Classifying Portuguese Red Wines

Indrani Mazumdar

March 6, 2018

Part I - Data Preperation

1-1 Data transformation and partition

```
library(caret)
library(ggplot2)
library(tidyverse)
library(dplyr)
library(caret)
library(rpart)
library(rpart.plot)
library(ipred)
library(adabag)
library(randomForest)
library(FNN) library(class)
library(gridExtra)

# read in the original data
wine.orig <- read.csv("C:/Users/indrani/Desktop/My R Work/BUS212/case2-
sp018team-a/Case2_data/winequality-red-v2.csv")



# make the target variable into a factor wine.df
<- wine.orig %>%
  mutate(quality = ifelse(quality %in% c(3,4), "Poor", ifelse(quality %in%
c(5,6), "Normal", "Excellent")))



## Partition the Data ## set.seed(100)

train.index <- sample(c(1:dim(wine.df)[1]), dim(wine.df)[1]*0.8)
train.df <- wine.df[train.index, ] valid.df <- wine.df[-
train.index, ]

train.df$quality <- as.factor(train.df$quality)


valid.df$quality <- as.factor(valid.df$quality)

summary(train.df)
```

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 4.600   Min.   :0.1200   Min.   :0.0000   Min.   : 0.900
##  1st Qu.: 7.100   1st Qu.:0.4000   1st Qu.:0.0900   1st Qu.: 1.900
##  Median : 7.900   Median :0.5200   Median :0.2600   Median : 2.200
##  Mean   : 8.313   Mean   :0.5311   Mean   :0.2709   Mean   : 2.557
##  3rd Qu.: 9.200   3rd Qu.:0.6400   3rd Qu.:0.4300   3rd Qu.: 2.600
##  Max.   :15.900   Max.   :1.5800   Max.   :0.7900   Max.   :15.500
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00
##  Median :0.07900   Median :14.00       Median : 38.00
##  Mean   :0.08754   Mean   :15.96       Mean   : 46.22
##  3rd Qu.:0.09050   3rd Qu.:21.00       3rd Qu.: 62.00
##  Max.   :0.61100   Max.   :72.00       Max.   :289.00
##     density             pH           sulphates         alcohol
##  Min.   :0.9901   Min.   :2.860   Min.   :0.3300   Min.   : 8.40
##  1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##  Median :0.9967   Median :3.310   Median :0.6200   Median :10.20
##  Mean   :0.9967   Mean   :3.312   Mean   :0.6562   Mean   :10.43
##  3rd Qu.:0.9979   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##  Max.   :1.0037   Max.   :4.010   Max.   :1.9800   Max.   :14.90   ##
quality
##  Excellent: 173
##  Normal   :1050
##  Poor     :  56
##
##
##
```

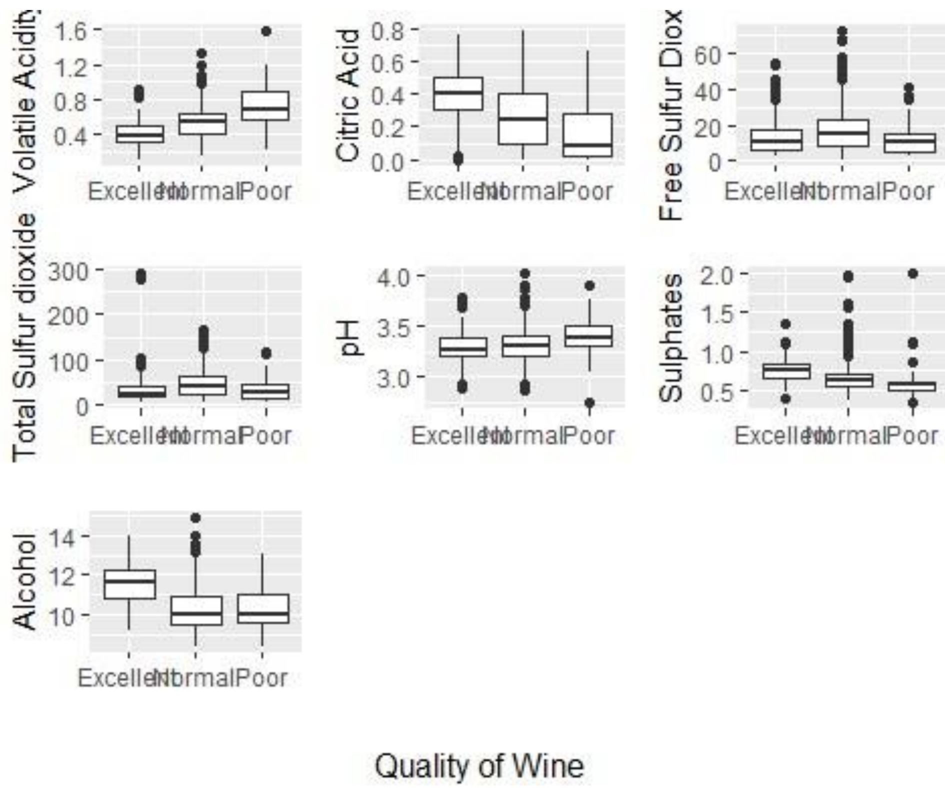
The data set is for understanding the quality of wine, it has a total of 12 variables. The wine quality is changed to character format which is three categories - Excellent - Normal - Poor. Then the data is set.seed to 100. The training set is 80% and validation set is 20%.

1-2 Study of variable correlations

We use Box plot analysis to study each of the variable relationship with Quality and we come to the below results.

```
## Variables Correlation ##

p2 <- ggplot(train.df, aes(x = as.factor(quality), y = volatile.acidity)) +
geom_boxplot()+  xlab("")+
  ylab("Volatile Acidity")
```

```r
p3 <- ggplot(train.df, aes(x = as.factor(quality), y = citric.acid)) +
geom_boxplot()+  xlab("")+
  ylab("Citric Acid")


p6 <- ggplot(train.df, aes(x = as.factor(quality), y = free.sulfur.dioxide))
+
  geom_boxplot() +
xlab("") +
  ylab("Free Sulfur Dioxide")


p7 <- ggplot(train.df, aes(x = as.factor(quality), y =
total.sulfur.dioxide))+  geom_boxplot() +  xlab("") +
  ylab("Total Sulfur dioxide")


p9 <- ggplot(wine.df, aes(x = as.factor(quality), y = pH)) +
geom_boxplot() +  xlab("") +  ylab("pH")

p10 <- ggplot(wine.df, aes(x = as.factor(quality), y = sulphates)) +
geom_boxplot() +  xlab("") +  ylab("Sulphates")

p11 <- ggplot(wine.df, aes(x = as.factor(quality), y = alcohol)) +
geom_boxplot() +  xlab("") +
  ylab("Alcohol")

grid.arrange(p2,p3,p6,p7,p9,p10,p11, bottom = "Quality of Wine")
```




```
# correlation between total sulfur dioxide and free sulfur dioxide
cor(train.df$free.sulfur.dioxide, train.df$total.sulfur.dioxide)

## [1] 0.6702136
```

- Variable Correlation Understanding

The most critical Variables basis their boxplots are:- 2 - volatile acidity 3 - citric acid 7 - total sulfur dioxide * 9 - pH 10 - sulphates 11 - alcohol

(*The correlation between total sulfur dioxide and free sulfur dioxide is 0.67. We chose total sulfur dioxide which has higher variation among different levels of qualities.)

Part II - Model Selection

Basis of our model selection of tree based models

For the Tree models, the four models are based on an understanding as shared below:

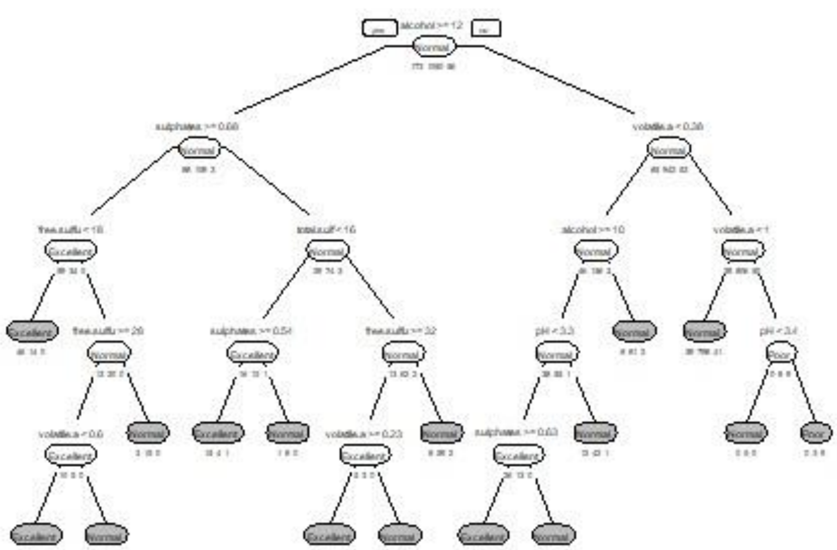
Model 1 - All Variables are taken into consideration for this model and the accuracy and other parameters. Model 2 - Variables taken basis our study of variable correlation study. Model 3 - Variables taken basis the Model 1 variable importance plot.

2-1 Best Pruned Tree

The best pruned tree is the Model 3 using the variables selected by the pruned tree: 2 - volatile acidity 6 - free sulfur dioxide 7 - total sulfur dioxide 9 - pH 10 - sulphates 11 - alcohol

Detailed coding and confusion matrix can be found below.

```
## use the variables selected from the first pruned tree
# 2,6,7,9,10,11 #
select variables
set.seed(1)
train.df.2 <- train.df[, c(2,6,7,9,10,11,12)]
# cross validation
cv.ct <- rpart(quality~., data = train.df.2, method = "class",
cp = 0.000001, minsplit = 4, maxdepth = 5,xval = 5) # use
printcp() to print the table

pruned.ct <- prune(cv.ct,                     cp =
cv.ct$cptable[which.min(cv.ct$cptable[,"xerror"]),"CP"])
length(pruned.ct$frame$var[pruned.ct$frame$var == "<leaf>"])

## [1] 16

prp(pruned.ct, type = 1, extra = 1, under = TRUE, split.font = 1, varlen = -
10,
    box.col=ifelse(pruned.ct$frame$var == "<leaf>", 'gray', 'white'))
```



```
## Model Evaluation ##

# for validation data confusion matrix
class.tree.pred.valid <- predict(pruned.ct,valid.df,type = "class")
confusionMatrix(class.tree.pred.valid, valid.df$quality)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Excellent Normal Poor
##   Excellent        24      9    0
##   Normal           20    260    7
##   Poor              0      0    0
##
## Overall Statistics
##
##                Accuracy : 0.8875
##                  95% CI : (0.8477, 0.92)
##     No Information Rate : 0.8406
##     P-Value [Acc > NIR] : 0.0109
##
##                   Kappa : 0.5148
##  Mcnemar's Test P-Value : NA            ##
## Statistics by Class:
##
##                      Class: Excellent Class: Normal Class: Poor
## Sensitivity                    0.5455        0.9665     0.00000
## Specificity                    0.9674        0.4706     1.00000
## Pos Pred Value                 0.7273        0.9059         NaN
## Neg Pred Value                 0.9303        0.7273     0.97813
## Prevalence                     0.1375        0.8406     0.02187
## Detection Rate                 0.0750        0.8125     0.00000
## Detection Prevalence           0.1031        0.8969     0.00000
## Balanced Accuracy              0.7564        0.7186     0.50000
```

We train three pruned trees and find that Model 3 is the best. The detailed analysis is as follows:

- Model 1 - Pruned tree with all variables

First, we run a pruned tree with all variables using cross validation. The number of split is 10. Testing the performance of the validation data, the overall accuracy is 88.12%. The sensitivity for excellent class is 52.27%, 96.28% for normal class and 0% for poor class. The pruned tree with all variables did a good job predicting normal wines, but a pretty poor job for poor wines.It misclassified all the poor wines.

After investigating the tree, the variables actually used in this model are: 2 - volatile acidity 6 - free sulfur dioxide 7 - total sulfur dioxide 9 - pH 10 - sulphates 11 - alcohol

In our previous analysis, we ruled out free sulfur dioxide due to the its correlation with total sulfur dioxide. However, it was selected by the pruned tree, indicating that it should be important to classify the quality of wines. In the next step, we will train two more models, one with variables selected from boxplots and one with variables selected by the tree.

- Model 2 - Pruned tree with variables selected from boxplots

Using the variables we previously studied with boxplots, the overall accuracy for the performance of validation data improved to 88.44%. The power to predict excellent class decreases a little bit (from 52.27% to 50%), while the performance to predict normal class increased to 97.03%. Again, the model still performed weakly on poor quality wines.

- Model 3 (Best Model) - Pruned tree with variables selected by Model 1

Using variables selected by the first pruned tree, the overall accuracy improved to 88.75%, highest among these three pruned trees. Evaluating the performance on validation data, the model correctly classified 54.55% of excellent wines, 96.65% of normal wine but still 0% of poor wines. We came to the conclusion that the trained model performed very poor toward poor wines, possibly due to the size of the sample. The performance toward excellent wines also raised some concerns. This is the best pruned tree we got with highest accuracy and good sensitivity on validation data.

2-2 Boosted Tree

Variable Importance

- Model 1 - Boosted tree with all variables

Using all the variables, the boosted tree gives the accuracy of 89.06% toward the validation data. The sensitivity is 63.64%, 95.54% and 0% for excellent wines, normal wines and poor wines respectively. The accuracy is not bad for this model, but the prediction of poor wine is still very weak.

Additionally, boosted tree gives us another important information - importance of the variable. As shown below, the first six important variables are:- 2 - volatile acidity 10 - sulphates 11 - alcohol 5 - chlorides 3 - citric acid 7 - total sulfur dioxide

```
# boosted trees with all variables set.seed(1)
boost.tree <- boosting(quality ~ ., data = train.df)

# importance of the variables sort(boost.tree$importance)

##  free.sulfur.dioxide       residual.sugar                   pH
##             6.262846             6.845090             7.294294
##              density        fixed.acidity total.sulfur.dioxide
##             7.403125             7.406770             8.125538
##          citric.acid            chlorides              alcohol
##             8.589495             9.096026            11.636720
##            sulphates     volatile.acidity
##            12.979973            14.360122
```

Best Boosted Tree

- Model 2 (Best Model) - Boosted tree with variables selected from boxplots

```
# with variables selected from the correlation boxplots set.seed(1)
train.df.1 <- train.df[, c(2,3,7,9,10,11,12)]
boost.tree <- boosting(quality ~ ., data = train.df.1)

class.boost.valid <- predict(boost.tree, valid.df, type = "class")

# confusion matrix confusionMatrix(class.boost.valid$class,
valid.df$quality)

## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Excellent Normal Poor
```

```
##   Excellent        29      8    0
##   Normal          15    258    5
##   Poor             0      3    2
##
## Overall Statistics
##
##                Accuracy : 0.9031
##                  95% CI : (0.8653, 0.9332)
##     No Information Rate : 0.8406
##     P-Value [Acc > NIR] : 0.0008254
##
##                   Kappa : 0.6178          ##
Mcnemar's Test P-Value : NA              ##
## Statistics by Class:
##
##                      Class: Excellent Class: Normal Class: Poor
## Sensitivity                   0.65909        0.9591     0.28571
## Specificity                   0.97101        0.6078     0.99042
## Pos Pred Value                0.78378        0.9281     0.40000
## Neg Pred Value                0.94700        0.7381     0.98413
## Prevalence                    0.13750        0.8406     0.02187
## Detection Rate                0.09062        0.8063     0.00625
## Detection Prevalence          0.11563        0.8688     0.01562
## Balanced Accuracy             0.81505        0.7835     0.63806
```

The second boosted tree is the best boosted tree using the variables selected from the boxplots. The accuracy is the highest among the three models, as of 90.31%, and most of the sensitivities and specificities of the three levels of quality are the highest. The most important thing is that this model at least predict some poor wines successfully, with the sensitivity of 28.57% for the poor wines.

Additionally, we also run Model 3 with variables selected by Model 1 to see if there is any improvement.

- Model 3 - Boosted tree with variables selected by Model 1

Using the first six important variables selected from the first boosted tree with all the variables, the third model is supposed to perform better. Surprisingly, this model did not perform very well compared to the second one with variables selected from boosted trees. The accuracy is only 87.5% and other parameters also decrease. Detailed information can be found in the wine performance table.

2-3 Bagged Tree

Variable Importance

```
# with all variables set.seed(1)
bagged.tree <- bagging(formula = quality ~ ., data = train.df, coob = TRUE)
sort(bagged.tree$importance)

##            chlorides              density                   pH
##             3.029653             3.347527             3.810941
##  free.sulfur.dioxide       residual.sugar        fixed.acidity
##             4.522323             5.724002             5.877959
##          citric.acid total.sulfur.dioxide            sulphates  ##
6.162998             7.239533             11.750442
##     volatile.acidity              alcohol
##            15.167130            33.367490
```

- Model 1 - Bagged tree with all variables

With all the variables, the trained model gives the accuracy of 87.19%. Again, the model performs well for the normal wines in the validation set, with sensitivity of 97.77%, but the specificity of normal wine is a little bit low, as of 31.37%. The Model performs poor toward excellent wines and poor wines, with sensitivity of 36.36% and 0% respectively.

The variables selected basis the variable importance are: 11 - alcohol 2 - volatile acidity 10 - sulphates 7- total sulfur dioxide 3 - citric acid 1 - fixed acidity

Best Bagged Tree

The best bagged tree (Model 3) is the one with the variables selected by the importance shown in the Model 1.

The selection of variables is as follows: 11 - alcohol 2 - volatile acidity 10 - sulphates 7- total sulfur dioxide 3 - citric acid 1 - fixed acidity

Detailed coding and confusion matrix can be found below.

```
# with variables from first Bagged Tree set.seed(1)
train.df.4 <- train.df[ , c(1,2,3,7,10,11,12)]
bagged.tree <- bagging(formula = quality ~ ., data = train.df.4, coob = TRUE)

class.bag.valid <- predict(bagged.tree, valid.df, type = "class")
confusionMatrix(class.bag.valid$class, valid.df$quality)

## Confusion Matrix and Statistics
##
##            Reference
```

```
## Prediction  Excellent Normal Poor
##   Excellent        19      7    0
##   Normal           25    262    7
##   Poor              0      0    0
##
## Overall Statistics
##
##                Accuracy : 0.8781
##                  95% CI : (0.8372, 0.9119)
##     No Information Rate : 0.8406
##     P-Value [Acc > NIR] : 0.03612
##
##                   Kappa : 0.4371          ##
Mcnemar's Test P-Value : NA              ##
## Statistics by Class:
##
##                      Class: Excellent Class: Normal Class: Poor
## Sensitivity                   0.43182        0.9740     0.00000
## Specificity                   0.97464        0.3725     1.00000
## Pos Pred Value                0.73077        0.8912         NaN
## Neg Pred Value                0.91497        0.7308     0.97813
## Prevalence                    0.13750        0.8406     0.02187
## Detection Rate                0.05937        0.8187     0.00000
## Detection Prevalence          0.08125        0.9187     0.00000
## Balanced Accuracy             0.70323        0.6733     0.50000
```

The detailed comparison between Model 2 and Model 3 is as follows:

- Model 2 - Boosted tree with variables selected by boxplot studies

Using the variables selected from the studies of the boxplots, the performance improves a little bit but still very poor. The accuracy is 87.5% and the parameters for excellent and poor wines are not satisfied.

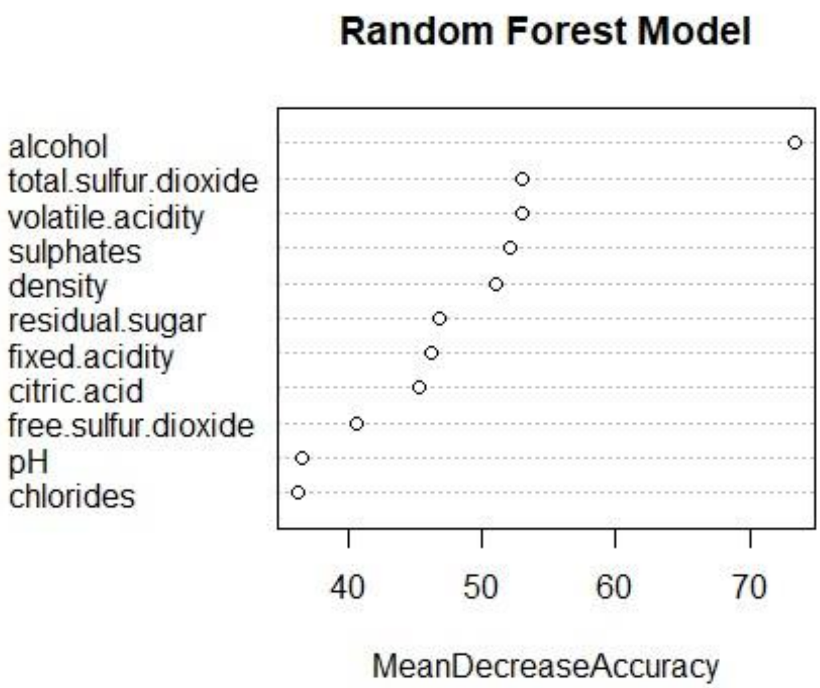
- Model 3 (Best Model) - Boosted tree with variables selected by Model 1

Using the variables selected by the first model, the sensitivities and specificities are almost the same compared to the second one. The overall accuracy increases slightly to 87.81%. All the three bagged trees don’t perform very well, but Model 3 is the best one among these three models.

2-4 Random Forest

Variable Importance

```
# with all the variables set.seed(1)
rf <- randomForest(as.factor(quality) ~ ., data = train.df, ntree = 2000,

                    mtry = 3, nodesize = 5, importance = TRUE)

## variable importance plot varImpPlot(rf, type = 1,main = "Random Forest
Model")
```



- Model 1 - Random forest with all the variables

By nature Random Forest is the highest accuracy Tree format in R , we take the ntree = 2000 , mtry= 3 (as the mtry is a square root of the total variables) , and basis the plot we get an accuracy of 89.69% with sensitivity and specificity as mentioned in the model performance csv file.

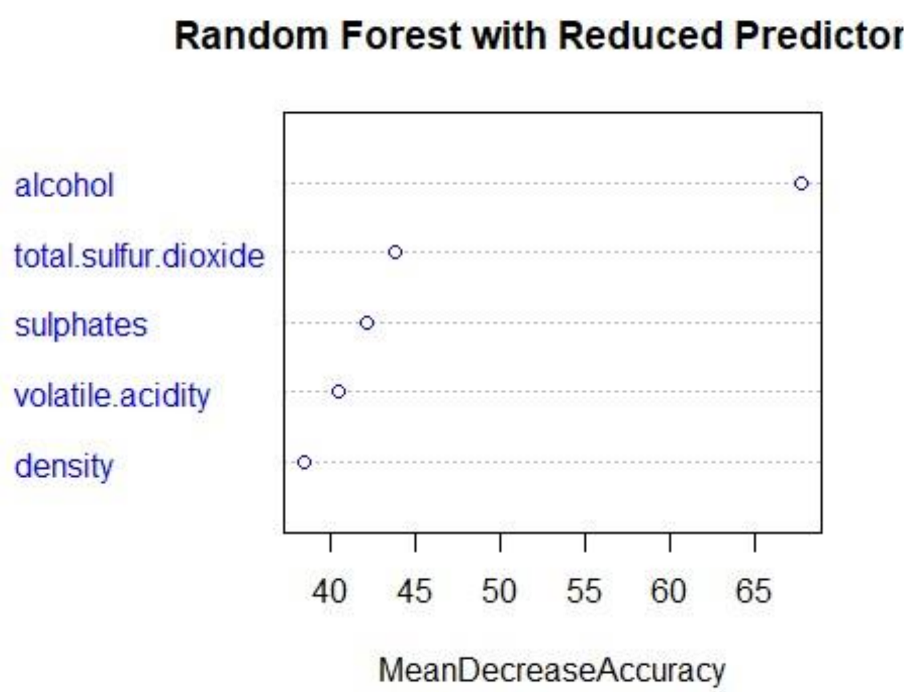
The variable importance shows ranking for first four predictors as: 11- Alcohol 2 - Volatile Acidity 7 - Total Sulfur Dioxide 10 - Sulphates.

The OOB error rate is 14.46% , which is just calculated by using the OOB code in R.

Best Random Forest Tree

The third RF model (Model 3) is the best Random Forest tree in terms of accuracy , specificity and sensitivity for the variables calculated in through Confusion matrix.

```r
#random forest 2 with second variable reduction basis variable Importance
set.seed(1)

train.df.mod1 <-train.df[ , c(2,7,8,10,11,12)]
valid.df.mod1 <- valid.df[ , c(2,7,8,10,11,12)] rf2 <-
randomForest(quality ~., data = train.df.mod1, ntree = 1000,
mtry = 4, nodesize = 5, importance = TRUE)

varImpPlot(rf2, type = 1,main = "Random Forest with Reduced Predictors",color
="Blue")
```



```r
#confusion matrix for the rf2  rf2.pred <-
predict(rf2, valid.df.mod1)
confusionMatrix(rf2.pred, valid.df.mod1$quality)
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction  Excellent Normal Poor
##   Excellent        26      4    0
##   Normal           18    265    7
##   Poor              0      0    0 ##
## Overall Statistics
##
##                Accuracy : 0.9094
##                  95% CI : (0.8724, 0.9385)
##     No Information Rate : 0.8406
##     P-Value [Acc > NIR] : 0.0002342
##
##                   Kappa : 0.5977
##  Mcnemar's Test P-Value : NA              ##
## Statistics by Class:
##
##                        Class: Excellent Class: Normal Class: Poor
## Sensitivity                    0.59091        0.9851     0.00000
## Specificity                    0.98551        0.5098     1.00000
## Pos Pred Value                 0.86667        0.9138         NaN
## Neg Pred Value                 0.93793        0.8667     0.97813
## Prevalence                     0.13750        0.8406     0.02187
## Detection Rate                 0.08125        0.8281     0.00000
## Detection Prevalence           0.09375        0.9062     0.00000
## Balanced Accuracy              0.78821        0.7475     0.50000
```

```r
#OOB Estimate for rf2 err2
<- rf2$err.rate head(err2)
```

```
##            OOB Excellent     Normal      Poor
## [1,] 0.2161017 0.5254237 0.12239583 0.8275862
## [2,] 0.2095491 0.5638298 0.12038523 0.8108108
## [3,] 0.2070438 0.5280000 0.12353706 0.7674419
## [4,] 0.1890130 0.5034965 0.10407240 0.8297872
## [5,] 0.1972318 0.5414013 0.10642782 0.8400000 ##
[6,] 0.1798501 0.5123457 0.08915907 0.8653846
```

```r
oob_err2 <- err2[nrow(err2), "OOB"] print(oob_err2)
```

```
##       OOB
## 0.1422987
```

- Model 3 (Best Model) - Random Forest with selected variables based on Model 1

The last Tree plotted in Random Forest is basis the Variable Importance plot for the First RF , the total variables chosen are five: 2 - Volatile Acidity 7 - Total Sulfur Dioxide 8 - Density 10 -Sulphates 11 - Alcohol.

Total Accuracy is 90.94 % with sensitivity and specificity for excellent and normal category the highest out of the three models. OOB error rate is 14.62%.

- Model 2 - Random Forest with selected variables based on boxplots

For a deeper and accurate understanding , a new Random Forest is plotted with variables chosen as per the Variable selection basis the Boxplots . Total of 7 variables are chosen and the ntree =2000 and mtry =3 . We get an accuracy of 89.38% and specificity and sensitivity increases when compared to the previous RF plot. The variable importance plot shows: 11 Alcohol 2 - Volatile Acidity 10 -Sulphates 7 -Total Sulfur Dioxide , which is different from the previous plot.

The OOB error rate is 14.62%

2-5 KNN model

In the first step, we build a knn model using all variables with k=1. In our KNN Model 1, accuracy is 81.56%, which needs improvement. After taking a look into different classes of prediction, we find out that Normal class has the highest sensitivity of 89.22%, while Excellent class has 43.18% accuracy and Poor class has 28.57% accuracy. Thus, KNN Model 1 predicts relatively well in Normal class but does not perfrom well in Poor class. Next, we try to identify the best k values to improve our model performance.

Find the best K with highest accuracy

```
#Normalize Data normalize <- function(x){return((x-
min(x))/(max(x)-min(x)))}

wine.df.nor <- wine.df

for (i in 1:11) {wine.df.nor[, i] <- normalize(wine.df[, i])}

# partition the normalized data set.seed(100)

train.index <- sample(c(1:dim(wine.df.nor)[1]), dim(wine.df.nor)[1]*0.8)
train.df.nor <- wine.df[train.index, ] valid.df.nor <- wine.df[-
train.index, ]



#Separate data set set.seed(1)

#Select the best k value accuracy.df <-
data.frame(k=seq(1,30,1),accuracy=rep(0,30))

for (i in 1:30){knn.pred <- knn(train.df.nor[,1:11],
valid.df.nor[,1:11],
cl=train.df$quality,k=i)                    accuracy.df[i,2]<-
confusionMatrix(knn.pred,valid.df.nor$quality)$overall[1]
}

accuracy.df

##     k accuracy
## 1   1 0.815625
## 2   2 0.803125
## 3   3 0.828125
## 4   4 0.825000
## 5   5 0.831250
```

```
## 6    6 0.821875
## 7    7 0.831250
## 8    8 0.831250
## 9    9 0.831250
## 10 10 0.834375
## 11 11 0.834375
## 12 12 0.840625
## 13 13 0.834375
## 14 14 0.831250
## 15 15 0.843750
## 16 16 0.843750
## 17 17 0.840625
## 18 18 0.840625
## 19 19 0.846875
## 20 20 0.850000
## 21 21 0.853125
## 22 22 0.846875
## 23 23 0.843750
## 24 24 0.846875
## 25 25 0.846875
## 26 26 0.846875
## 27 27 0.846875
## 28 28 0.846875
## 29 29 0.846875 ##
30 30 0.843750
```

```
summary(accuracy.df)
```

```
##        k            accuracy
##  Min.   : 1.00   Min.   :0.8031
##  1st Qu.: 8.25   1st Qu.:0.8313
##  Median :15.50   Median :0.8406
##  Mean   :15.50   Mean   :0.8376
##  3rd Qu.:22.75   3rd Qu.:0.8469
##  Max.   :30.00   Max.   :0.8531
```

According to previous results, we got the best accuracy of 85.31% when k=21. So we will build our KNN Model 2 using k=21.

Best KNN model - Build a KNN model using k=21 with all the variables

```
set.seed(1)
quality_pred <- knn(train.df.nor[,1:11],valid.df.nor[,1:11],
cl=train.df.nor$quality,k=21) quality_actual <- valid.df.nor$quality
confusionMatrix(data=quality_pred, reference = quality_actual)
```

```
## Confusion Matrix and Statistics
##
```

```
##             Reference
## Prediction  Excellent Normal Poor
##   Excellent         4      0    0
##   Normal           40    269    7
##   Poor              0      0    0
##
## Overall Statistics
##
##                Accuracy : 0.8531
##                  95% CI : (0.8095, 0.89)
##     No Information Rate : 0.8406
##     P-Value [Acc > NIR] : 0.3009
##
##                   Kappa : 0.1266        ##
Mcnemar's Test P-Value : NA            ##
## Statistics by Class:
##
##                      Class: Excellent Class: Normal Class: Poor
## Sensitivity                   0.09091       1.00000     0.00000
## Specificity                   1.00000       0.07843     1.00000
## Pos Pred Value                1.00000       0.85127         NaN
## Neg Pred Value                0.87342       1.00000     0.97813
## Prevalence                    0.13750       0.84062     0.02187
## Detection Rate                0.01250       0.84062     0.00000
## Detection Prevalence          0.01250       0.98750     0.00000 ##
Balanced Accuracy             0.54545       0.53922     0.50000
mean(quality_actual == quality_pred)

## [1] 0.853125
```

After setting k=21, the accuracy of KNN Model 2 goes up to 85.31%, with sensitivity of Normal class increased to 100% and sensitivity of Excellent class, while sensitivity of Poor class drop dramatically to 0. In other words, KNN Model 2 perfectly predicts Normal class but fails to identify Poor class. We will try to select some variables to see if we can increase our model's performance.

Variable selection

From our prior variable study, we plot boxplots for each variable and quality of wine and find out that fixed acidity , density, residual sugar, chlorides and free sulfur dioxide play less important role in determining wine's quality. So we decided to drop these variables and to see how the new combination of variables will improve our model performance.

In our KNN Model 3, unfortunately the accuracy does not improve. It turns out to be 82.19%. Sensitivities of Excellent and Normal classes dropped and sensitivity of Poor class stays at 0. So we decided not to use KNN Model 3.

Reflections on KNN model

The best KNN model we developed in predicting wine quality is KNN Model 2 with k = 21 and all the variables. We noticed that as overall accuracy improved, the sensitivity of Poor Class decreased. What's worse, the sensitivity for Poor class drops to 0 in our KNN Model 2, which means that the model fails to predict Poor class. Since the poor class has relatively small number of observations, predictions might be disturbed by outliers. Generally speaking, KNN model does not perform well in predicting for wine quality in this dataset. The highest accuracy we got for KNN model is 85.31%, with all variables and with k=21. The model does a great job in predicting Normal class, but fails to predict Poor class.

Part III - Conclusion

On understanding the wine data quality we can understand that the data has 11 predictors which predict the quality of Wine. We can conclude our understanding into two parts :-

1. The Best Model selected

The best model is basis the Trees and KNN and the Confusion matrix parameters - Accuracy , Sensitivity and Specificity. The models and the parameters are shared on the performance csv file. Basis the values we conclude to select Boosted Tree - Model 2 as our best indicator for the Quality of Wine. This tree model has an accuracy of 90.31% which is second highest, only less than the Model 3 for Random Forest as of 90.94%. The reason to prefer this model is because the boosted tree has more variables as predictors as compared to Model 3 for Random Forest.

Another important reason to choose boosted tree-model 2 is that the sensitivity of poor wines is 28.57%, whereas other predictions only give 0% correct prediction for poor wines. This model at least predict some poor wines successfully and this is an important plus.

Sensitivity Parameter for Model 2 - Boosted is higher for all the segments (Excellent - Normal - Poor) as compared to other Models

Specificity Parameter for Model 2 - Boosted is higher for all segments (Excellent - Normal - Poor) as compared to other Models, with segment Poor showing some changes from 1.

2. 4 Best Predictors for the Quality of Wine

The Tree models - Prune, bagged, boosted and Random Forest give us various accuracies and plots with variable importance. After studying the plot for each model- prune tree, boosted, bagged, we can conclude that the four best Predictors for the dataset are: 10 - Sulphates 2 - Volatile Acidity 11 - Alcohol 7 - Total Sulfur Dioxide

Appendix - Model Performance Comparision

```r
performance <- read.csv("C:/Users/indrani/Desktop/My R Work/BUS212/Data/Wine Model Performance.csv")
print(performance)
```

```
##                                   Models  Accuracy Sensitivity_Ex
## 1                       Best Pruned Tree  Accuracy Sensitivity_Ex
## 2                                Model 1    0.8812         0.5227
## 3                                Model 2    0.8844            0.5
## 4                                Model 3    0.8875         0.5455
## 5                     Boosted Tree Model  Accuracy Sensitivity_Ex
## 6                                Model 1    0.8906         0.6364
## 7                                Model 2    0.9031         0.6591
## 8                                Model 3     0.875         0.5909
## 9                     Bagged Tree Model  Accuracy  Sensitivity_Ex
## 10                              Model 1     0.8719         0.3636
## 11                               Model 2     0.875         0.4318
## 12                               Model 3    0.8781         0.4318
## 13                   Random Forest Model Accuracy  Sensitivity_Ex
## 14                              Model 1     0.8969         0.5455
## 15                               Model 2    0.8938         0.5455
## 16                               Model 3    0.9094         0.5909
## 17                                   KNN Accuracy  Sensitivity_Ex
## 18                        Model 1 (k=1)     0.8156         0.4318
## 19                       Model 2 (k=21)     0.8531         0.0909
## 20 Model 3 (k=21) with selected variables    0.8219        0.29545
##    Sensitivity_Nor Sensitivity_P Specificity_Ex Specificity_Nor
## 1  Sensitivity_Nor Sensitivity_P Specificity_Ex Specificity_Nor
## 2           0.9628             0         0.9638           0.451
## 3           0.9703             0         0.9746          0.4314
## 4           0.9665             0         0.9674          0.4706
## 5  Sensitivity_Nor Sensitivity_P Specificity_Ex Specificity_Nor
## 6           0.9554             0         0.9601           0.549
## 7           0.9591        0.2857          0.971          0.6078
## 8           0.9442             0         0.9601          0.5098
## 9  Sensitivity_Nor Sensitivity_P Specificity_Ex Specificity_Nor
## 10          0.9777             0         0.9783          0.3137
## 11          0.9703             0          0.971          0.3725
## 12           0.974             0         0.9746          0.3725
## 13 Sensitivity_Nor Sensitivity_P Specificity_Ex Specificity_Nor
## 14          0.9777             0         0.9783          0.4706
## 15           0.974             0         0.9746          0.4706
## 16          0.9851             0         0.9855          0.5098
## 17 Sensitivity_Nor Sensitivity_P Specificity_Ex Specificity_Nor
## 18          0.8922        0.2857         0.9239          0.4314
## 19               1             0              1          0.0784 ##
20          0.9294             0         0.9384          0.2549 ##
Specificity_P
```

```
## 1  Specificity_P
## 2              1
```

```
## 3         0.9968
## 4              1
## 5  Specificity_P
## 6         0.9968
## 7         0.9904
## 8         0.9872
## 9  Specificity_P
## 10             1
## 11             1
## 12             1
## 13 Specificity_P
## 14             1
## 15             1
## 16             1
## 17 Specificity_P
## 18        0.9713
## 19             1
## 20        0.9936
```