IST 707 Group Project: Wine Quality Prediction

Omar Hanif, Elizabeth Westbrook

## Load packages

library(ggplot2)  
library(dplyr)  
library(gridExtra)  
library(RColorBrewer)  
library(randomForest)  
library(party)  
library(ggthemes)  
library(corrplot)  
library(caret)  
library(kernlab)  
library(e1071)  
library(rattle)  
library(rpart)  
library(klaR)  
library(naivebayes)  
library(ISLR)  
library(gbm)  
library(bnclassify)  
theme\_set(theme\_wsj())

# Introduction

Wine quality is assessed based on more than just flavor; experts use the range of their senses to do determine the quality of wine. These experts rate a wine’s quality on a scale of 1-10. Quality ratings are a popular feature when choosing a wine to purchase. However, quality ratings can also assist vintners and enologists compare their wines not only to other wines/wineries but also to their own previous vintages.  
Because wine quality ratings are subjective, enologists would be well-served by having objective measures that can help determine a wine’s quality, and with the quality determined, they can take steps to improve the quality.

# Analysis

## About the Data

In this data set, there are 4,900 observations of 11 physically measurable attributes of wine. This project examines these attributes of white wines in order to determine the predictability of quality with any combination of attributes. The scope is limited to Portuguese white wines and no record of the names of the assessed wines is available. The data set has been obtained from the following link: <https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/>

Variables in the data set Input variables: 1 - fixed acidity (tartaric acid - g / dm^3) 2 - volatile acidity (acetic acid - g / dm^3) 3 - citric acid (g / dm^3) 4 - residual sugar (g / dm^3) 5 - chlorides (sodium chloride - g / dm^3 6 - free sulfur dioxide (mg / dm^3) 7 - total sulfur dioxide (mg / dm^3) 8 - density (g / cm^3) 9 - pH 10 - sulphates (potassium sulphate - g / dm3) 11 - alcohol (% by volume) Output variable (based on sensory data): 12 - quality (score between 0 and 10)

The data set is complete; there are no missing values. There are no obviously incorrect values; outliers are not too far outside the ranges of each attribute. Quality is a discrete variable while all the input variables are continuous; however, Quality requires discretization. Here it is discretized into two bins: “good” and “bad.”

## Load the data

## Explore

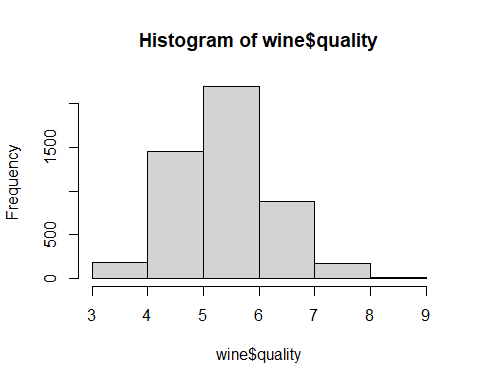
summary(wine)

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

# EDA

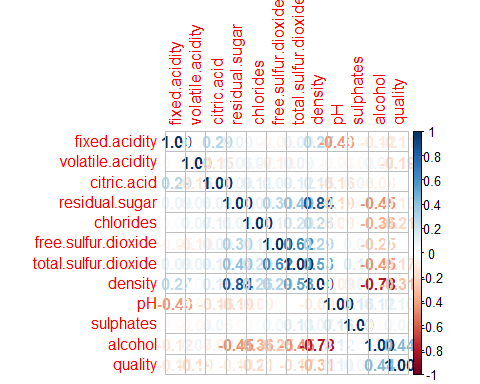
The histogram plot below displays the range of wine quality in the data set. Here, 6.5-7 is typically set as the minimum value for “Good” quality. It can, however, be observed that the majority of the wines 80% have scored a quality value below 7.

hist(wine$quality, breaks = c(3:9))

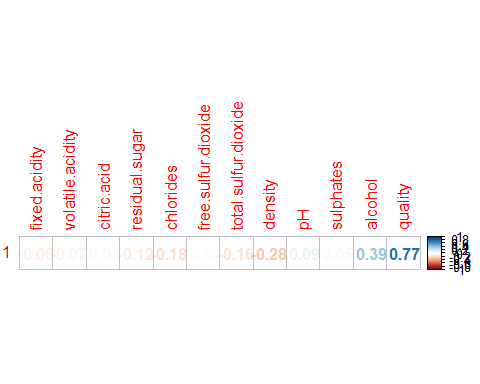


## Correlation

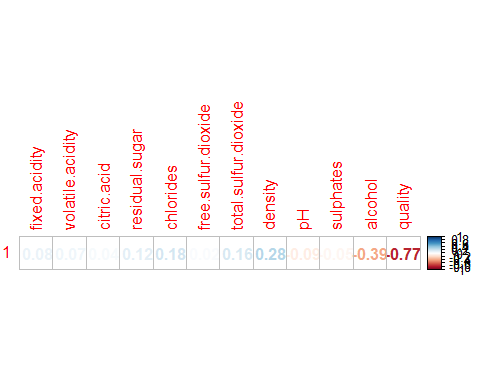
mar = c(9, 5, 9, 5)  
M <- cor(wine)  
corrplot(M, method = "number")



N <- cor(wine$quality >= 7, wine)  
corrplot(N, method = "number")



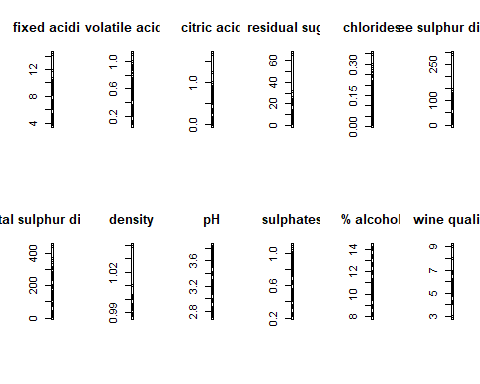
O <- cor(wine$quality < 7, wine)  
corrplot(O, method = "number")



## Discussion: Correlation among variables

The two strongest correlations with quality are moderate positive correlation to alcohol and moderate negative correlation with density.  
Additionally, Alcohol is negatively correlated with density of wine. Density is strongly positively correlated with residual sugar quantity and moderately correlated with pH. Free sulfur dioxide and total sulfur dioxide are strongly correlated.

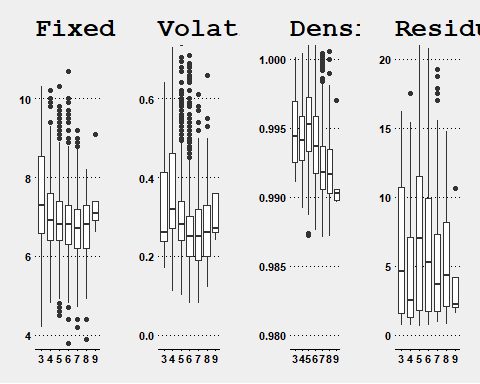
## Individual attribute exploration



## Boxplots of variable correlation with quality

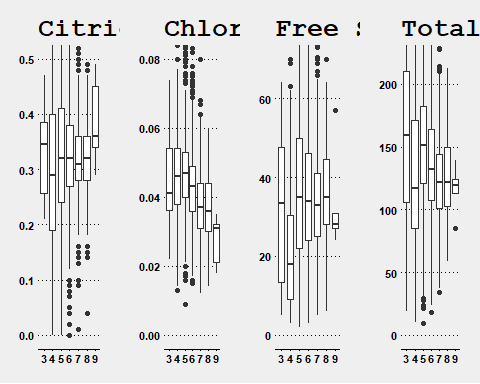
### Acidity: Fixed Acidity, Volatile Acidity, Density, Residual Sugar Plus Quality

grid.arrange(p1 + theme\_wsj(base\_size = 8, color = "gray"),p2+ theme\_wsj(base\_size = 8, color = "gray"), p8 + theme\_wsj(base\_size = 8, color = "gray"),p4 + theme\_wsj(base\_size = 8, color = "gray"), nrow=1)



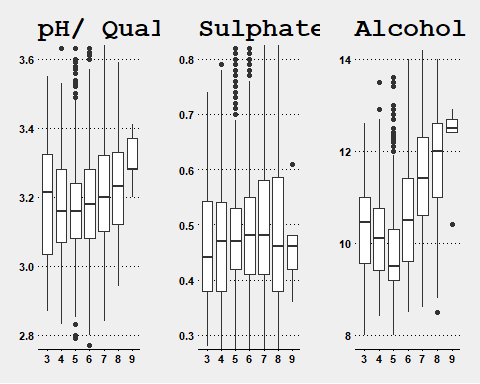
### Citric Acid, Chlorides, Free Sulphur Dioxide, Total Sulphur Dioxide Plus Quality

grid.arrange(p3 + theme\_wsj(base\_size = 8, color = "gray"),p5+ theme\_wsj(base\_size = 8, color = "gray"), p6 + theme\_wsj(base\_size = 8, color = "gray"),p7 + theme\_wsj(base\_size = 8, color = "gray"), nrow=1)



### pH, Sulphates, Alcohol Content

grid.arrange(p9 + theme\_wsj(base\_size = 8, color = "gray"),p10+ theme\_wsj(base\_size = 8, color = "gray"), p11 + theme\_wsj(base\_size = 8, color = "gray"), nrow=1)



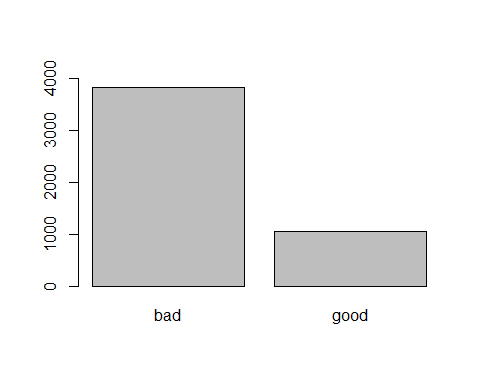
## Discretize Quality

### Bins: Good, Bad

wineClass <- wine  
wineClass$rating <- as.factor(ifelse(wineClass$quality >= 7, "good", "bad"))  
wineClass <- dplyr::select(wineClass, -quality)  
head(wineClass)

## 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 7.2 0.23 0.32 8.5 0.058  
## 6 8.1 0.28 0.40 6.9 0.050  
## 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 47 186 0.9956 3.19 0.40 9.9  
## 6 30 97 0.9951 3.26 0.44 10.1  
## rating  
## 1 bad  
## 2 bad  
## 3 bad  
## 4 bad  
## 5 bad  
## 6 bad

plot(wineClass$rating, ylim= c(0, 4000))



# Methods and Models

All models are run three times with 10-fold cross validation. Preprocessing is done with each train function call. Scale and center are the methods used. The train set contains 75% of the data; the remaining 25% is the test set.

## set train and test sets

set.seed(424)  
  
train\_index <- createDataPartition(wineClass$rating, p= 0.75, list=FALSE)  
  
WC\_train <- wineClass[train\_index, ]  
WC\_test <- wineClass[-train\_index, ]  
  
table(WC\_train$rating)

##   
## bad good   
## 2879 795

table(WC\_test$rating)

##   
## bad good   
## 959 265

# Decision Tree

Decision trees are for classification problems and use training and test data. It recursively splits until a stop condition is met.Stop conditions include: when all data points belong to same class, all records have same attribute values, or model control parameters for pruning have been met.

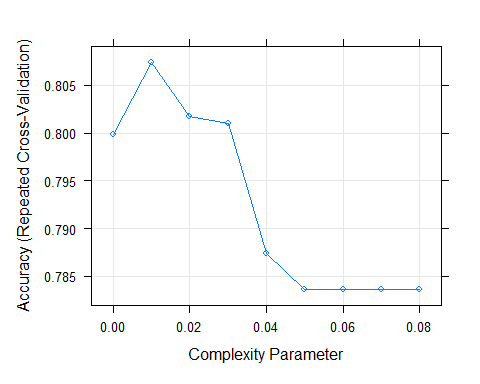
model\_DT1 <- train(rating ~. , data=WC\_train, method = "rpart",  
 preProcess = c("center", "scale"),  
 control = rpart.control(minsplit =200, maxdepth = 15),  
 tuneGrid = expand.grid(cp= seq(0, .08, 0.01)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_DT1

## CART   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3306, 3306, 3307, 3307, 3307, 3307, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00 0.7998532 0.28727427  
## 0.01 0.8073906 0.26671717  
## 0.02 0.8017692 0.24257058  
## 0.03 0.8010384 0.22251877  
## 0.04 0.7874270 0.09553875  
## 0.05 0.7836160 0.00000000  
## 0.06 0.7836160 0.00000000  
## 0.07 0.7836160 0.00000000  
## 0.08 0.7836160 0.00000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.01.

predict\_DT1 <- predict(model\_DT1, newdata = WC\_test, type = "raw")  
confusionMatrix(predict\_DT1, WC\_test$rating, positive= "good")

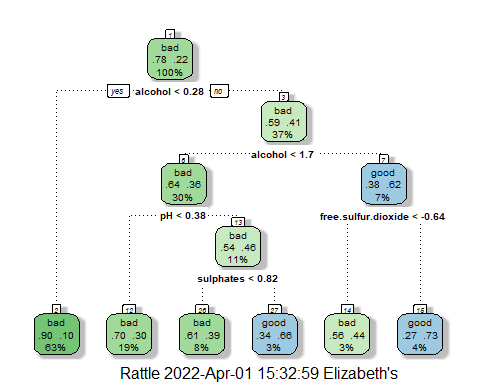
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 925 214  
## good 34 51  
##   
## Accuracy : 0.7974   
## 95% CI : (0.7738, 0.8196)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.1256   
##   
## Kappa : 0.2082   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.19245   
## Specificity : 0.96455   
## Pos Pred Value : 0.60000   
## Neg Pred Value : 0.81212   
## Prevalence : 0.21650   
## Detection Rate : 0.04167   
## Detection Prevalence : 0.06944   
## Balanced Accuracy : 0.57850   
##   
## 'Positive' Class : good   
##

plot(model\_DT1)



plot(predict\_DT1, ylim= c(0, 1200), ylab = "Density", main = "Decision Tree Density Plot")

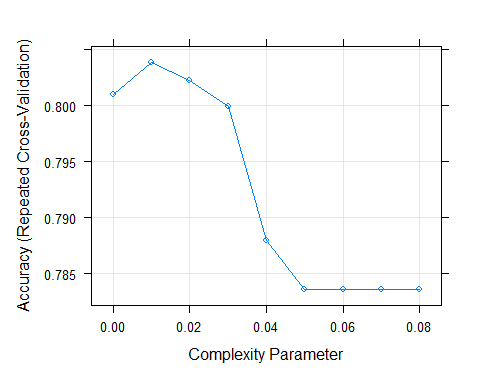
fancyRpartPlot(model\_DT1$finalModel)



model\_DT2 <- train(rating ~. , data=WC\_train, method = "rpart",  
 preProcess = c("center", "scale"),  
 control = rpart.control(minsplit= 200, minbucket = 100, maxdepth = 4),  
 tuneGrid = expand.grid(cp= seq(0, .08, 0.01)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_DT2

## CART   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3306, 3308, 3307, 3306, 3306, 3307, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00 0.8010287 0.2440036  
## 0.01 0.8038424 0.2764323  
## 0.02 0.8022095 0.2571148  
## 0.03 0.7999428 0.2222275  
## 0.04 0.7879692 0.0958083  
## 0.05 0.7836160 0.0000000  
## 0.06 0.7836160 0.0000000  
## 0.07 0.7836160 0.0000000  
## 0.08 0.7836160 0.0000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.01.

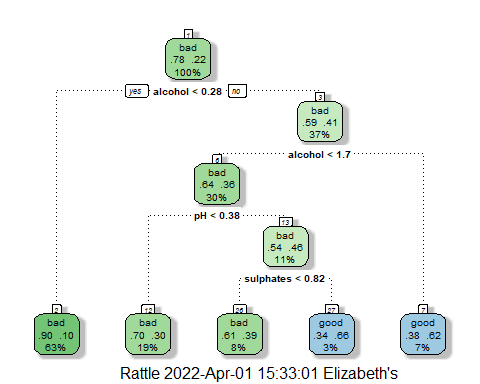
plot(model\_DT2)



predict\_DT2 <- predict(model\_DT2, newdata = WC\_test, type = "raw")  
confusionMatrix(predict\_DT2, WC\_test$rating, positive= "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 912 193  
## good 47 72  
##   
## Accuracy : 0.8039   
## 95% CI : (0.7806, 0.8258)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.04332   
##   
## Kappa : 0.2781   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.27170   
## Specificity : 0.95099   
## Pos Pred Value : 0.60504   
## Neg Pred Value : 0.82534   
## Prevalence : 0.21650   
## Detection Rate : 0.05882   
## Detection Prevalence : 0.09722   
## Balanced Accuracy : 0.61134   
##   
## 'Positive' Class : good   
##

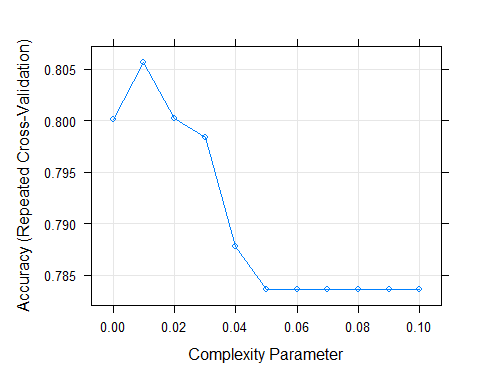
fancyRpartPlot(model\_DT2$finalModel)



model\_DT3 <- train(rating ~. , data=WC\_train, method = "rpart",  
 preProcess = c("center", "scale"),  
 control = rpart.control( maxdepth = 9),  
 tuneGrid = expand.grid(cp= seq(0, .1, 0.01)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_DT3

## CART   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3307, 3306, 3307, 3306, 3307, 3306, ...   
## Resampling results across tuning parameters:  
##   
## cp Accuracy Kappa   
## 0.00 0.8001327 0.3717449  
## 0.01 0.8056571 0.2970164  
## 0.02 0.8002137 0.2399916  
## 0.03 0.7984036 0.2096427  
## 0.04 0.7877909 0.1006138  
## 0.05 0.7836161 0.0000000  
## 0.06 0.7836161 0.0000000  
## 0.07 0.7836161 0.0000000  
## 0.08 0.7836161 0.0000000  
## 0.09 0.7836161 0.0000000  
## 0.10 0.7836161 0.0000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was cp = 0.01.

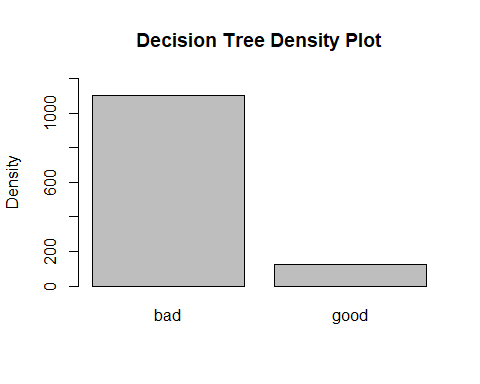
plot(model\_DT3)



predict\_DT3 <- predict(model\_DT3, newdata = WC\_test, type = "raw")  
confusionMatrix(predict\_DT3, WC\_test$rating, positive= "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 917 183  
## good 42 82  
##   
## Accuracy : 0.8162   
## 95% CI : (0.7933, 0.8375)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.002655   
##   
## Kappa : 0.329   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.30943   
## Specificity : 0.95620   
## Pos Pred Value : 0.66129   
## Neg Pred Value : 0.83364   
## Prevalence : 0.21650   
## Detection Rate : 0.06699   
## Detection Prevalence : 0.10131   
## Balanced Accuracy : 0.63282   
##   
## 'Positive' Class : good   
##

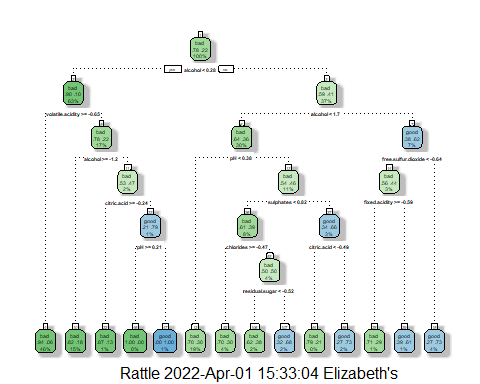
plot(predict\_DT3, ylim= c(0, 1200), ylab = "Density", main = "Decision Tree Density Plot")



print(model\_DT3$finalModel)

## n= 3674   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 3674 795 bad (0.78361459 0.21638541)   
## 2) alcohol< 0.2751388 2306 232 bad (0.89939289 0.10060711)   
## 4) volatile.acidity>=-0.6483597 1679 97 bad (0.94222752 0.05777248) \*  
## 5) volatile.acidity< -0.6483597 627 135 bad (0.78468900 0.21531100)   
## 10) alcohol>=-1.188038 546 97 bad (0.82234432 0.17765568) \*  
## 11) alcohol< -1.188038 81 38 bad (0.53086420 0.46913580)   
## 22) citric.acid>=-0.2419187 39 5 bad (0.87179487 0.12820513) \*  
## 23) citric.acid< -0.2419187 42 9 good (0.21428571 0.78571429)   
## 46) pH>=0.2107949 9 0 bad (1.00000000 0.00000000) \*  
## 47) pH< 0.2107949 33 0 good (0.00000000 1.00000000) \*  
## 3) alcohol>=0.2751388 1368 563 bad (0.58845029 0.41154971)   
## 6) alcohol< 1.657028 1110 402 bad (0.63783784 0.36216216)   
## 12) pH< 0.3763346 697 211 bad (0.69727403 0.30272597) \*  
## 13) pH>=0.3763346 413 191 bad (0.53753027 0.46246973)   
## 26) sulphates< 0.8200884 309 122 bad (0.60517799 0.39482201)   
## 52) chlorides>=-0.4679707 163 49 bad (0.69938650 0.30061350) \*  
## 53) chlorides< -0.4679707 146 73 bad (0.50000000 0.50000000)   
## 106) residual.sugar< -0.5152819 87 33 bad (0.62068966 0.37931034) \*  
## 107) residual.sugar>=-0.5152819 59 19 good (0.32203390 0.67796610) \*  
## 27) sulphates>=0.8200884 104 35 good (0.33653846 0.66346154)   
## 54) citric.acid< -0.4891307 14 3 bad (0.78571429 0.21428571) \*  
## 55) citric.acid>=-0.4891307 90 24 good (0.26666667 0.73333333) \*  
## 7) alcohol>=1.657028 258 97 good (0.37596899 0.62403101)   
## 14) free.sulfur.dioxide< -0.6367623 96 42 bad (0.56250000 0.43750000)   
## 28) fixed.acidity>=-0.5868983 52 15 bad (0.71153846 0.28846154) \*  
## 29) fixed.acidity< -0.5868983 44 17 good (0.38636364 0.61363636) \*  
## 15) free.sulfur.dioxide>=-0.6367623 162 43 good (0.26543210 0.73456790) \*

fancyRpartPlot(model\_DT3$finalModel)



varimp\_dt <- varImp(model\_DT3)  
varimp\_dt

## rpart variable importance  
##   
## Overall  
## alcohol 100.0000  
## density 62.9693  
## chlorides 55.1813  
## residual.sugar 37.3168  
## pH 23.3493  
## total.sulfur.dioxide 12.8692  
## citric.acid 8.8123  
## free.sulfur.dioxide 8.0648  
## fixed.acidity 7.6881  
## sulphates 0.3778  
## volatile.acidity 0.0000

plot\_dt <- plot(varimp\_dt, main = "Variable Importance with Decision Tree")

# Decision Tree Results

The algorithm has several arguments and parameters that can be used to improve the model performance. After sufficient trial and error, it was determined that with a ***maxdepth*** value of 9 and a ***cp*** of 0.01, the model yielded the best results for this dataset. The model also identified 'alcohol' as the most relevant variable here, which was not surprising in light of the earlier findings gained through correlation matrix.

**Results:** Upon running different DT models, different accuracies were obtained, none of which are impressive compared to other algorithms.

Model 1: Accuracy = 79.74%

Model 2: Accuracy = 80.39%

Model 3: Accuracy = 81.62%

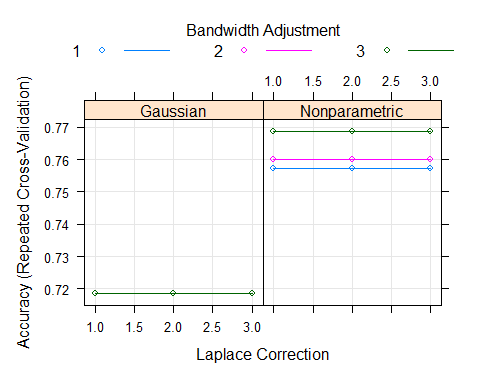
One of the most interesting results from the Decision Tree model is that the top node–alcohol content–branches not to bad and good, but to bad and bad. This reveals the importance of alcohol content in a wine’s composition: if the alcohol content is less than 11%, 90% of wines will be bad Yet, above 11%, there’s still a 59% chance the wine will still be bad. However, above 13% alcohol, there’s a 62% chance the wine will be good.

# Naive Bayes

model\_nb1 <-caret::train(rating ~. , data=WC\_train, method="nb",  
 preProcess = c("center", "scale"),  
 tuneGrid = expand.grid(usekernel = c(TRUE, FALSE), fL = 1:3, adjust = 1:3),  
   
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
print(model\_nb1)

## Naive Bayes   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3306, 3307, 3306, 3307, 3306, 3307, ...   
## Resampling results across tuning parameters:  
##   
## usekernel fL adjust Accuracy Kappa   
## FALSE 1 1 0.7184610 0.3423836  
## FALSE 1 2 0.7184610 0.3423836  
## FALSE 1 3 0.7184610 0.3423836  
## FALSE 2 1 0.7184610 0.3423836  
## FALSE 2 2 0.7184610 0.3423836  
## FALSE 2 3 0.7184610 0.3423836  
## FALSE 3 1 0.7184610 0.3423836  
## FALSE 3 2 0.7184610 0.3423836  
## FALSE 3 3 0.7184610 0.3423836  
## TRUE 1 1 0.7572994 0.3563814  
## TRUE 1 2 0.7600218 0.3559955  
## TRUE 1 3 0.7688228 0.3526540  
## TRUE 2 1 0.7572994 0.3563814  
## TRUE 2 2 0.7600218 0.3559955  
## TRUE 2 3 0.7688228 0.3526540  
## TRUE 3 1 0.7572994 0.3563814  
## TRUE 3 2 0.7600218 0.3559955  
## TRUE 3 3 0.7688228 0.3526540  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were fL = 1, usekernel = TRUE and adjust  
## = 3.

plot(model\_nb1)



predict\_model\_nb1 <- predict(model\_nb1, newdata = WC\_test)  
confusionMatrix(predict\_model\_nb1, WC\_test$rating, positive= "good")

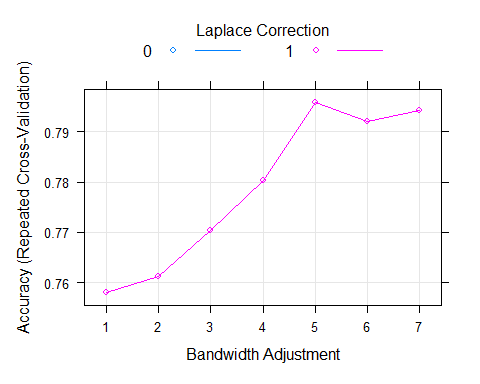
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 805 102  
## good 154 163  
##   
## Accuracy : 0.7908   
## 95% CI : (0.767, 0.8133)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.279076   
##   
## Kappa : 0.4244   
##   
## Mcnemar's Test P-Value : 0.001435   
##   
## Sensitivity : 0.6151   
## Specificity : 0.8394   
## Pos Pred Value : 0.5142   
## Neg Pred Value : 0.8875   
## Prevalence : 0.2165   
## Detection Rate : 0.1332   
## Detection Prevalence : 0.2590   
## Balanced Accuracy : 0.7273   
##   
## 'Positive' Class : good   
##

model\_nb2 <-train(rating ~. , data=WC\_train, method="nb",  
 preProcess = c("center", "scale"),  
 tuneGrid = expand.grid(usekernel = TRUE, fL = 0:1, adjust = 1:7), #adjust = 1:6  
   
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))

print(model\_nb2)

## Naive Bayes   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3306, 3306, 3306, 3306, 3307, 3307, ...   
## Resampling results across tuning parameters:  
##   
## fL adjust Accuracy Kappa   
## 0 1 0.7581287 0.3590643  
## 0 2 0.7613037 0.3602282  
## 0 3 0.7703760 0.3585243  
## 0 4 0.7802660 0.3419707  
## 0 5 0.7957798 0.3122688  
## 0 6 0.7919639 0.2015369  
## 0 7 0.7943264 0.1253351  
## 1 1 0.7581287 0.3590643  
## 1 2 0.7613037 0.3602282  
## 1 3 0.7703760 0.3585243  
## 1 4 0.7802660 0.3419707  
## 1 5 0.7957798 0.3122688  
## 1 6 0.7919639 0.2015369  
## 1 7 0.7943264 0.1253351  
##   
## Tuning parameter 'usekernel' was held constant at a value of TRUE  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were fL = 0, usekernel = TRUE and adjust  
## = 5.

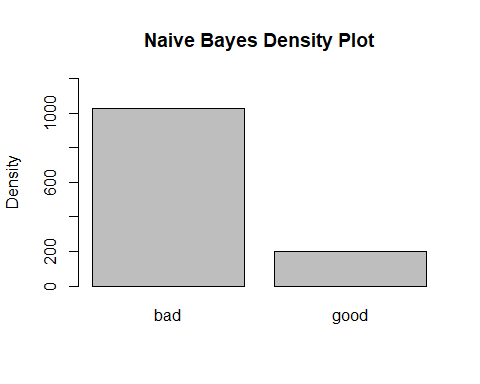
plot(model\_nb2)



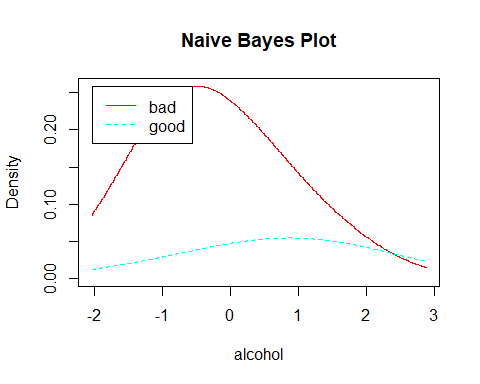
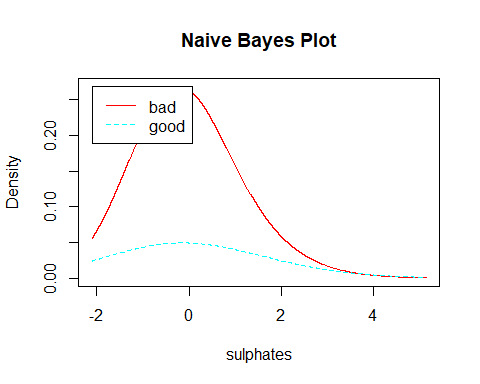
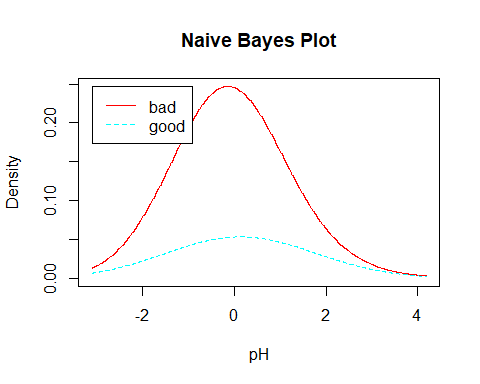
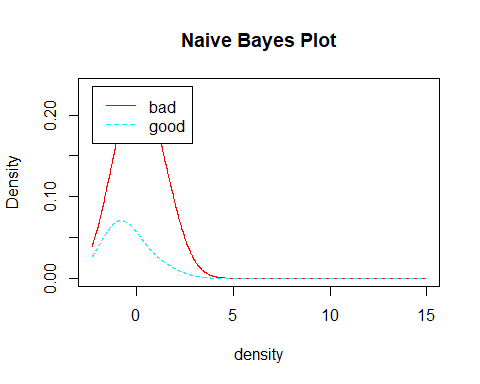
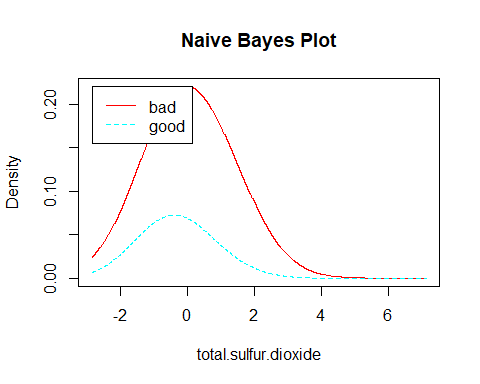
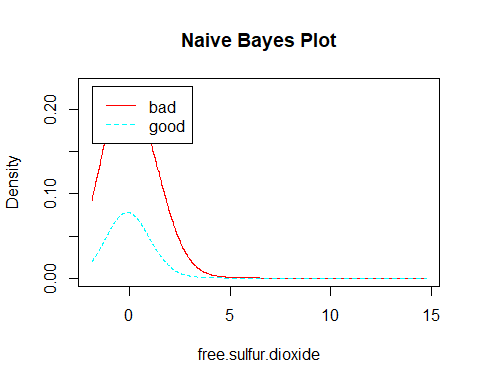
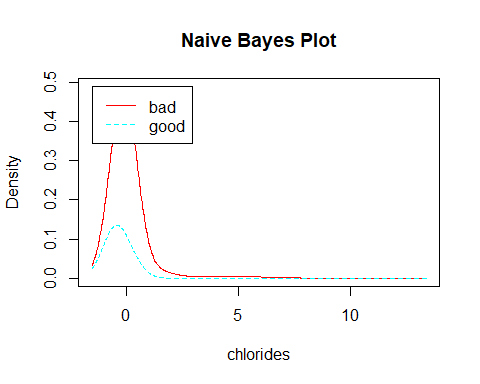
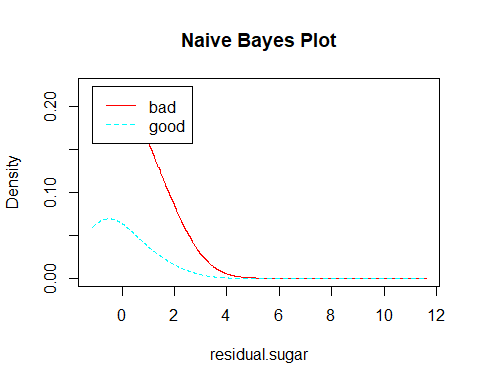
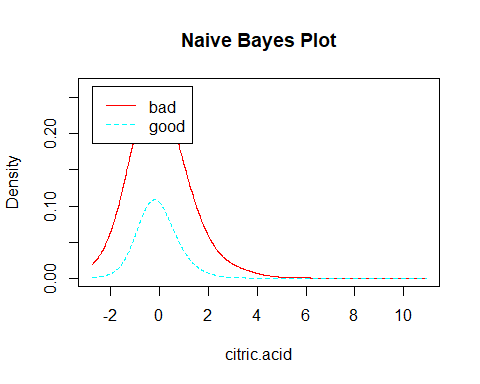
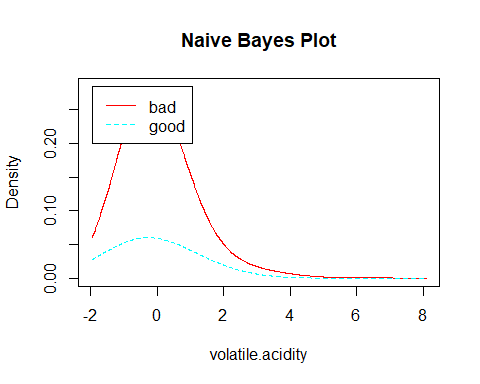
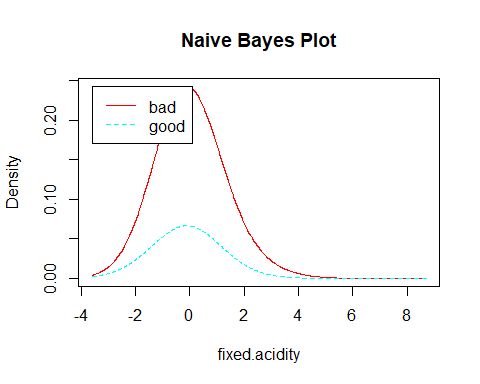
predict\_model\_nb2 <- predict(model\_nb2, newdata = WC\_test)  
confusionMatrix(predict\_model\_nb2, WC\_test$rating, positive= "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 872 153  
## good 87 112  
##   
## Accuracy : 0.8039   
## 95% CI : (0.7806, 0.8258)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.04332   
##   
## Kappa : 0.3648   
##   
## Mcnemar's Test P-Value : 2.72e-05   
##   
## Sensitivity : 0.4226   
## Specificity : 0.9093   
## Pos Pred Value : 0.5628   
## Neg Pred Value : 0.8507   
## Prevalence : 0.2165   
## Detection Rate : 0.0915   
## Detection Prevalence : 0.1626   
## Balanced Accuracy : 0.6660   
##   
## 'Positive' Class : good   
##

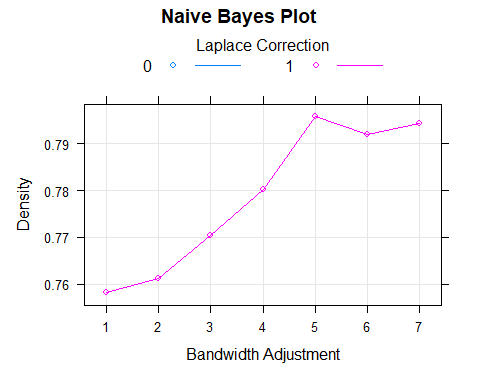
plot(predict\_model\_nb2, ylim= c(0, 1200), ylab = "Density", main = "Naive Bayes Density Plot")



plot(model\_nb2$finalModel)



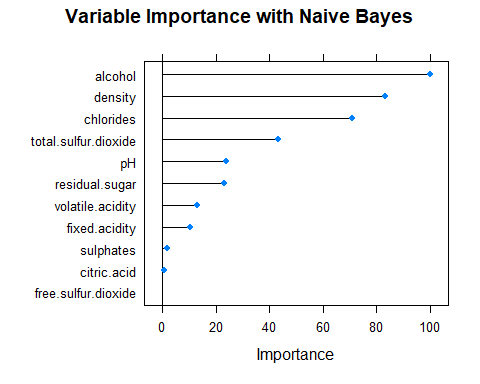
plot(model\_nb2, n = 500, legendplot = TRUE,   
 ylab = "Density", main = "Naive Bayes Plot")



varimp\_nb <- varImp(model\_nb2)  
varimp\_nb

## ROC curve variable importance  
##   
## Importance  
## alcohol 100.0000  
## density 83.2936  
## chlorides 70.9580  
## total.sulfur.dioxide 43.2664  
## pH 23.7961  
## residual.sugar 22.9430  
## volatile.acidity 13.1293  
## fixed.acidity 10.3608  
## sulphates 1.8046  
## citric.acid 0.6892  
## free.sulfur.dioxide 0.0000

plot(varimp\_nb, main = "Variable Importance with Naive Bayes")



## Results

Laplace smoothing parameters of 0-4 were tested, as well as bandwidth adjustments of 0-7. Gaussian kernel models consistently returned accuracies 5% lower than nonparametric kernels.

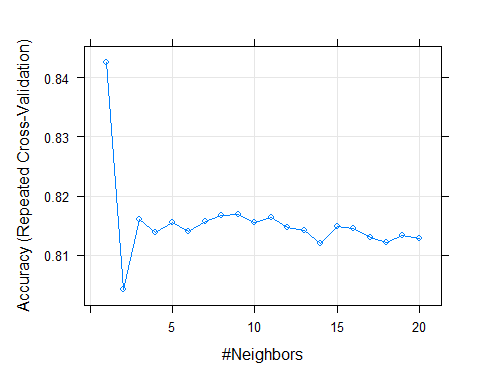
In the final model, nonparametric kernel was used, the bandwidth adjustment was set to 5 (in order to allow a more flexible density estimate, and Laplace smoothing was set to 0 (though final results were identical with Laplace smoothing of 0 and 1). Performance accuracy improved from about 76% to 80.4% in the final.

# KNN

model\_knn1 <-train(rating ~. , data=WC\_train, method="knn",  
 preProcess = c("center", "scale"),  
 tuneGrid= data.frame(k= seq(1, 20)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
print(model\_knn1)

## k-Nearest Neighbors   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3306, 3306, 3307, 3306, 3308, 3307, ...   
## Resampling results across tuning parameters:  
##   
## k Accuracy Kappa   
## 1 0.8425908 0.5348864  
## 2 0.8042035 0.4310363  
## 3 0.8160050 0.4422601  
## 4 0.8139096 0.4363186  
## 5 0.8155450 0.4261179  
## 6 0.8140096 0.4183060  
## 7 0.8156437 0.4118014  
## 8 0.8166445 0.4091012  
## 9 0.8168247 0.4044169  
## 10 0.8154586 0.3987384  
## 11 0.8163708 0.3945471  
## 12 0.8147362 0.3874386  
## 13 0.8141016 0.3805485  
## 14 0.8120151 0.3742360  
## 15 0.8148248 0.3773569  
## 16 0.8145525 0.3759533  
## 17 0.8130149 0.3683166  
## 18 0.8121042 0.3626872  
## 19 0.8132862 0.3655026  
## 20 0.8127419 0.3621241  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was k = 1.

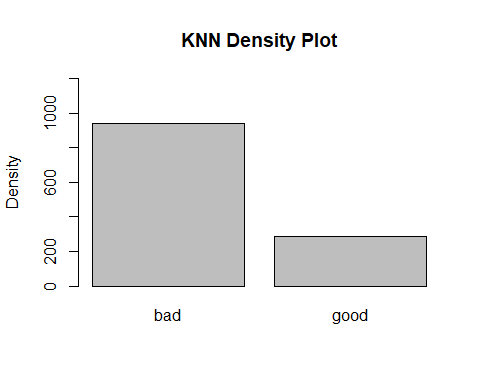
plot(model\_knn1)



predict\_model\_knn1 <- predict(model\_knn1, newdata = WC\_test)  
confusionMatrix(predict\_model\_knn1, WC\_test$rating, positive= "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 860 78  
## good 99 187  
##   
## Accuracy : 0.8554   
## 95% CI : (0.8344, 0.8746)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 1.041e-10   
##   
## Kappa : 0.5856   
##   
## Mcnemar's Test P-Value : 0.1328   
##   
## Sensitivity : 0.7057   
## Specificity : 0.8968   
## Pos Pred Value : 0.6538   
## Neg Pred Value : 0.9168   
## Prevalence : 0.2165   
## Detection Rate : 0.1528   
## Detection Prevalence : 0.2337   
## Balanced Accuracy : 0.8012   
##   
## 'Positive' Class : good   
##

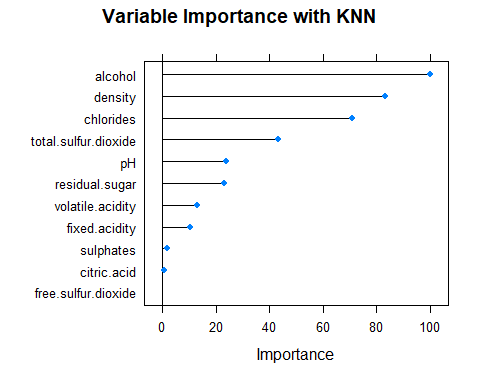
plot(predict\_model\_knn1, ylim= c(0, 1200), ylab = "Density", main = "KNN Density Plot")



varimp\_knn <- varImp(model\_knn1)  
varimp\_knn

## ROC curve variable importance  
##   
## Importance  
## alcohol 100.0000  
## density 83.2936  
## chlorides 70.9580  
## total.sulfur.dioxide 43.2664  
## pH 23.7961  
## residual.sugar 22.9430  
## volatile.acidity 13.1293  
## fixed.acidity 10.3608  
## sulphates 1.8046  
## citric.acid 0.6892  
## free.sulfur.dioxide 0.0000

plot(varimp\_knn, main = "Variable Importance with KNN")



## Results

The optimal model exhibited prediction accuracy of 85.54% using k = 1. Like Decision Trees, KNN also evaluated 'alcohol' as the most significant variable impacting the wine quality followed by 'density' and 'chlorides'.

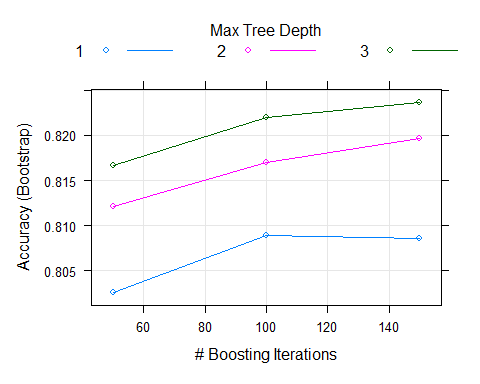
# Random Forest and Gradient Boost Machine

Prediction performance improved from the untrained gbm model (Accuracy : 0.8333) with the following tuning parameters:

model\_gbm1<-train(rating~. , data= WC\_train,method="gbm",verbose=FALSE)  
model\_gbm1

## Stochastic Gradient Boosting   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## No pre-processing  
## Resampling: Bootstrapped (25 reps)   
## Summary of sample sizes: 3674, 3674, 3674, 3674, 3674, 3674, ...   
## Resampling results across tuning parameters:  
##   
## interaction.depth n.trees Accuracy Kappa   
## 1 50 0.8026446 0.2261365  
## 1 100 0.8089612 0.2906769  
## 1 150 0.8086317 0.3057812  
## 2 50 0.8121283 0.3049669  
## 2 100 0.8170596 0.3512430  
## 2 150 0.8196284 0.3727965  
## 3 50 0.8166519 0.3366191  
## 3 100 0.8219796 0.3833811  
## 3 150 0.8236252 0.3978294  
##   
## Tuning parameter 'shrinkage' was held constant at a value of 0.1  
##   
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 150, interaction.depth =  
## 3, shrinkage = 0.1 and n.minobsinnode = 10.

plot(model\_gbm1)

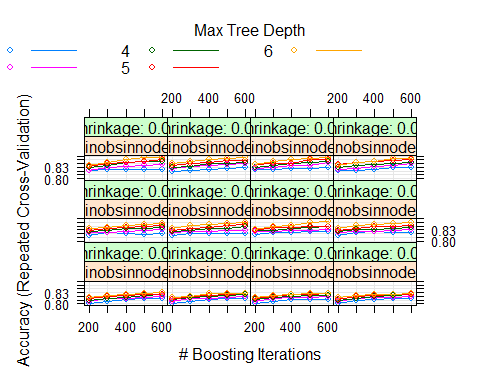


predict\_gbm1 <- predict(model\_gbm1, newdata = WC\_test)  
confusionMatrix(predict\_gbm1, WC\_test$rating, positive= "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 906 152  
## good 53 113  
##   
## Accuracy : 0.8325   
## 95% CI : (0.8104, 0.853)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 1.078e-05   
##   
## Kappa : 0.4292   
##   
## Mcnemar's Test P-Value : 7.668e-12   
##   
## Sensitivity : 0.42642   
## Specificity : 0.94473   
## Pos Pred Value : 0.68072   
## Neg Pred Value : 0.85633   
## Prevalence : 0.21650   
## Detection Rate : 0.09232   
## Detection Prevalence : 0.13562   
## Balanced Accuracy : 0.68557   
##   
## 'Positive' Class : good   
##

grid<-expand.grid(.n.trees=seq(200,600,by=100),.interaction.depth=seq(2,6,by=1),.shrinkage=seq(.01,.09,by=.04),  
 .n.minobsinnode=seq(2,11,by=3)) #grid features  
control<-trainControl(method="repeatedcv",number = 5, repeats = 2) #control

plot(model\_gbm\_train)



predict\_gbm <- predict(model\_gbm\_train, newdata = WC\_test)  
confusionMatrix(predict\_gbm, WC\_test$rating, positive= "good")

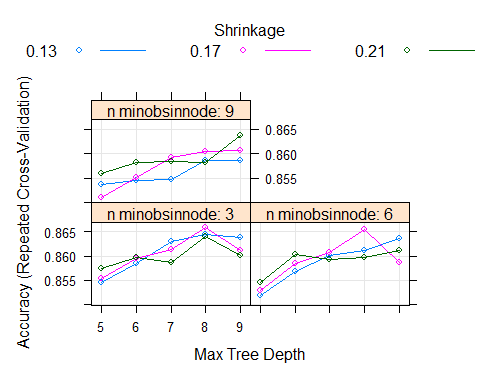
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 903 111  
## good 56 154  
##   
## Accuracy : 0.8636   
## 95% CI : (0.843, 0.8823)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 5.097e-13   
##   
## Kappa : 0.5652   
##   
## Mcnemar's Test P-Value : 2.933e-05   
##   
## Sensitivity : 0.5811   
## Specificity : 0.9416   
## Pos Pred Value : 0.7333   
## Neg Pred Value : 0.8905   
## Prevalence : 0.2165   
## Detection Rate : 0.1258   
## Detection Prevalence : 0.1716   
## Balanced Accuracy : 0.7614   
##   
## 'Positive' Class : good   
##

grid<-expand.grid(.n.trees=500,.interaction.depth=seq(5,9,by=1),.shrinkage=seq(.13,.21,by=.04),  
 .n.minobsinnode=seq(3,9,by=3)) #grid featuresseq(500,700,by=100)  
control<-trainControl(method="repeatedcv",number = 10, repeats = 3) #control

print(model\_gbm3)

## Stochastic Gradient Boosting   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3307, 3306, 3306, 3308, 3307, 3306, ...   
## Resampling results across tuning parameters:  
##   
## shrinkage interaction.depth n.minobsinnode Accuracy Kappa   
## 0.13 5 3 0.8545676 0.5289309  
## 0.13 5 6 0.8518443 0.5234843  
## 0.13 5 9 0.8535670 0.5274508  
## 0.13 6 3 0.8583749 0.5468432  
## 0.13 6 6 0.8568341 0.5413947  
## 0.13 6 9 0.8544735 0.5350694  
## 0.13 7 3 0.8630078 0.5610734  
## 0.13 7 6 0.8601890 0.5558036  
## 0.13 7 9 0.8546552 0.5387510  
## 0.13 8 3 0.8643650 0.5689046  
## 0.13 8 6 0.8611838 0.5592763  
## 0.13 8 9 0.8586466 0.5531565  
## 0.13 9 3 0.8637339 0.5659031  
## 0.13 9 6 0.8635476 0.5669048  
## 0.13 9 9 0.8585528 0.5534129  
## 0.17 5 3 0.8553835 0.5415578  
## 0.17 5 6 0.8530253 0.5316558  
## 0.17 5 9 0.8509395 0.5254138  
## 0.17 6 3 0.8595549 0.5567551  
## 0.17 6 6 0.8583749 0.5518532  
## 0.17 6 9 0.8550190 0.5420173  
## 0.17 7 3 0.8612786 0.5618134  
## 0.17 7 6 0.8607391 0.5571161  
## 0.17 7 9 0.8591923 0.5560446  
## 0.17 8 3 0.8659068 0.5743405  
## 0.17 8 6 0.8653594 0.5729361  
## 0.17 8 9 0.8603728 0.5576264  
## 0.17 9 3 0.8611856 0.5615152  
## 0.17 9 6 0.8586491 0.5519678  
## 0.17 9 9 0.8606409 0.5565163  
## 0.21 5 3 0.8573741 0.5491845  
## 0.21 5 6 0.8545639 0.5407679  
## 0.21 5 9 0.8560191 0.5488028  
## 0.21 6 3 0.8597380 0.5565137  
## 0.21 6 6 0.8602788 0.5623695  
## 0.21 6 9 0.8581903 0.5550678  
## 0.21 7 3 0.8586518 0.5569169  
## 0.21 7 6 0.8591904 0.5564103  
## 0.21 7 9 0.8583803 0.5543367  
## 0.21 8 3 0.8640024 0.5716042  
## 0.21 8 6 0.8596415 0.5591128  
## 0.21 8 9 0.8580985 0.5543398  
## 0.21 9 3 0.8601065 0.5602618  
## 0.21 9 6 0.8610928 0.5628686  
## 0.21 9 9 0.8638181 0.5713324  
##   
## Tuning parameter 'n.trees' was held constant at a value of 500  
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were n.trees = 500, interaction.depth =  
## 8, shrinkage = 0.17 and n.minobsinnode = 3.

plot(model\_gbm3)



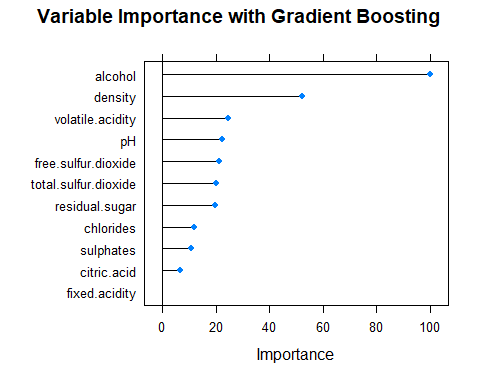
predict\_gbm3 <- predict(model\_gbm3, newdata = WC\_test)  
confusionMatrix(predict\_gbm3, WC\_test$rating, positive= "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 900 102  
## good 59 163  
##   
## Accuracy : 0.8685   
## 95% CI : (0.8482, 0.8869)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 1.513e-14   
##   
## Kappa : 0.5881   
##   
## Mcnemar's Test P-Value : 0.0009327   
##   
## Sensitivity : 0.6151   
## Specificity : 0.9385   
## Pos Pred Value : 0.7342   
## Neg Pred Value : 0.8982   
## Prevalence : 0.2165   
## Detection Rate : 0.1332   
## Detection Prevalence : 0.1814   
## Balanced Accuracy : 0.7768   
##   
## 'Positive' Class : good   
##

varimp\_gbm <- varImp(model\_gbm3)  
varimp\_gbm

## gbm variable importance  
##   
## Overall  
## alcohol 100.000  
## density 52.153  
## volatile.acidity 24.729  
## pH 22.502  
## free.sulfur.dioxide 21.340  
## total.sulfur.dioxide 19.942  
## residual.sugar 19.679  
## chlorides 11.897  
## sulphates 10.638  
## citric.acid 6.508  
## fixed.acidity 0.000

plot(varimp\_gbm, main = "Variable Importance with Gradient Boosting")



## Results

Prediction performance improved from the untrained gbm model (Accuracy : 0.8333) with the following tuning parameters: .n.trees=500,.interaction.depth=seq(5,9,by=1),.shrinkage=seq(.13,.21,by=.04), .n.minobsinnode=seq(3,9,by=3)). Final prediction accuracy came in at an impressive 88.64%

Models were run with few trees (iterations) and with many trees (first, 50-150, then 400-700). After multiple experiments tuning with sequences, n.trees (number of iterations) was set to 500, as performance was negligibly improved with n.trees above 500. The remaining parameters were narrowed to the best range of options, and best model parameters fluctuated within these ranges, yielding different “winning” parameters each time the model was run.

Interaction.depth (highest level of variable interactions) was tested from 1-12. The most accurate model used 9. Shrinkage (learning rate) was tested with parameters from 0.01 to 0.25. Sucessful models performed best between .13 and .17, with the final model using a learning rate of 0.13. Minimum observations per tree (n.minobsinnode) was tested from 2 to 12. The final model utilized n.minobsinnode = 6

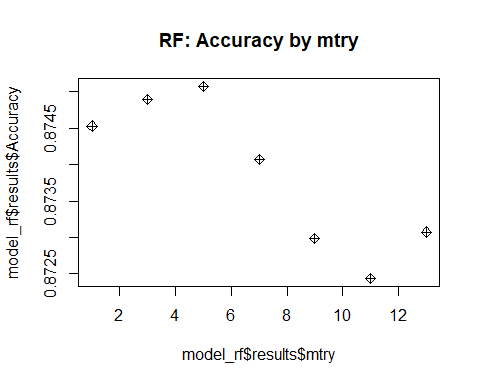
model\_rf <- caret::train(rating~. , data= WC\_train, method= "rf",  
 preProcess = c("center", "scale"),  
 tuneGrid = expand.grid(mtry = seq(1, 13, 2)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_rf

## Random Forest   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3307, 3307, 3307, 3306, 3306, 3307, ...   
## Resampling results across tuning parameters:  
##   
## mtry Accuracy Kappa   
## 1 0.8745272 0.5685409  
## 3 0.8748878 0.5856265  
## 5 0.8750700 0.5896283  
## 7 0.8740721 0.5867244  
## 9 0.8729849 0.5849954  
## 11 0.8724397 0.5829573  
## 13 0.8730747 0.5870557  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 5.

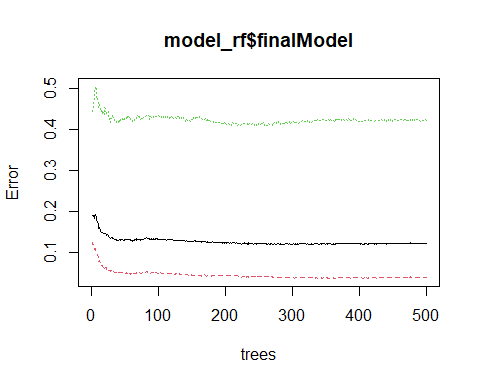
summary(model\_rf$resample$Accuracy)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.8392 0.8668 0.8748 0.8751 0.8822 0.8992

plot(model\_rf$results$mtry, model\_rf$results$Accuracy, main = "RF: Accuracy by mtry", pch = 9 )



plot(model\_rf$finalModel)

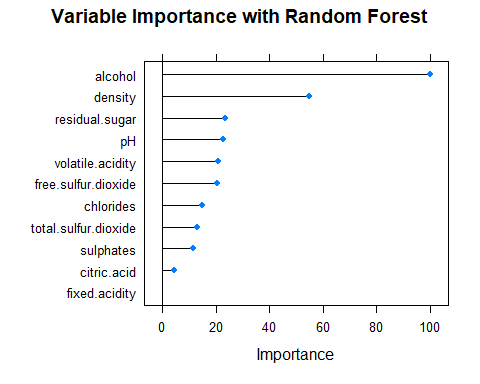


## Attribute Importance

varimp\_rf <- varImp(model\_rf)  
varimp\_rf

## rf variable importance  
##   
## Overall  
## alcohol 100.00  
## density 54.80  
## residual.sugar 23.61  
## pH 22.81  
## volatile.acidity 20.65  
## free.sulfur.dioxide 20.35  
## chlorides 14.74  
## total.sulfur.dioxide 12.85  
## sulphates 11.44  
## citric.acid 4.45  
## fixed.acidity 0.00

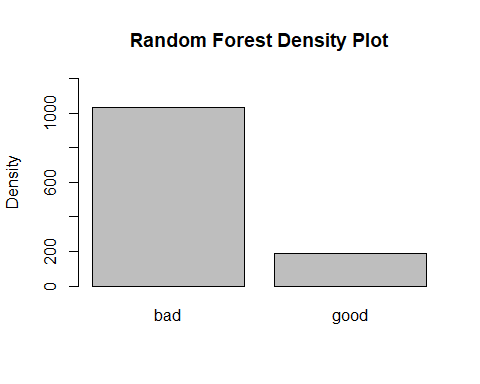
plot(varimp\_rf, main = "Variable Importance with Random Forest")



predict\_rf <- predict(model\_rf, newdata = WC\_test)  
confusionMatrix(predict\_rf, WC\_test$rating)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 929 106  
## good 30 159  
##   
## Accuracy : 0.8889   
## 95% CI : (0.8699, 0.906)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6346   
##   
## Mcnemar's Test P-Value : 1.266e-10   
##   
## Sensitivity : 0.9687   
## Specificity : 0.6000   
## Pos Pred Value : 0.8976   
## Neg Pred Value : 0.8413   
## Prevalence : 0.7835   
## Detection Rate : 0.7590   
## Detection Prevalence : 0.8456   
## Balanced Accuracy : 0.7844   
##   
## 'Positive' Class : bad   
##

plot(predict\_rf, ylim= c(0, 1200), ylab = "Density", main = "Random Forest Density Plot")



For Random Forest Method, the resampling and preprocessing techniques remain the same : 10-fold Cross-Validation repeated 3 times with “center” and “scale.” The final value used for the model was mtry = 2. Very little tuning is required.

## Results

This model performed better than previous models for this dataset resulting in a very high accuracy of 88.73%. The kappa value was also observed to be significantly higher at a statistic of 0.629.

# SVM, Radial Basis Function Kernel

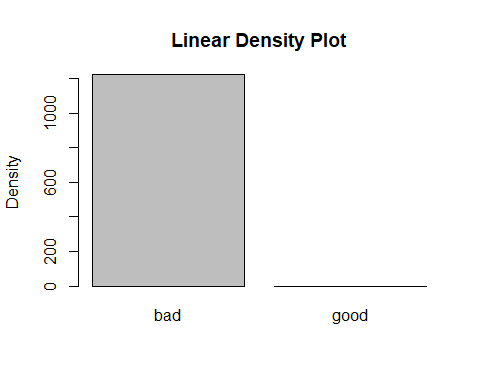
model\_linear <- caret::train(rating ~. , WC\_train,  
 preProcess = c("center", "scale"),  
 method= "svmLinear",  
 tuneGrid = expand.grid(C = seq(0, 12, 2)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_linear

## Support Vector Machines with Linear Kernel   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3307, 3306, 3306, 3306, 3307, 3307, ...   
## Resampling results across tuning parameters:  
##   
## C Accuracy Kappa  
## 0 NaN NaN   
## 2 0.7836158 0   
## 4 0.7836158 0   
## 6 0.7836158 0   
## 8 0.7836158 0   
## 10 0.7836158 0   
## 12 0.7836158 0   
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final value used for the model was C = 2.

predict\_svm\_linear <- predict(model\_linear, WC\_test)  
confusionMatrix(predict\_svm\_linear, WC\_test$rating, positive = "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 959 265  
## good 0 0  
##   
## Accuracy : 0.7835   
## 95% CI : (0.7593, 0.8063)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.5165   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.7835   
## Prevalence : 0.2165   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : good   
##

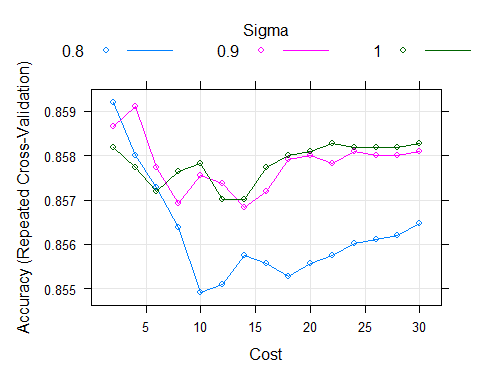
plot(predict\_svm\_linear, ylim= c(0, 1200), ylab = "Density", main = "Linear Density Plot")



model\_svm\_rbf2 <- caret::train(rating ~. , WC\_train,  
 preProcess = c("center", "scale"),  
 method= "svmRadial",  
 tuneGrid = expand.grid(sigma = seq(0.8, 1, 0.1),  
 C = seq(2, 30, 2)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_svm\_rbf2

## Support Vector Machines with Radial Basis Function Kernel   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: centered (11), scaled (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3306, 3307, 3307, 3307, 3307, 3307, ...   
## Resampling results across tuning parameters:  
##   
## sigma C Accuracy Kappa   
## 0.8 2 0.8591923 0.5301239  
## 0.8 4 0.8580138 0.5349955  
## 0.8 6 0.8572865 0.5335392  
## 0.8 8 0.8563794 0.5313842  
## 0.8 10 0.8549269 0.5278264  
## 0.8 12 0.8551083 0.5290763  
## 0.8 14 0.8557436 0.5310603  
## 0.8 16 0.8555622 0.5306823  
## 0.8 18 0.8552890 0.5300487  
## 0.8 20 0.8555610 0.5308626  
## 0.8 22 0.8557424 0.5312543  
## 0.8 24 0.8560146 0.5321878  
## 0.8 26 0.8561052 0.5325661  
## 0.8 28 0.8561958 0.5327787  
## 0.8 30 0.8564680 0.5336036  
## 0.9 2 0.8586486 0.5250537  
## 0.9 4 0.8591020 0.5332618  
## 0.9 6 0.8577396 0.5296354  
## 0.9 8 0.8569224 0.5274029  
## 0.9 10 0.8575575 0.5303823  
## 0.9 12 0.8573763 0.5301154  
## 0.9 14 0.8568323 0.5287526  
## 0.9 16 0.8571941 0.5299562  
## 0.9 18 0.8579198 0.5316970  
## 0.9 20 0.8580101 0.5319284  
## 0.9 22 0.8578292 0.5314909  
## 0.9 24 0.8581009 0.5323052  
## 0.9 26 0.8580099 0.5320620  
## 0.9 28 0.8580099 0.5320620  
## 0.9 30 0.8581004 0.5322860  
## 1.0 2 0.8581940 0.5194159  
## 1.0 4 0.8577379 0.5231942  
## 1.0 6 0.8571941 0.5219258  
## 1.0 8 0.8576475 0.5245285  
## 1.0 10 0.8578292 0.5247760  
## 1.0 12 0.8570122 0.5227754  
## 1.0 14 0.8570125 0.5226270  
## 1.0 16 0.8577376 0.5247101  
## 1.0 18 0.8580096 0.5253740  
## 1.0 20 0.8581004 0.5255953  
## 1.0 22 0.8582818 0.5260101  
## 1.0 24 0.8581905 0.5257883  
## 1.0 26 0.8581903 0.5257908  
## 1.0 28 0.8581903 0.5256067  
## 1.0 30 0.8582809 0.5258321  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were sigma = 0.8 and C = 2.

plot(model\_svm\_rbf2)



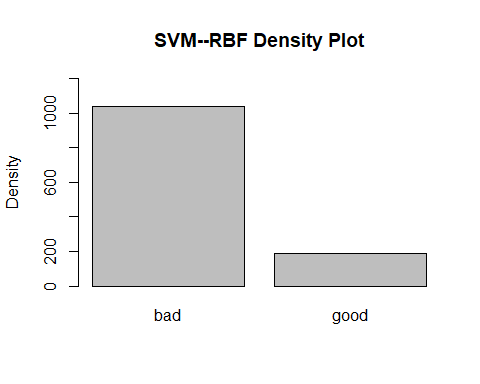
predict\_svm\_rbf2 <- predict(model\_svm\_rbf2, WC\_test)  
confusionMatrix(predict\_svm\_rbf2, WC\_test$rating, positive = "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 921 116  
## good 38 149  
##   
## Accuracy : 0.8742   
## 95% CI : (0.8543, 0.8923)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.5849   
##   
## Mcnemar's Test P-Value : 5.475e-10   
##   
## Sensitivity : 0.5623   
## Specificity : 0.9604   
## Pos Pred Value : 0.7968   
## Neg Pred Value : 0.8881   
## Prevalence : 0.2165   
## Detection Rate : 0.1217   
## Detection Prevalence : 0.1528   
## Balanced Accuracy : 0.7613   
##   
## 'Positive' Class : good   
##

svm\_table <-confusionMatrix(predict\_svm\_rbf2, WC\_test$rating, positive = "good")  
(svm\_table$table)

## Reference  
## Prediction bad good  
## bad 921 116  
## good 38 149

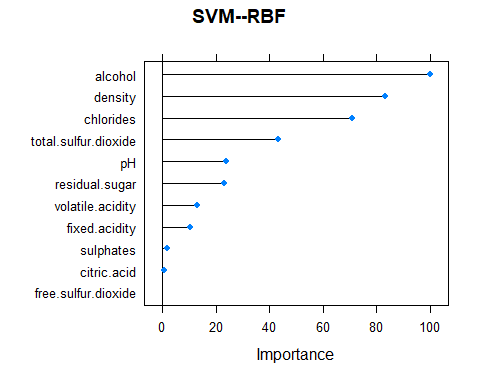
plot(predict\_svm\_rbf2, ylim= c(0, 1200), ylab = "Density", main = "SVM--RBF Density Plot")



varimp\_svm2 <- (varImp(model\_svm\_rbf2))  
(varimp\_svm2)

## ROC curve variable importance  
##   
## Importance  
## alcohol 100.0000  
## density 83.2936  
## chlorides 70.9580  
## total.sulfur.dioxide 43.2664  
## pH 23.7961  
## residual.sugar 22.9430  
## volatile.acidity 13.1293  
## fixed.acidity 10.3608  
## sulphates 1.8046  
## citric.acid 0.6892  
## free.sulfur.dioxide 0.0000

plot(varimp\_svm2, main="SVM--RBF")



Models were run with sequences of sigma values between 0 and 1, sequencing at intervals of 0.1. Later, cost was sequenced from 0.8-1, with intervals of 0.1. Costs of 0 – 40 were explored. Early on, costs in the mid-20’s appeared.

## Results

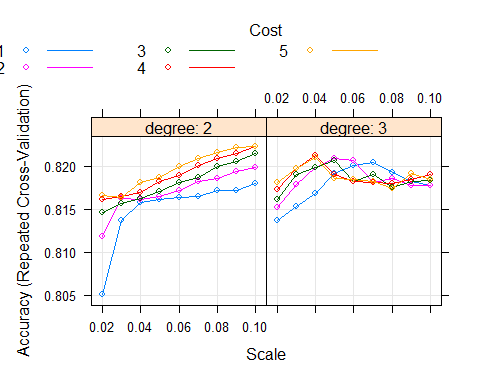
In the final model, sigma is 0.9 and cost is 2; this returned a prediction accuracy of 87.34% and a kappa value of 0.584.

# SVM Polynomial

model\_svm\_poly <- caret::train(rating ~. , WC\_train,  
 preProcess = c("scale", "center"),  
 method = 'svmPoly',  
 tuneGrid = expand.grid(scale = seq(0.02, .1, 0.01),  
 C = seq(1, 5, 1),  
 degree = seq(2, 3, 1)),  
 trControl = trainControl(method = "repeatedcv", number = 10, repeats=3))  
model\_svm\_poly

## Support Vector Machines with Polynomial Kernel   
##   
## 3674 samples  
## 11 predictor  
## 2 classes: 'bad', 'good'   
##   
## Pre-processing: scaled (11), centered (11)   
## Resampling: Cross-Validated (10 fold, repeated 3 times)   
## Summary of sample sizes: 3307, 3307, 3307, 3307, 3306, 3307, ...   
## Resampling results across tuning parameters:  
##   
## scale C degree Accuracy Kappa   
## 0.02 1 2 0.8050277 0.1924198  
## 0.02 1 3 0.8137345 0.2741948  
## 0.02 2 2 0.8118296 0.2507999  
## 0.02 2 3 0.8151860 0.2908501  
## 0.02 3 2 0.8146427 0.2803628  
## 0.02 3 3 0.8161865 0.3028121  
## 0.02 4 2 0.8160942 0.2918984  
## 0.02 4 3 0.8172740 0.3107855  
## 0.02 5 2 0.8166387 0.2978677  
## 0.02 5 3 0.8180907 0.3157976  
## 0.03 1 2 0.8137352 0.2635237  
## 0.03 1 3 0.8152783 0.2944165  
## 0.03 2 2 0.8162756 0.2940096  
## 0.03 2 3 0.8179095 0.3150106  
## 0.03 3 2 0.8156416 0.2966343  
## 0.03 3 3 0.8190890 0.3259074  
## 0.03 4 2 0.8164573 0.3052275  
## 0.03 4 3 0.8197228 0.3324429  
## 0.03 5 2 0.8163665 0.3085115  
## 0.03 5 3 0.8197221 0.3366143  
## 0.04 1 2 0.8158220 0.2895896  
## 0.04 1 3 0.8168201 0.3095100  
## 0.04 2 2 0.8161846 0.3023888  
## 0.04 2 3 0.8198139 0.3332492  
## 0.04 3 2 0.8162754 0.3099819  
## 0.04 3 3 0.8198129 0.3399356  
## 0.04 4 2 0.8170027 0.3161185  
## 0.04 4 3 0.8212659 0.3487718  
## 0.04 5 2 0.8180907 0.3226262  
## 0.04 5 3 0.8209949 0.3510252  
## 0.05 1 2 0.8161855 0.2967852  
## 0.05 1 3 0.8191781 0.3275066  
## 0.05 2 2 0.8164575 0.3110110  
## 0.05 2 3 0.8209038 0.3453681  
## 0.05 3 2 0.8170931 0.3182191  
## 0.05 3 3 0.8206316 0.3492900  
## 0.05 4 2 0.8181813 0.3251936  
## 0.05 4 3 0.8190885 0.3477024  
## 0.05 5 2 0.8187265 0.3285974  
## 0.05 5 3 0.8185438 0.3503709  
## 0.06 1 2 0.8163665 0.3039836  
## 0.06 1 3 0.8200864 0.3385077  
## 0.06 2 2 0.8171836 0.3182346  
## 0.06 2 3 0.8206321 0.3497258  
## 0.06 3 2 0.8180909 0.3260008  
## 0.06 3 3 0.8182718 0.3490803  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were degree = 2, scale = 0.1 and C = 5.

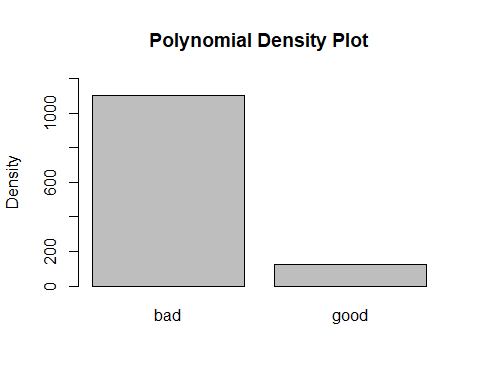
plot(model\_svm\_poly)



predict\_svm\_poly <- predict(model\_svm\_poly, WC\_test)  
confusionMatrix(predict\_svm\_poly, WC\_test$rating, positive = "good")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction bad good  
## bad 922 180  
## good 37 85  
##   
## Accuracy : 0.8227   
## 95% CI : (0.8001, 0.8437)  
## No Information Rate : 0.7835   
## P-Value [Acc > NIR] : 0.0003789   
##   
## Kappa : 0.3506   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.32075   
## Specificity : 0.96142   
## Pos Pred Value : 0.69672   
## Neg Pred Value : 0.83666   
## Prevalence : 0.21650   
## Detection Rate : 0.06944   
## Detection Prevalence : 0.09967   
## Balanced Accuracy : 0.64109   
##   
## 'Positive' Class : good   
##

plot(predict\_svm\_poly, ylim= c(0, 1200), ylab = "Density", main = "Polynomial Density Plot")



For SVM Polynomial Kernel, parameters include scale (set initially to 0.001, 0.01, 0.1, then honed to 0.02 - 0.1). Costs of 0-10 were explored, as well as setting the degree to 1 – 5.

## Results

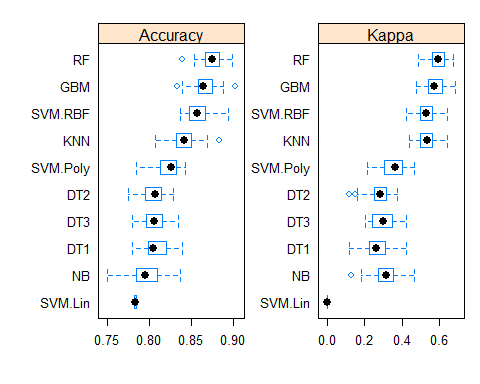
Results from this model were unstable and varied significantly with each trial. The final values used for the model were degree = 3, scale = 0.05, and C = 3, resulting in an accuracy value of 81.86% and a kappa value of 0.334.

# Model Comparison

model\_comparison <-resamples(list(DT1 = model\_DT1, DT2 = model\_DT2, DT3 = model\_DT3, NB= model\_nb2 ,KNN = model\_knn1,   
 RF = model\_rf, GBM= model\_gbm3, SVM.Lin = model\_linear,   
 SVM.RBF = model\_svm\_rbf2, SVM.Poly = model\_svm\_poly ))  
summary(model\_comparison)

##   
## Call:  
## summary.resamples(object = model\_comparison)  
##   
## Models: DT1, DT2, DT3, NB, KNN, RF, GBM, SVM.Lin, SVM.RBF, SVM.Poly   
## Number of resamples: 30   
##   
## Accuracy   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## DT1 0.7798913 0.7985021 0.8040813 0.8073906 0.8205300 0.8392371 0  
## DT2 0.7744565 0.7961838 0.8070652 0.8038424 0.8147139 0.8283379 0  
## DT3 0.7792916 0.7961957 0.8054437 0.8056571 0.8145380 0.8342391 0  
## NB 0.7493188 0.7847411 0.7945593 0.7957798 0.8085831 0.8365123 0  
## KNN 0.8070652 0.8315217 0.8417497 0.8425908 0.8504417 0.8831522 0  
## RF 0.8392371 0.8668478 0.8748297 0.8750700 0.8822340 0.8991826 0  
## GBM 0.8333333 0.8583106 0.8641304 0.8659068 0.8746594 0.9019074 0  
## SVM.Lin 0.7820163 0.7826087 0.7833809 0.7836158 0.7847411 0.7847411 0  
## SVM.RBF 0.8369565 0.8478261 0.8571407 0.8591923 0.8663938 0.8940217 0  
## SVM.Poly 0.7847411 0.8125000 0.8260870 0.8222645 0.8315217 0.8423913 0  
##   
## Kappa   
## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  
## DT1 0.1183891 0.2258267 0.2602932 0.2667172 0.3076975 0.4212662 0  
## DT2 0.1202439 0.2515115 0.2849183 0.2764323 0.3145747 0.3740084 0  
## DT3 0.2032333 0.2438207 0.3013017 0.2970164 0.3451558 0.4214433 0  
## NB 0.1301525 0.2803140 0.3164751 0.3122688 0.3508340 0.4671377 0  
## KNN 0.4355340 0.4954292 0.5339371 0.5348864 0.5621068 0.6386555 0  
## RF 0.4838502 0.5599794 0.5931659 0.5896283 0.6238665 0.6730552 0  
## GBM 0.4764317 0.5390288 0.5704720 0.5743405 0.6114941 0.6802826 0  
## SVM.Lin 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0.0000000 0  
## SVM.RBF 0.4216262 0.4948769 0.5268738 0.5301239 0.5602670 0.6378684 0  
## SVM.Poly 0.2150264 0.3075563 0.3641290 0.3521680 0.4019898 0.4642570 0

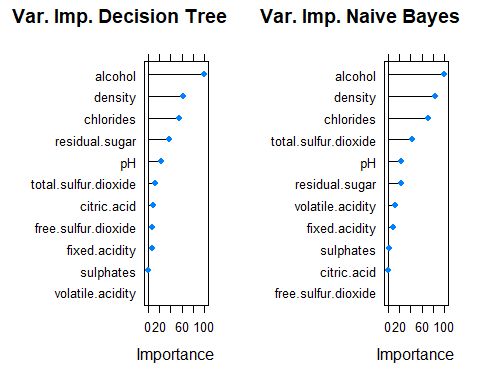
scales <- list(x= list(relation = "free"),  
 y = list(relation = "free"))  
bwplot(model\_comparison, scales = scales)



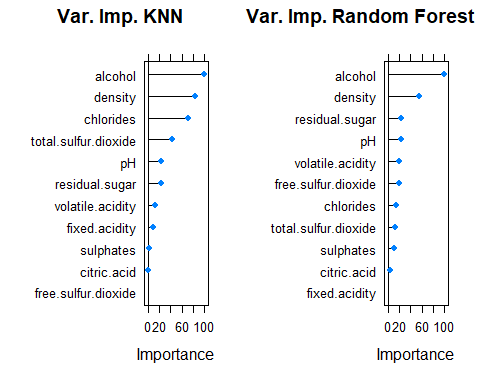
# Variable Importance

plot\_dt <- plot(varimp\_dt, main = "Var. Imp. Decision Tree")  
plot\_nb <- plot(varimp\_nb, main = "Var. Imp. Naive Bayes")  
plot\_knn <-plot(varimp\_knn, main = "Var. Imp. KNN")  
plot\_rf <- plot(varimp\_rf, main = "Var. Imp. Random Forest")  
plot\_gbm <- plot(varimp\_gbm, main = "Var. Imp. Gradient Boost")  
plot\_svm <- plot(varimp\_svm2, main= "Var. Imp. SVM RBF")

grid.arrange(plot\_dt, plot\_nb, nrow=1)



grid.arrange(plot\_knn, plot\_rf, nrow=1)



grid.arrange(plot\_gbm, plot\_svm, nrow=1)

