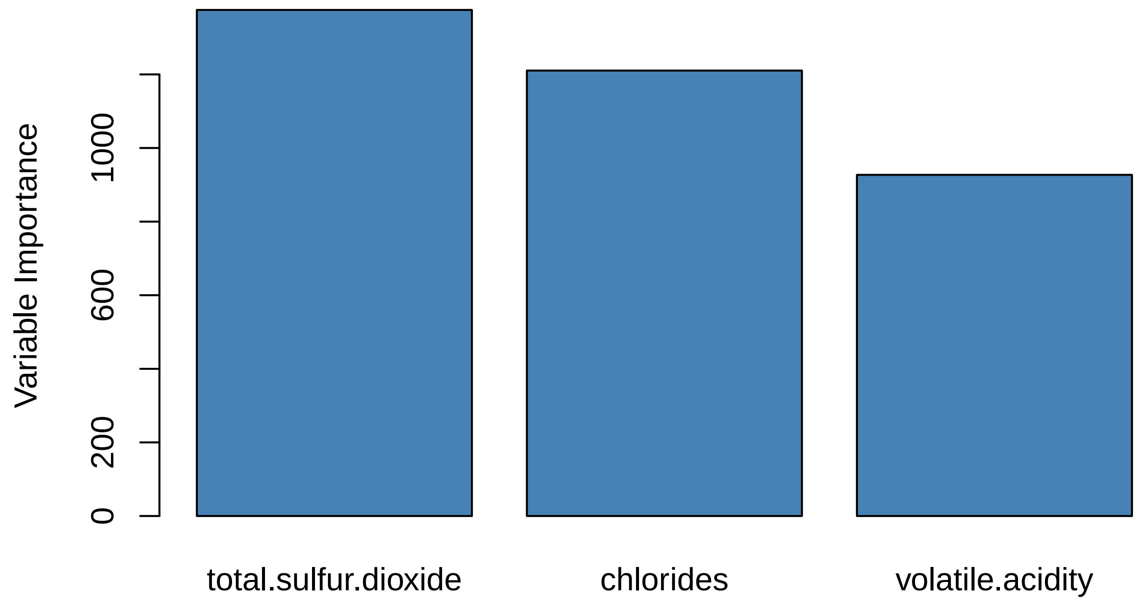
Our Bagging model has test error of 0.8411%, which means that it has predicted 0.8411% of classes of the test data incorrectly.

Now, we want to find out the 3 most important variables in our Bagging model.

```
# Find important variables
library(caret)
vi <- data.frame(var=names(df.train[,-13]), imp=varImp(bag.model))
vi
```

```
##                                       var     Overall
## alcohol                     fixed.acidity   32.733478
## chlorides                volatile.acidity 1210.403429
## citric.acid                   citric.acid   71.353925
## density                    residual.sugar  274.777054
## fixed.acidity                   chlorides  142.047710
## free.sulfur.dioxide   free.sulfur.dioxide  495.246378
## pH                   total.sulfur.dioxide  112.698622
## quality                           density    8.828909
## residual.sugar                         pH  158.507218
## sulphates                       sulphates  432.736867
## total.sulfur.dioxide              alcohol 1375.140357
## volatile.acidity                  quality  926.994031
```

```
vi_plot <- vi[order(vi$Overall, decreasing=TRUE),][0:3,]
barplot(vi_plot$Overall, names.arg=rownames(vi_plot), horiz=F, col='steelblue',
        ylab='Variable Importance')
```



From the table and plot above we can observe that the three most important predictors are "total.sulfur.dioxide", "chlorides", and "volatile.acidity".

Random Forest

Our fifth and final approach is Random Forest. In this approach, first, we construct a Random Forest modelbased on the "randomForest" function. Then, in the next step, we consider the number of the "mtry" (selected variables) as a hyperparameter, and we try to tune our Random Forest for the best and optimal number of "mtry". Then, finally, we train the best Random Forest model and we measure the misclassification on test data.

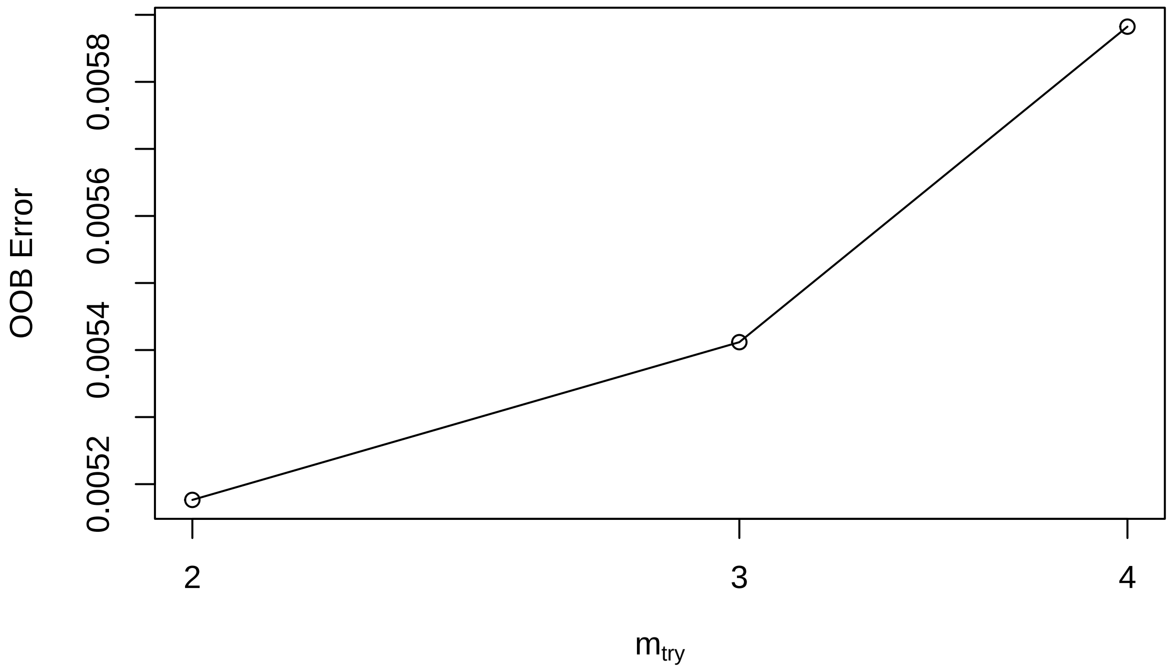
```r
# Random Forest
library(randomForest)
rf.model <- randomForest(wine ~ . - quality, data = df.train, ntree = 100)
rf.model
```

```
##
## Call:
##  randomForest(formula = wine ~ . - quality, data = df.train, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 0.61%
## Confusion matrix:
```

```
##      0    1 class.error
## 0 1084   21 0.019004525
## 1    5 3140 0.001589825
```

```
# Tune the hyperparameter of 'mtry'
mtry <- tuneRF(df.train[-12:-13], df.train$wine, ntreeTry = 1000,
               stepFactor = 1.5,improve = 0.01, trace = TRUE, plot = TRUE)
```

```
## mtry = 3  OOB error = 0.54%
## Searching left ...
## mtry = 2     OOB error = 0.52%
## 0.04347826 0.01
## Searching right ...
## mtry = 4     OOB error = 0.59%
## -0.1363636 0.01
```



```
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
mtry
```

```
##       mtry    OOBError
## 2.OOB    2 0.005176471
## 3.OOB    3 0.005411765
## 4.OOB    4 0.005882353
```

```
best.m
```

```
## [1] 2
```

For this purpose, we use different "mtry" values for and then measure the error with respect to OOB(Out of Bag) data. Note that there is a possibility that some of the observations may not be used for training our Random Forest model, because they are randomly selected, and we call those unspoiled data Out of Bag (OOB).

Here the best number of "mtry" is 2, thus, we construct our best Random Forest model with "mtry" equal to 2.

```
# Build the Random Forest based on the optimal 'mtry'
rf.model.best <- randomForest(wine ~ . - quality, data = df.train, importance = TRUE, mtry = best.m, nt
rf.model.best
```

```
##
## Call:
##  randomForest(formula = wine ~ . - quality, data = df.train, importance = TRUE,      mtry = best.m, 
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.49%
## Confusion matrix:
##      0    1 class.error
## 0 1088   17  0.01538462
## 1    4 3141  0.00127186
```

Now, Let us measure the misclassification of our Random Forest model on test data.

```
# Test the model on the test data
pred.test.rf <- predict(rf.model.best, df.test, type = "class")
rf.error <- mean(pred.test.rf != df.test$wine)
rf.error
```

```
## [1] 0.006542056
```

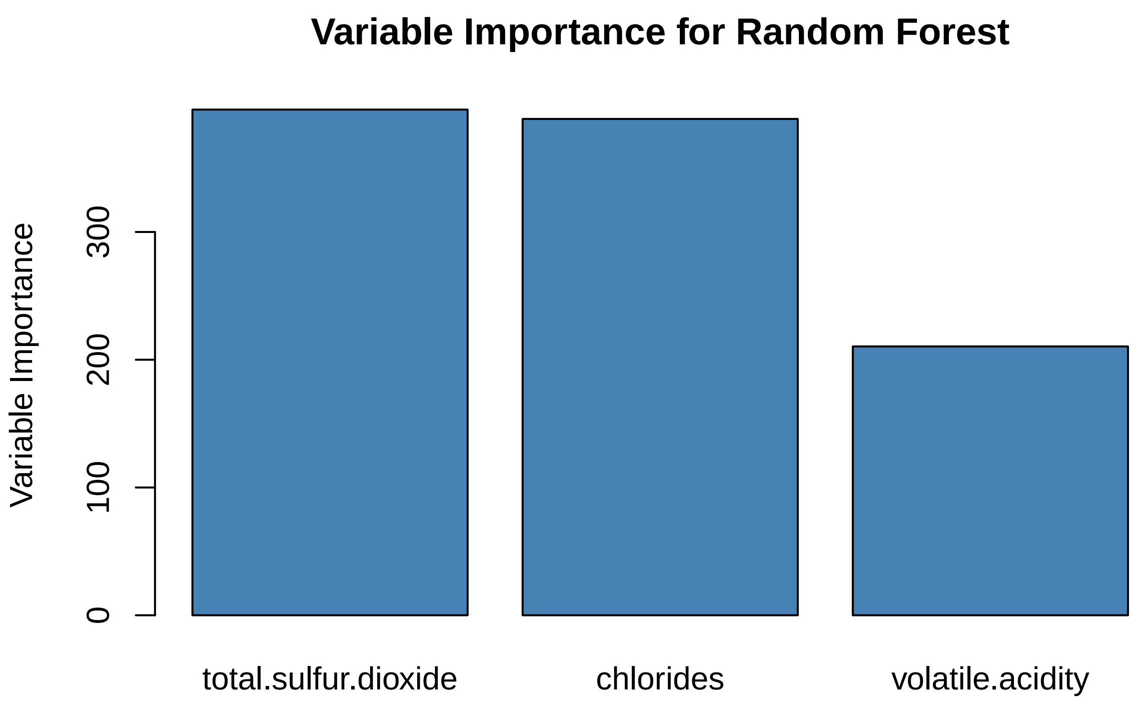
Our LDA model has test error of 0.6542%, which means that it has predicted 0.6542% of classes of the test data incorrectly.

Now, we want to find out the 3 most important variables in our Random Forest model.

```
# Find important variables
importance(rf.model.best)
```

```
##                             0        1 MeanDecreaseAccuracy MeanDecreaseGini
## fixed.acidity        29.83909 28.99534             38.79086         80.98989
## volatile.acidity     42.38209 36.34008             51.48874        210.40077
## citric.acid          27.97392 21.00657             30.41765         47.42808
## residual.sugar       41.05625 22.44327             37.57958        105.73965
## chlorides            66.68248 57.45494             76.23617        395.77480
## free.sulfur.dioxide  21.25771 28.61369             33.69452        105.16039
## total.sulfur.dioxide 72.60223 60.62931             84.22735        388.45772
## density              49.59747 29.33283             48.78423        127.71594
## pH                   32.12725 26.60387             37.01471         44.74634
## sulphates            40.51601 37.57417             47.72813        105.70052
## alcohol              26.32437 21.97704             29.56227         22.88389
```

```
data <- data.frame(importance(rf.model.best)[,4])
barplot(sort(data[,1], decreasing = T)[1:3], horiz=F, col='steelblue',
        ylab='Variable Importance',
        main="Variable Importance for Random Forest",
        names=c("total.sulfur.dioxide", "chlorides", "volatile.acidity"))
```



From the table and plot above we can observe that the three most important predictors are "total.sulfur.dioxide", "chlorides", and "volatile.acidity", with respect to Mean Decrease in Gini Index.

4

In conclusion, we have deployed 5 classification approaches to classify the color of the wine with respect to 11 physicochemical characteristics. We have used Logistic Regression, Linear Discriminant Analysis, Decision Tree, Bagging, and Random Forest approaches to predict the "wine" class, "Red" or "White".

Test Error of Models

We had the following test errors from the mentioned approaches

```r
nerr <- cbind(c("Logistic Regression", "Linear Discriminant Analysis",
                "Decision Tree", "Bagging", "Random Forest"))
err <- cbind(c(logistic.error, lda.error, tree.error, bag.error, rf.error))
err <- round(err * 100, 4)
errors <- cbind(nerr,err)
errors
```

```
##      [,1]                           [,2]
## [1,] "Logistic Regression"          "0.8411"
## [2,] "Linear Discriminant Analysis" "0.7477"
## [3,] "Decision Tree"                "2.6168"
## [4,] "Bagging"                      "0.8411"
## [5,] "Random Forest"                "0.6542"
```

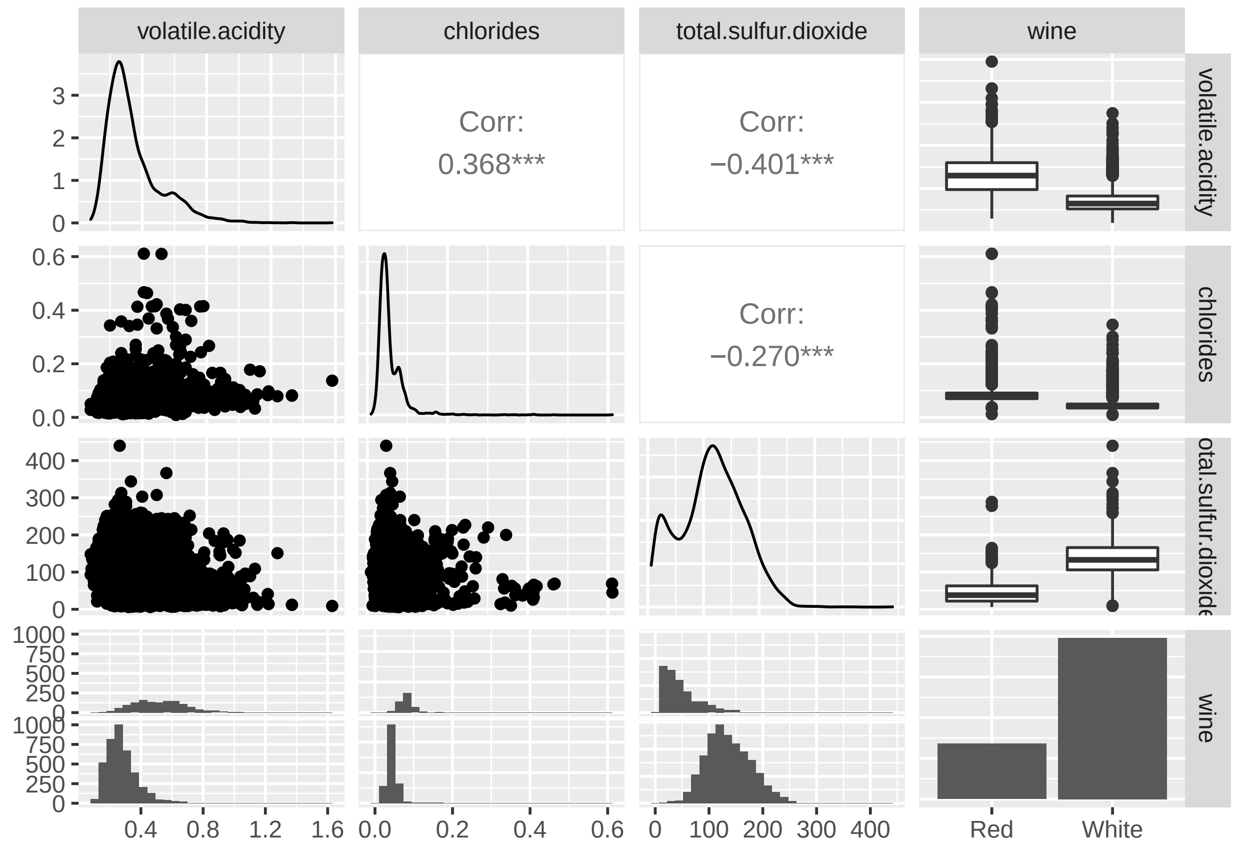
Note that all the values are in terms of (%).

As it can be observed the Random Forest model performed the best among all 5 approaches with only having 0.65% misclassification. However, it would worth mentioning that for presenting purposes, Decision Trees are much better, because they do not require heavy math and it is easy for the audience to follow.

Important Variables

Apart from LDA, the three most variables for all of our models were "total.sulfur.dioxide", "chlorides", and "volatile.acidity". Additionally, it worth to look at the plot below.

```r
# Check the important variables in Decision Tree, Bagging, and Random Forest
library(ggplot2)
library(GGally)
ggpairs(df[,c(2,5,7,13)], progress = F)
```



From the plot above we can observe the distributions of each variable, scatter plot for each pair, correlation of each pair, and the relation of each variable with respect to "wine".