Wine_Quality_Project_RMD

R.K.Kulkarni

03/03/2022

Introduction

This project is based on Portuguese Vinho Verde wine data collected in 2009 by Cortez et all. The datasets are available from UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/index.php).

The citation reference for using this database is: P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties. In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

The predictor/feature variables based on physicochemical tests:-

1. fixed acidity
2. volatile acidity
3. citric acid
4. residual sugar
5. chlorides
6. free sulfur dioxide
7. total sulfur dioxide
8. density
9. pH
10. sulphates
11. alcohol

The outcome variable (based on sensory data) is quality (integer score between 0 and 10)

The predictor variables have continuous values but the outcome is a factor. The predictors are given with the dataset and are therefore known. The machine learning used is therefore *unsupervised*.

The purpose of this project is to find a way of accurately predicting the *quality* for a given wine based on previous the *quality* value ratings and a combination of two or more the candidate predictors supplied by the dataset given. It was decided that it may be interesting to consider red and white wines separately and together to see if red and white had different predictors affecting quality.

The steps involved were:-

1. downloading the red and white wine csv datasets from the UCI Machine Learning Repository
2. loading into dataframes
3. converting the continuous predictors into factors
4. creating red and white wine dataframes consisting of just factor versions of the predictors and the outcome.
5. merging the the factor versions of red and white dataframes into a single combined wine dataframe.
6. creating training and test partitions from the red, white and combined factor-version datasets.
7. using the rpart (recursive partitioning), rf (random forest), knn (k nearest neighbours) and knn cross-validation algorithms on the red, white and combined training and test sets.
8. Select the algorithmic model that has the highest accuracy ratio.

Methods

The following packages were installed (if required) and libraries loaded:-

1. plyr - to access version number of R in use
2. caret - to provide functions for training datasets
3. tidyverse
4. dplyr - provide data manipualtion functionality eg mutate()
5. rpart - to perform regression tree analyses
6. randomForest - to perform randomforest analyses
7. e1071 - additional functions to support randomForest
8. data.table
9. knitr - to provide tabulation of results

```
# Note: this process may take about two hours
#Start timer
start_time <- Sys.time()
#Process started
start_time
```

```
## [1] "2022-03-02 14:24:26 GMT"
```

```
#set up variable vector for version number
v<-as.numeric(substr(R.Version()$version.string,11,13))
#limit digits displayed to 5
options(digits = 5)
#install packages if not already loaded
if(!require(plyr)) install.packages("plyr", repos = "http://cran.us.r-project.org")
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(knitr)) install.packages("knitr", repos = "http://cran.us.r-project.org")
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
if(!require(data.table)) install.packages("data.table", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
if(!require(e1071)) install.packages("e1071", repos = "http://cran.us.r-project.org")
#if(!require(matrixStats)) install.packages("matrixStats", repos = #"http://cran.us.r-project.org")


library(plyr) ##to access version number of R in use
library(caret) ##to provide functions for training datasets
library(tidyverse)
library(dplyr) ##provide data manipualtion functionality eg mutate()
library(rpart) ##to perform regression tree analyses
library(randomForest) ##to perform randomforest analyses
library(e1071) ##additional functions to support randomForest
library(data.table)
library(knitr) ##to provide tabulation of results
#library(matrixStats)
```

The script imports red & white wines separately by downloading the CSV files directly from the UCI Machine Learning Repository. These are then fed into dataframes. The cut() function is used to convert the continuous predictor variables into factors. During quartiles were chosen in attempt to ensure the number of *levels* in the factors in test and train matched. The red and white datasets are then merged to form a third combined wine data set. Training and test partitions were created for the three sets of data.

As a debugging stage the datasets were exported to XLSX files for inspection. This stage is commented-out in the live script. The dataframes were also tested to see if NA values existed and data cleaning was required.

```r
# Wine

# https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
# https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white.csv

dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.cs

#                 col.names = c("fixedAcidity","volatileAcidity","CitricAcid","residualSugar","Chloride
#Input variables (based on physicochemical tests):
#  1 - fixed acidity
#2 - volatile acidity
#3 - citric acid
#4 - residual sugar
#5 - chlorides
#6 - free sulfur dioxide
#7 - total sulfur dioxide
#8 - density
#9 - pH
#10 - sulphates
#11 - alcohol
#Output variable (based on sensory data):
#  12 - quality (score between 0 and 10)

quantile_ratio <- 0.25 #quartile #0.2 quintile


red_wines_init <- read.csv(dl, sep = ";")
colnames(red_wines_init) <- c("fixedAcidity","volatileAcidity","CitricAcid","residualSugar","Chlorides"

#populate red wines according to version
if (v >= 4.0) { red_wines_init <- as.data.frame(red_wines_init) %>% mutate(
  fixedAcidity = as.numeric(fixedAcidity),
  volatileAcidity = as.numeric(volatileAcidity),
  CitricAcid = as.numeric(CitricAcid),
  residualSugar = as.numeric(residualSugar),
  Chlorides = as.numeric(Chlorides),
  freeSulphurDioxide = as.numeric(freeSulphurDioxide),
  totalSulphurDioxide = as.numeric(totalSulphurDioxide),
  density  = as.numeric(density),
  pH = as.numeric(pH),
  sulphates = as.numeric(sulphates),
  alcohol = as.numeric(alcohol),
  quality = as.character(quality))} else {
    as.data.frame(red_wines_init) %>% mutate(
      fixedAcidity = as.numeric(fixedAcidity),
      volatileAcidity = as.numeric(volatileAcidity),
      CitricAcid = as.numeric(CitricAcid),
      residualSugar = as.numeric(residualSugar),
      Chlorides = as.numeric(Chlorides),
      freeSulphurDioxide = as.numeric(freeSulphurDioxide),
      totalSulphurDioxide = as.numeric(totalSulphurDioxide),
      density  = as.numeric(density),
```

```
      pH = as.numeric(pH),
      sulphates = as.numeric(sulphates),
      alcohol = as.numeric(alcohol),
      quality = as.character(quality))
  }

fact_red_wines<-as.data.frame(red_wines_init) %>% mutate(
  fact_fixedAcidity = cut(fixedAcidity, quantile(fixedAcidity,seq(0,1,quantile_ratio)),include.lowest =
  fact_volatileAcidity = cut(volatileAcidity, quantile(volatileAcidity,seq(0,1,quantile_ratio)),include
  fact_CitricAcid = cut(CitricAcid, quantile(CitricAcid,seq(0,1,quantile_ratio)),include.lowest = TRUE)
  fact_residualSugar = cut(residualSugar, quantile(residualSugar,seq(0,1,quantile_ratio)),include.lowes
  fact_Chlorides = cut(Chlorides, quantile(Chlorides,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_freeSulphurDioxide = cut(freeSulphurDioxide, quantile(freeSulphurDioxide,seq(0,1,quantile_ratio)
  fact_totalSulphurDioxide = cut(totalSulphurDioxide, quantile(totalSulphurDioxide,seq(0,1,quantile_rati
  fact_density  = cut(density, quantile(density,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_pH = cut(pH, quantile(pH,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_sulphates = cut(sulphates, quantile(sulphates,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_alcohol = cut(alcohol, quantile(alcohol,seq(0,1,quantile_ratio)),include.lowest = TRUE))

red_wines<-select(fact_red_wines, c("fact_fixedAcidity","fact_volatileAcidity","fact_CitricAcid","fact_

# Validation set will be 10% of red wine  data
validation_ratio <- 0.1

if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = red_wines$quality, times = 1, p = validation_ratio, list = FALSE)
train_red <- red_wines[-test_index,]
test_red <- red_wines[test_index,]

#white_wines

dl <- tempfile()
download.file("https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-white

white_wines_init <- read.csv(dl, sep = ";") #str_split_fixed(readLines(dl), "\\;;", 12)
colnames(white_wines_init) <- c("fixedAcidity","volatileAcidity","CitricAcid","residualSugar","Chlorides

#populate white wines according to version
if (v >= 4.0) {
  white_wines_init <- as.data.frame(white_wines_init) %>% mutate(
    fixedAcidity = as.numeric(fixedAcidity),
    volatileAcidity = as.numeric(volatileAcidity),
    CitricAcid = as.numeric(CitricAcid),
    residualSugar = as.numeric(residualSugar),
    Chlorides = as.numeric(Chlorides),
    freeSulphurDioxide = as.numeric(freeSulphurDioxide),
    totalSulphurDioxide = as.numeric(totalSulphurDioxide),
    density  = as.numeric(density),
```

```
    pH = as.numeric(pH),
    sulphates = as.numeric(sulphates),
    alcohol = as.numeric(alcohol),
    quality = as.character(quality))} else {
    as.data.frame(white_wines_init) %>% mutate(
        fixedAcidity = as.numeric(fixedAcidity),
        volatileAcidity = as.numeric(volatileAcidity),
        CitricAcid = as.numeric(CitricAcid),
        residualSugar = as.numeric(residualSugar),
        Chlorides = as.numeric(Chlorides),
        freeSulphurDioxide = as.numeric(freeSulphurDioxide),
        totalSulphurDioxide = as.numeric(totalSulphurDioxide),
        density  = as.numeric(density),
        pH = as.numeric(pH),
        sulphates = as.numeric(sulphates),
        alcohol = as.numeric(alcohol),
        quality = as.character(quality))
    }


fact_white_wines<-as.data.frame(white_wines_init) %>% mutate(
  fact_fixedAcidity = cut(fixedAcidity, quantile(fixedAcidity,seq(0,1,quantile_ratio)),include.lowest =
  fact_volatileAcidity = cut(volatileAcidity, quantile(volatileAcidity,seq(0,1,quantile_ratio)),include
  fact_CitricAcid = cut(CitricAcid, quantile(CitricAcid,seq(0,1,quantile_ratio)),include.lowest = TRUE)
  fact_residualSugar = cut(residualSugar, quantile(residualSugar,seq(0,1,quantile_ratio)),include.lowes
  fact_Chlorides = cut(Chlorides, quantile(Chlorides,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_freeSulphurDioxide = cut(freeSulphurDioxide, quantile(freeSulphurDioxide,seq(0,1,quantile_ratio)
  fact_totalSulphurDioxide = cut(totalSulphurDioxide, quantile(totalSulphurDioxide,seq(0,1,quantile_rat
  fact_density  = cut(density, quantile(density,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_pH = cut(pH, quantile(pH,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_sulphates = cut(sulphates, quantile(sulphates,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_alcohol = cut(alcohol, quantile(alcohol,seq(0,1,quantile_ratio)),include.lowest = TRUE))

white_wines<-select(fact_white_wines, c("fact_fixedAcidity","fact_volatileAcidity","fact_CitricAcid","fa

#white_wines<-white_wines%>%filter_all(all_vars(is.finite(.)))

if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = white_wines$quality, times = 1, p = validation_ratio, list = FALSE
test_white <- white_wines[test_index,]
train_white <- white_wines[-test_index,]
####DEBUG
  #nrow(white_wines)
  #nrow(test_white)
  #nrow(train_white)
###
wine_data_init<-rbind(red_wines_init,white_wines_init)
fact_wine_data<-as.data.frame(wine_data_init) %>% mutate(
```

```
  fact_fixedAcidity = cut(fixedAcidity, quantile(fixedAcidity,seq(0,1,quantile_ratio)),include.lowest =
  fact_volatileAcidity = cut(volatileAcidity, quantile(volatileAcidity,seq(0,1,quantile_ratio)),include
  fact_CitricAcid = cut(CitricAcid, quantile(CitricAcid,seq(0,1,quantile_ratio)),include.lowest = TRUE)
  fact_residualSugar = cut(residualSugar, quantile(residualSugar,seq(0,1,quantile_ratio)),include.lowes
  fact_Chlorides = cut(Chlorides, quantile(Chlorides,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_freeSulphurDioxide = cut(freeSulphurDioxide, quantile(freeSulphurDioxide,seq(0,1,quantile_ratio)
  fact_totalSulphurDioxide = cut(totalSulphurDioxide, quantile(totalSulphurDioxide,seq(0,1,quantile_rati
  fact_density  = cut(density, quantile(density,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_pH = cut(pH, quantile(pH,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_sulphates = cut(sulphates, quantile(sulphates,seq(0,1,quantile_ratio)),include.lowest = TRUE),
  fact_alcohol = cut(alcohol, quantile(alcohol,seq(0,1,quantile_ratio)),include.lowest = TRUE))

wine_data<-select(fact_wine_data, c("fact_fixedAcidity","fact_volatileAcidity","fact_CitricAcid","fact_
#wine_data<-white_wines%>%filter_all(all_vars(!is.infinite(.)))

#combined wines
if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
test_index <- createDataPartition(y = wine_data$quality, times = 1, p = validation_ratio, list = FALSE)
train_wine <- wine_data[-test_index,]
test_wine <- wine_data[test_index,]
#Sys.time() - start_time

#DEBUG -output dataframes to XLSX files in working directory (getwd())
#if(!require(tidyverse)) install.packages("writexl", repos = "http://cran.us.r-project.org")
#library("writexl")
#write_xlsx(train_red, "train_red.xlsx")
#write_xlsx(test_red, "test_red.xlsx")
#write_xlsx(train_red, "train_white.xlsx")
#write_xlsx(test_red, "test_white.xlsx")
#write_xlsx(train_red, "train_wine.xlsx")
#write_xlsx(test_red, "test_wine.xlsx")
#write_xlsx(fact_wine_data, "fact_wine_data.xlsx")
#write_xlsx(red_wines_init, "red_wines_init.xlsx")
#write_xlsx(white_wines_init, "white_wines_init.xlsx")
#anyNA(train_red)
#anyNA(train_white)
#anyNA(train_wine)
#anyNA(test_red)
#anyNA(test_white)
#anyNA(test_wine)
#DEBUG END

#Ended populating datasets
#set up variables for numbers of rosw to quote in results section
num_rows_comb <- nrow(wine_data)
num_rows_comb_train <- nrow(train_wine)
num_rows_comb_test <- nrow(test_wine)
num_rows_red <- nrow(red_wines)
```

```
num_rows_red_train <- nrow(train_red)
num_rows_red_test <- nrow(test_red)
num_rows_white <- nrow(white_wines)
num_rows_white_train <- nrow(train_white)
num_rows_white_test <- nrow(test_white)
```

The following algorithmic models were applied in turn to each of the sets of wine data (combined set, red wine and then white wine):-

1. RPart (recursive partitioning) - this would produce a decision tree that indicates the predictors that are important in any prediction model and yield prediction accuracy estimates.
2. RandomForest - this should list the predictors in descending order of importance and yield prediction accuracy estimates.
3. knn (k nearest neighbours) - this should yield prediction
4. knn-cv (k nearest neighbours cross-validate) - this should yield prediction accuracy estimates.

```
#Process algorithms for Combined, Red, White wine datasets in succession

#
#Combined wine data processing
#

#rpart combined
train_rpart_combined <- train(quality ~ .,
                      method = "rpart",
                      tuneGrid = data.frame(cp = seq(0, 0.01, 0.0001)), data = train_wine)
train_rpart_combined$bestTune
```

```
##        cp
## 22 0.0021
```

```
y_hat_rpart_combined <- predict(train_rpart_combined, test_wine)
rpart_comb_acc<-confusionMatrix(factor(as.integer(y_hat_rpart_combined)), factor(test_wine$quality))$ov
```

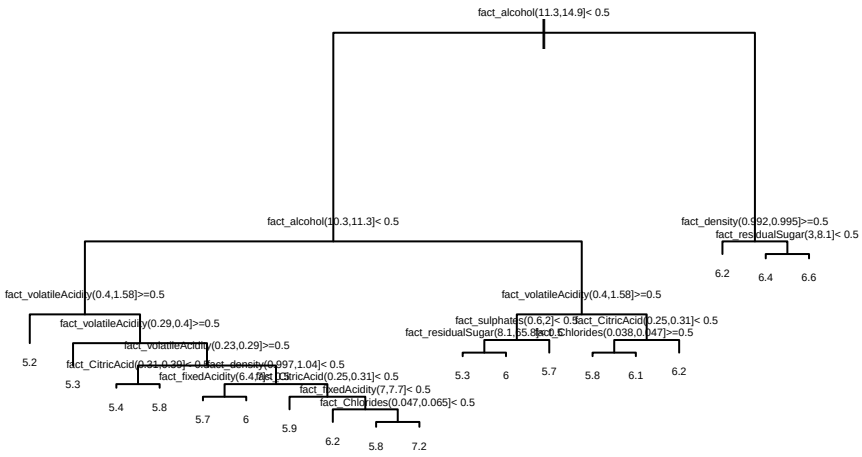
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_rpart_combined)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
rpart_comb_acc
```

```
## Accuracy
##  0.48387
```

```
plot(train_rpart_combined$finalModel, margin = 0.1)
text(train_rpart_combined$finalModel, cex = 0.3) #cex=0.75
```



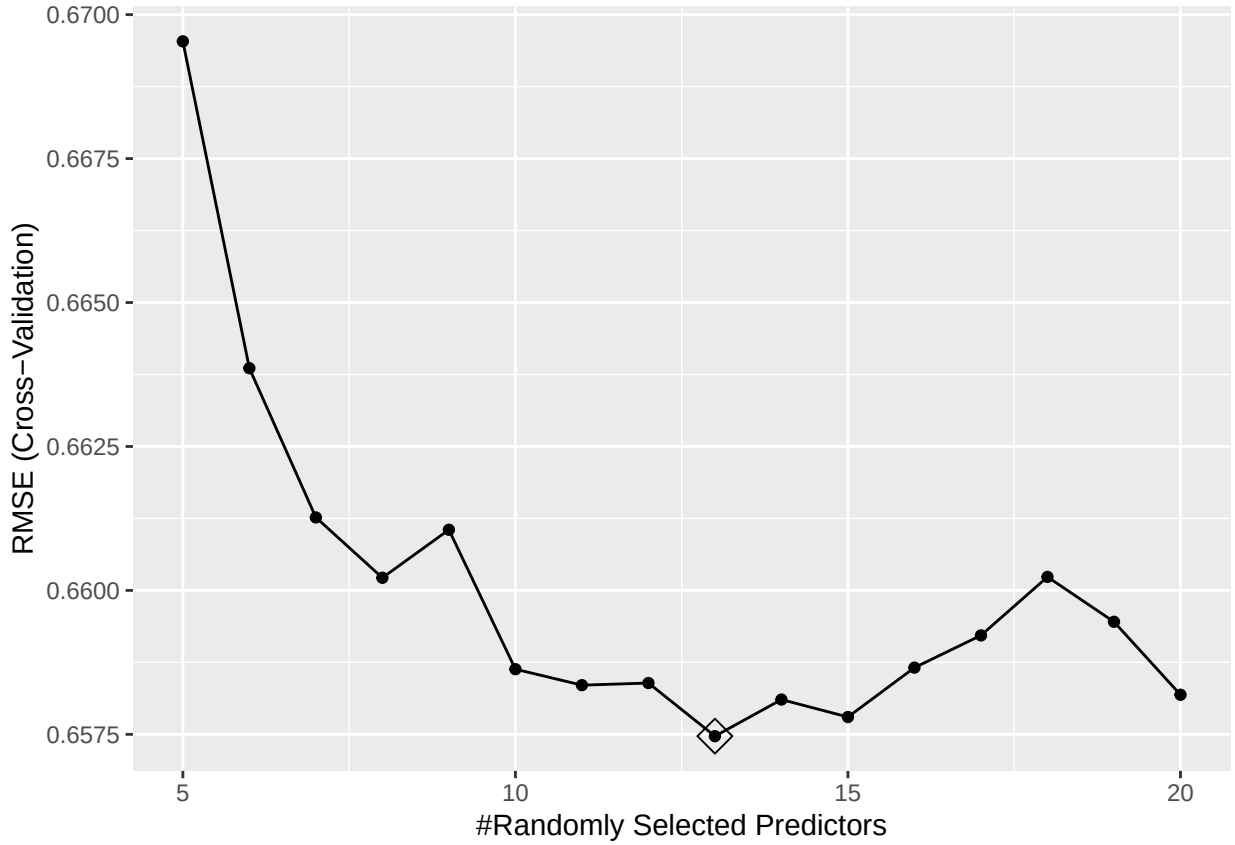

```
#rpart finished
#Sys.time() - start_time

if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#rf combined

control <- trainControl(method="cv", number = 10)
train_rf_combined <- train(quality ~ .,
                  method = "rf",
                  ntree = 150,
                  trControl = control,
                  tuneGrid=data.frame(mtry = seq(5,20)), #seq(5,15)
                  data = train_wine)
ggplot(train_rf_combined, highlight=TRUE)
```




```
train_rf_combined$bestTune
```

```
##   mtry
## 9   13
```

```
#
imp_combined<-varImp(train_rf_combined)
imp_combined
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                 Overall
## fact_alcohol(11.3,14.9]          100.00
## fact_alcohol(10.3,11.3]           24.01
## fact_volatileAcidity(0.4,1.58]    22.22
## fact_volatileAcidity(0.29,0.4]    13.45
## fact_sulphates(0.6,2]             13.08
## fact_fixedAcidity(7,7.7]          12.48
## fact_volatileAcidity(0.23,0.29]   11.29
## fact_CitricAcid(0.25,0.31]        10.96
## fact_pH(3.32,4.01]                10.84
## fact_residualSugar(3,8.1]         10.42
## fact_pH(3.21,3.32]                 9.79
## fact_freeSulphurDioxide(29,41]     9.46
## fact_pH(3.11,3.21]                 9.41
```

```
## fact_fixedAcidity(6.4,7]           9.32
## fact_freeSulphurDioxide(17,29]     8.81
## fact_sulphates(0.43,0.51]          8.46
## fact_CitricAcid(0.31,0.39]         7.98
## fact_residualSugar(1.8,3]          7.59
## fact_sulphates(0.51,0.6]           7.51
## fact_Chlorides(0.038,0.047]        7.45
```

```
y_hat_rf_combined <- predict(train_rf_combined, test_wine)
rf_comb_acc<-confusionMatrix(factor(as.integer(y_hat_rf_combined)),
                factor(test_wine$quality))$overall["Accuracy"]
```

```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_rf_combined)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
rf_comb_acc
```

```
## Accuracy
##  0.46083
```

```
#knn combined
if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
train_knn_combined <- train(quality ~ ., method = "knn", data = train_wine,tuneGrid = data.frame(k = se
y_hat_knn_combined <- predict(train_knn_combined, test_wine)
knn_comb_acc<-confusionMatrix(factor(as.integer(y_hat_knn_combined)),
                factor(test_wine$quality))$overall["Accuracy"]
```

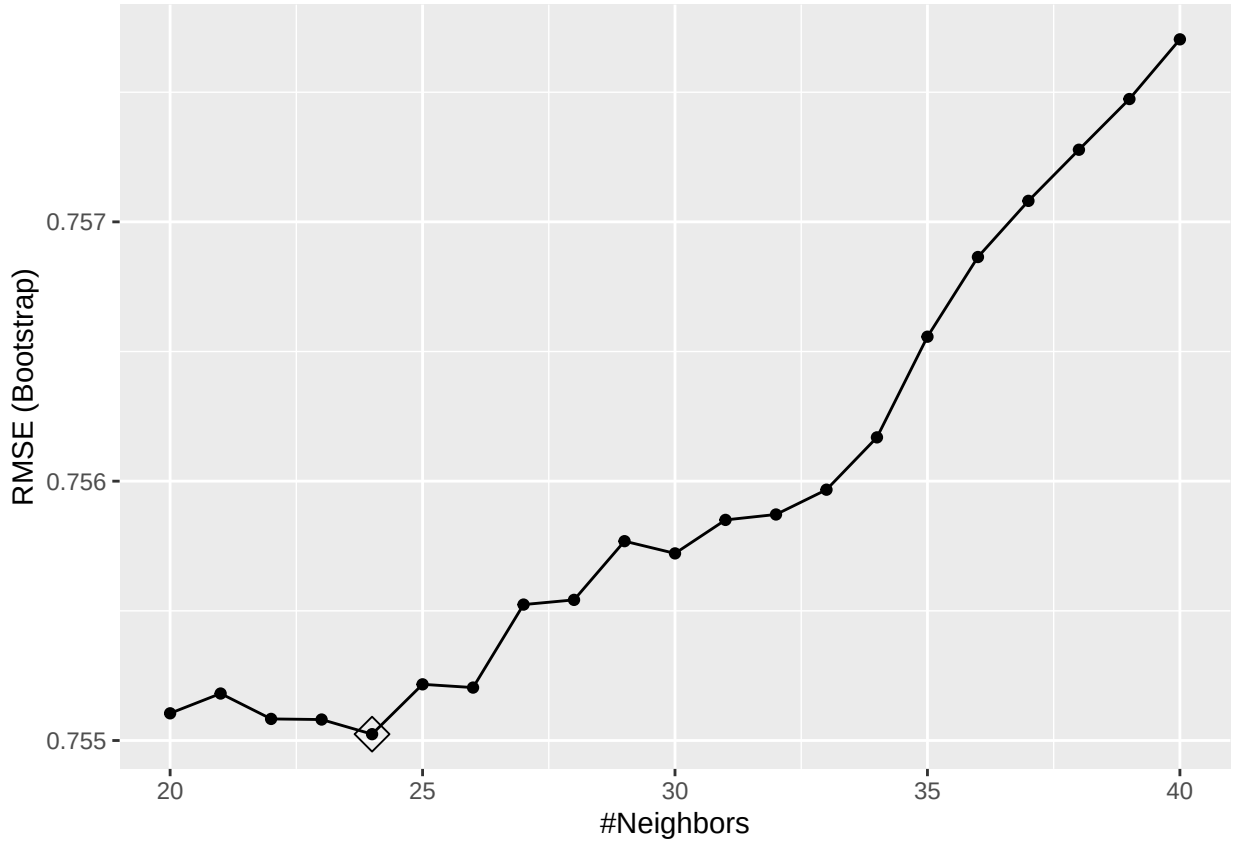
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_knn_combined)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
knn_comb_acc
```

```
## Accuracy
##  0.48694
```

```
ggplot(train_knn_combined,highlight=TRUE)
```




```
train_knn_combined$bestTune
```

```
##    k
## 5 24
```

```
#Sys.time() - start_time
#knn_cv combined
if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn_cv_combined <- train(quality ~ ., method = "knn", data = train_wine,tuneGrid = data.frame(k =
                               trControl = control)
y_hat_knn_cv_combined <- predict(train_knn_cv_combined, test_wine)
knn_cv_comb_acc<-confusionMatrix(factor(as.integer(y_hat_knn_cv_combined)),
                factor(test_wine$quality))$overall["Accuracy"]
```

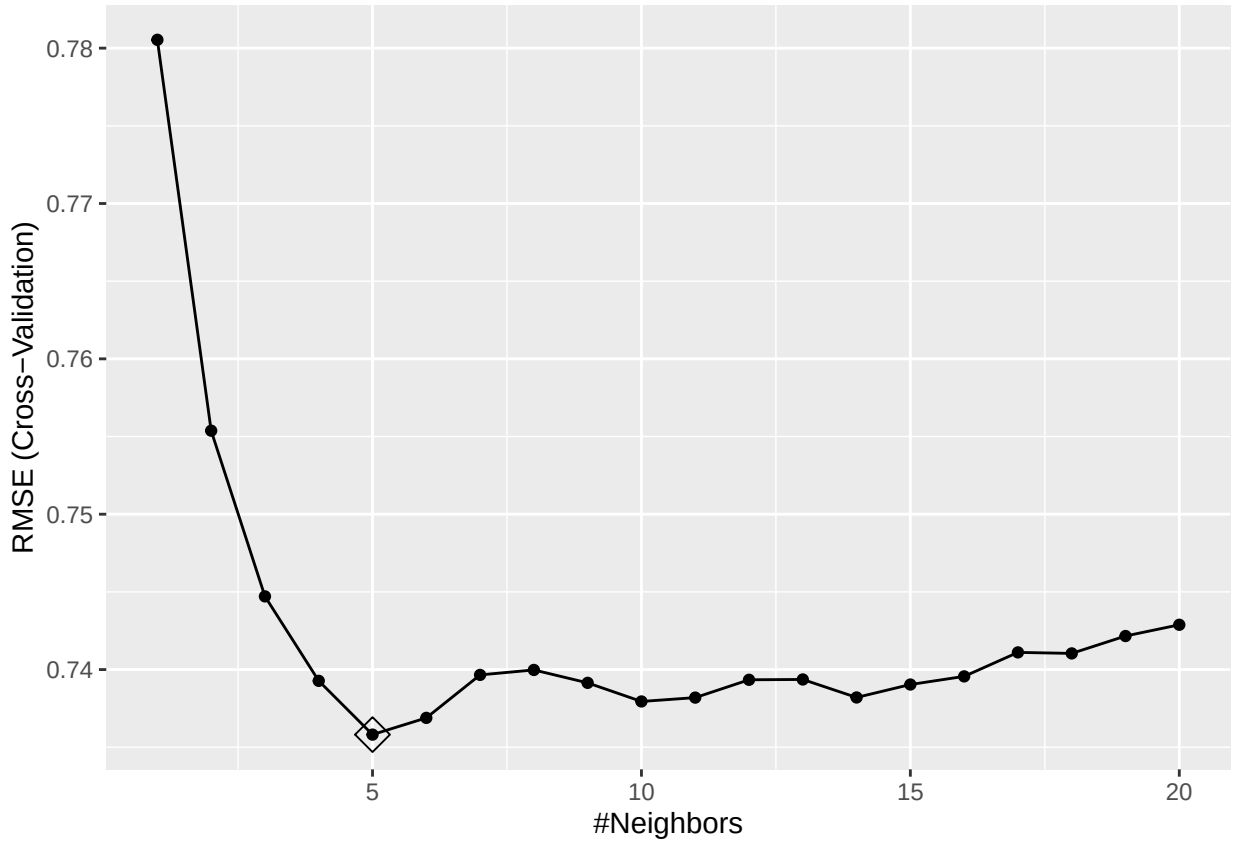
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_knn_cv_combined)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
knn_cv_comb_acc
```

```
## Accuracy
##   0.4977
```

```
ggplot(train_knn_cv_combined,highlight=TRUE)
```




```
train_knn_cv_combined$bestTune
```

```
##   k
## 5 5
```

```
#Sys.time() - start_time
#finished knn_cv combined
```

```
#
#red_wines
#

#rpart red
train_rpart_red <- train(quality ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0, 0.01, 0.0001)),
                     data = train_red)
train_rpart_red$bestTune
```

```
##        cp
## 98 0.0097
```

```
y_hat_rpart_red <- predict(train_rpart_red, test_red)
rpart_red_acc<-confusionMatrix(factor(as.integer(y_hat_rpart_red)),
               factor(test_red$quality))$overall["Accuracy"]
```

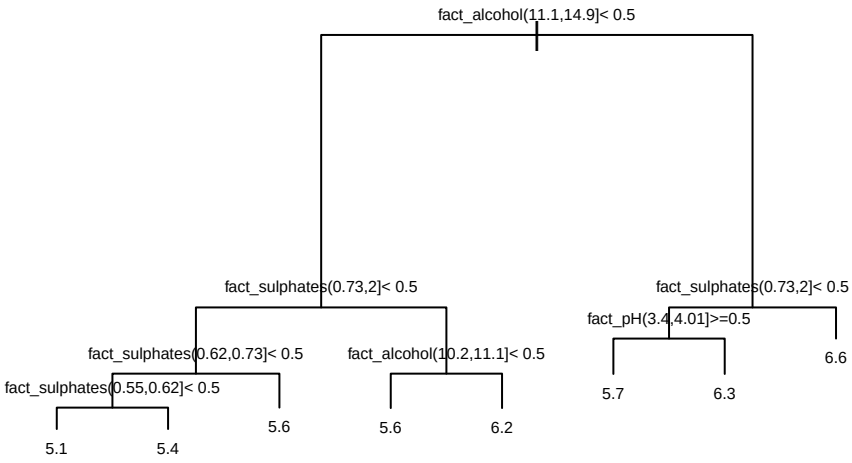
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_rpart_red)), : Levels
## are not in the same order for reference and data. Refactoring data to match.
```

```
rpart_red_acc
```

```
## Accuracy
##  0.51553
```

```
plot(train_rpart_red$finalModel, margin = 0.1)
text(train_rpart_red$finalModel, cex = 0.5)
```



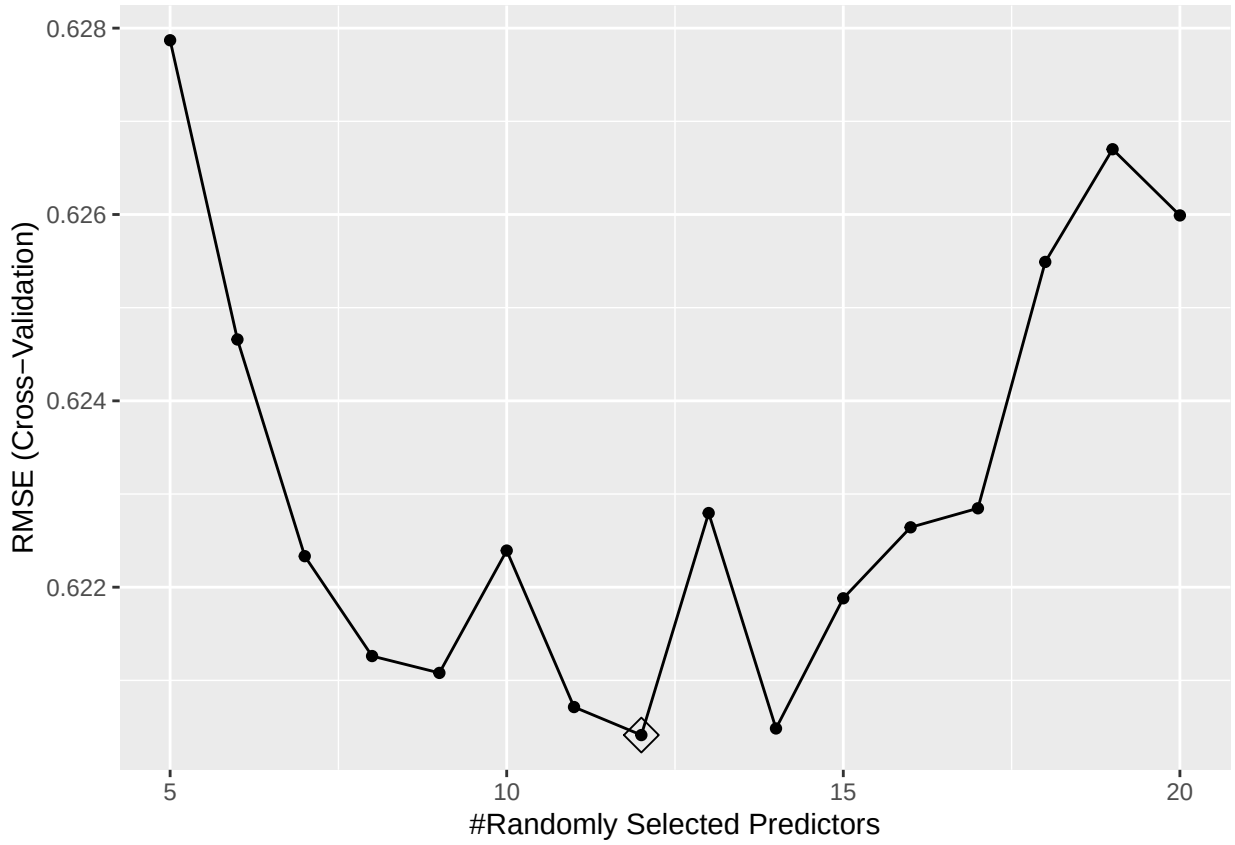

```
#rpart finished
#Sys.time() - start_time

if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#rf red

control <- trainControl(method="cv", number = 10)
train_rf_red <- train(quality ~ .,
                  method = "rf",
                  ntree = 150,
                  trControl = control,
                  tuneGrid=data.frame(mtry = seq(5,20)),
                  data = train_red)
ggplot(train_rf_red,highlight=TRUE)
```




```
train_rf_red$bestTune
```

```
##   mtry
## 8   12
```

```
imp_rf_red<-varImp(train_rf_red)
imp_rf_red
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                  Overall
## fact_alcohol(11.1,14.9]           100.00
## fact_sulphates(0.73,2]             38.47
## fact_volatileAcidity(0.64,1.58]    19.66
```

```
## fact_alcohol(10.2,11.1]            14.59
## fact_sulphates(0.62,0.73]          13.42
## fact_totalSulphurDioxide(62,289]   12.43
## fact_Chlorides(0.09,0.611]          8.66
## fact_pH(3.4,4.01]                   8.54
## fact_freeSulphurDioxide(7,14]       8.23
## fact_sulphates(0.55,0.62]           8.21
## fact_residualSugar(1.9,2.2]         7.74
## fact_totalSulphurDioxide(22,38]     7.02
## fact_CitricAcid(0.09,0.26]          6.31
## fact_fixedAcidity(7.1,7.9]          6.18
## fact_volatileAcidity(0.39,0.52]     5.98
## fact_residualSugar(2.2,2.6]         5.96
## fact_volatileAcidity(0.52,0.64]     5.33
## fact_Chlorides(0.07,0.079]          4.72
## fact_alcohol(9.5,10.2]              4.15
## fact_density(0.997,0.998]           3.81
```

```
y_hat_rf_red <- predict(train_rf_red, test_red)
rf_red_acc<-confusionMatrix(factor(as.integer(y_hat_rf_red)),
                factor(test_red$quality))$overall["Accuracy"]
```

```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_rf_red)),
## factor(test_red$quality)): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
rf_red_acc
```

```
## Accuracy
##  0.48447
```

```
#knn red
train_knn_red <- train(quality ~ ., method = "knn", data = train_red,tuneGrid = data.frame(k = seq(30, 5
y_hat_knn_red <- predict(train_knn_red, test_red)
knn_red_acc<-confusionMatrix(factor(as.integer(y_hat_knn_red)),
                factor(test_red$quality))$overall["Accuracy"]
```

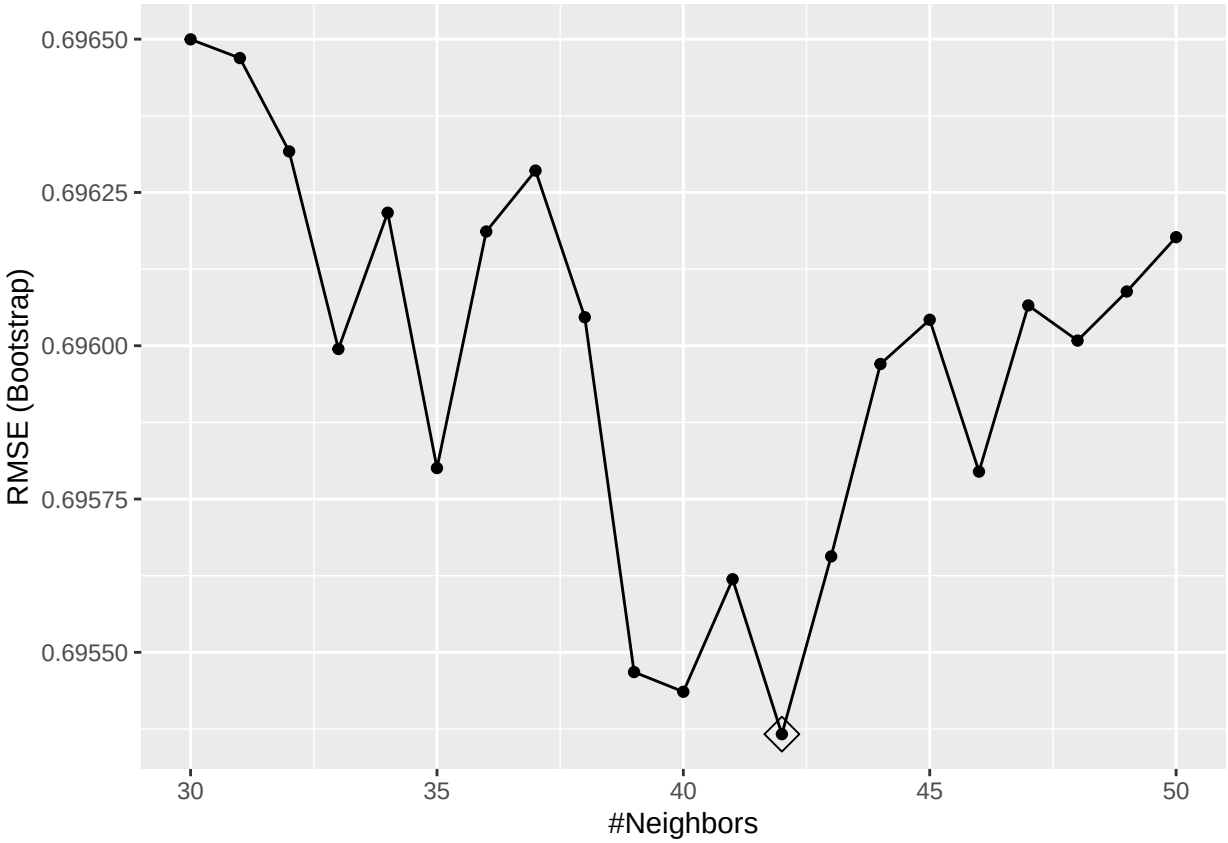
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_knn_red)),
## factor(test_red$quality)): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
knn_red_acc
```

```
## Accuracy
##  0.46584
```

```
ggplot(train_knn_red,highlight=TRUE)
```




```
train_knn_red$bestTune
```

```
##     k
## 13 42
```

```
#Sys.time() - start_time
#knn_cv red
if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn_cv_red <- train(quality ~ ., method = "knn", data = train_red,tuneGrid = data.frame(k = seq(5
                          trControl = control)
y_hat_knn_cv_red <- predict(train_knn_cv_red, test_red)
knn_cv_red_acc<-confusionMatrix(factor(as.integer(y_hat_knn_cv_red)),
                factor(test_red$quality))$overall["Accuracy"]
```

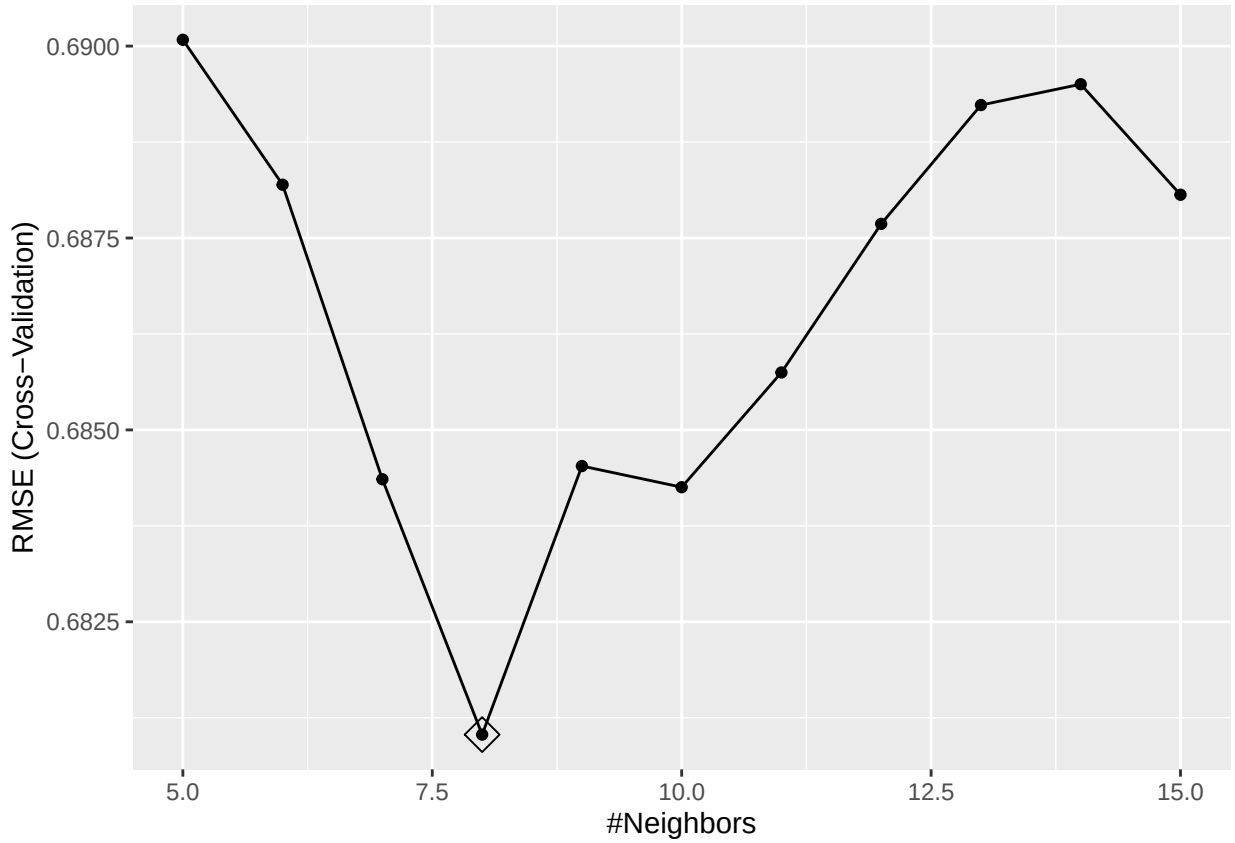
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_knn_cv_red)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
knn_cv_red_acc
```

```
## Accuracy
##  0.49068
```

```
ggplot(train_knn_cv_red,highlight=TRUE)
```




```
train_knn_cv_red$bestTune
```

```
##   k
## 4 8
```

```
#Sys.time() - start_time
#finished knn_cv red
```

```
#
#white wines
#

#rpart white
train_rpart_white <- train(quality ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0, 0.01, 0.0001)),
                     data = train_white)
train_rpart_white$bestTune
```

```
##        cp
## 39 0.0038
```

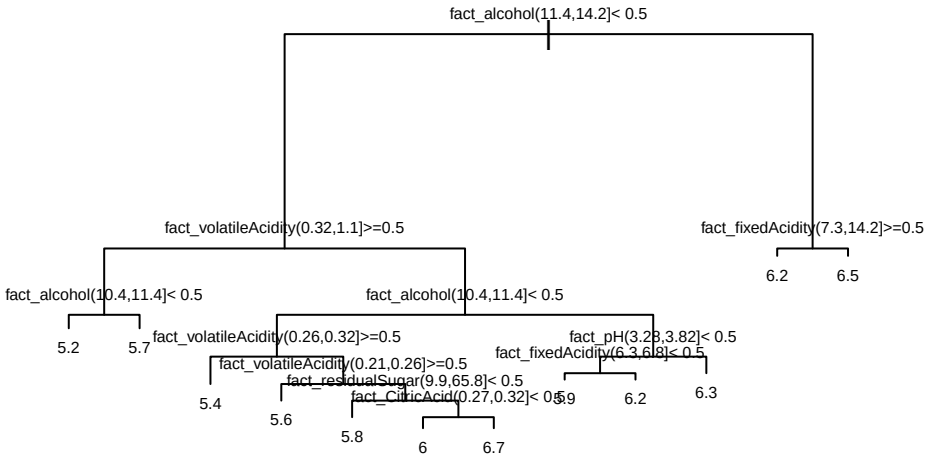
```
y_hat_rpart_white <- predict(train_rpart_white, test_white)
rpart_white_acc<-confusionMatrix(factor(as.integer(y_hat_rpart_white)),
                factor(test_white$quality))$overall["Accuracy"]
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_rpart_white)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
rpart_white_acc
```

```
## Accuracy
##   0.4102
```

```
plot(train_rpart_white$finalModel, margin = 0.1)
text(train_rpart_white$finalModel, cex = 0.5)
```



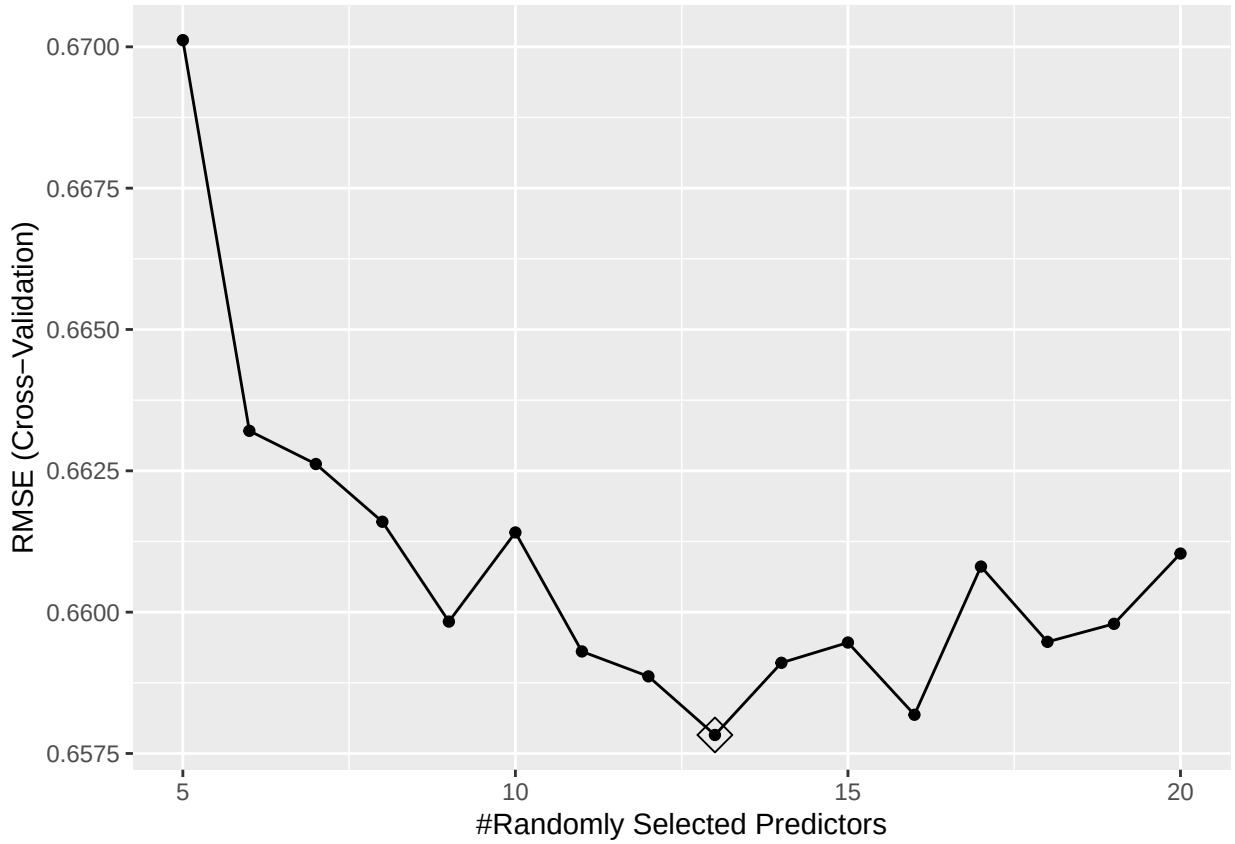

```
#rpart finished
#Sys.time() - start_time

if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
#rf white
control <- trainControl(method="cv", number = 10)
```

```
train_rf_white <- train(quality ~ .,
                  method = "rf",
                  ntree = 150,
                  trControl = control,
                  tuneGrid=data.frame(mtry = seq(5,20)),
                  data = train_white)
ggplot(train_rf_white,highlight=TRUE)
```




```
train_rf_white$bestTune
```

```
##   mtry
## 9   13
```

```
imp_rf_white<-varImp(train_rf_white)
imp_rf_white
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                   Overall
## fact_alcohol(11.4,14.2]            100.00
## fact_volatileAcidity(0.32,1.1]      33.37
## fact_alcohol(10.4,11.4]             18.60
## fact_CitricAcid(0.27,0.32]          15.68
## fact_volatileAcidity(0.26,0.32]     14.30
## fact_Chlorides(0.05,0.346]          13.25
```

```
## fact_pH(3.28,3.82]                  12.31
## fact_freeSulphurDioxide(23,34]      11.20
## fact_freeSulphurDioxide(34,46]      10.77
## fact_totalSulphurDioxide(167,440]   10.45
## fact_fixedAcidity(6.8,7.3]          10.31
## fact_fixedAcidity(7.3,14.2]         10.29
## fact_residualSugar(1.7,5.2]          9.69
## fact_CitricAcid(0.32,0.39]           9.65
## fact_volatileAcidity(0.21,0.26]      9.31
## fact_sulphates(0.41,0.47]            9.01
## fact_sulphates(0.55,1.08]            8.59
## fact_totalSulphurDioxide(108,134]    8.31
## fact_sulphates(0.47,0.55]            8.01
## fact_CitricAcid(0.39,1.66]           7.84
```

```
y_hat_rf_white <- predict(train_rf_white, test_white)
rf_white_acc<-confusionMatrix(factor(as.integer(y_hat_rf_white)),
                factor(test_white$quality))$overall["Accuracy"]
```

```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_rf_white)),
## factor(test_white$quality)): Levels are not in the same order for reference and
## data. Refactoring data to match.
```

```
rf_white_acc
```

```
## Accuracy
##  0.45306
```

```
#knn white
train_knn_white <- train(quality ~ ., method = "knn", data = train_white,tuneGrid = data.frame(k = seq(
y_hat_knn_white <- predict(train_knn_white, test_white)
knn_white_acc<-confusionMatrix(factor(as.integer(y_hat_knn_white)),
                factor(test_white$quality))$overall["Accuracy"]
```

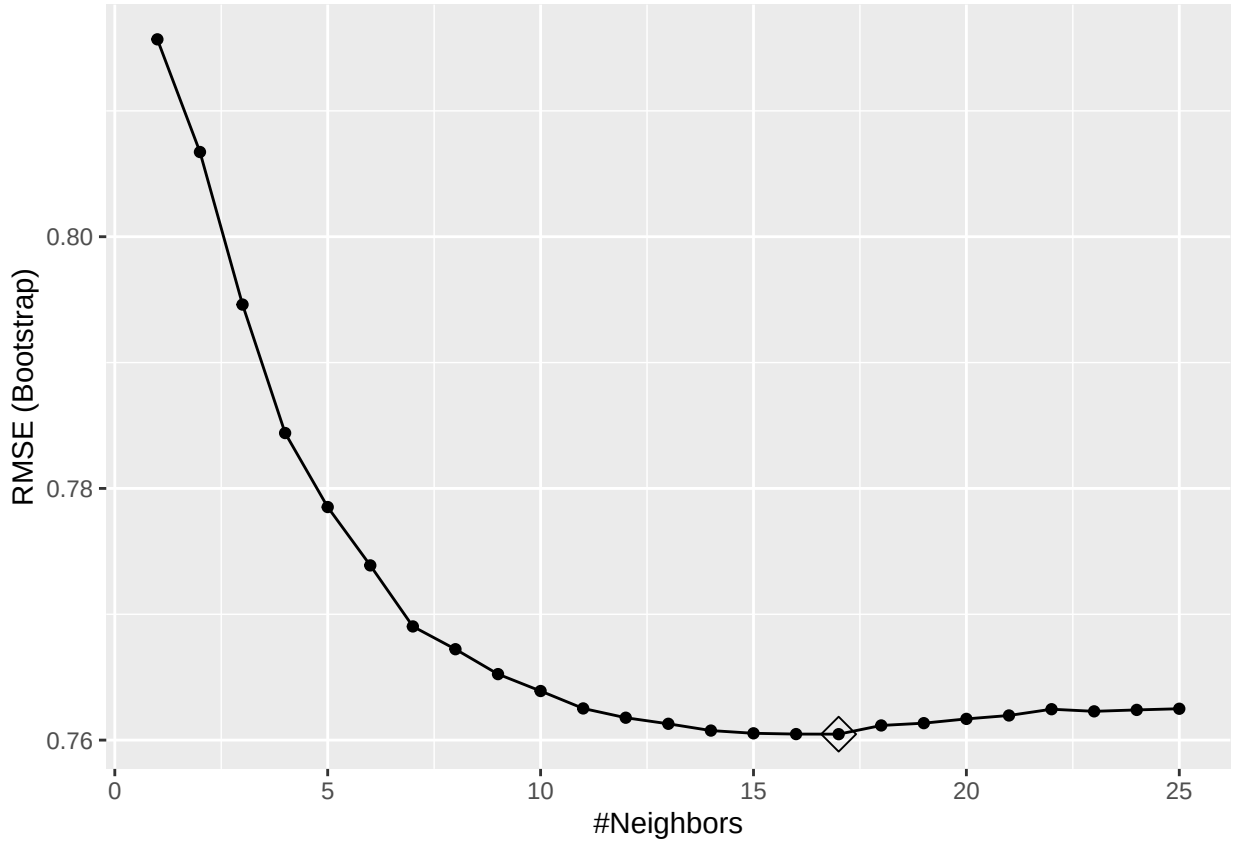
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_knn_white)), : Levels
## are not in the same order for reference and data. Refactoring data to match.
```

```
knn_white_acc
```

```
## Accuracy
##  0.41429
```

```
ggplot(train_knn_white,highlight=TRUE)
```




```
train_knn_white$bestTune
```

```
##     k
## 17 17
```

```
#Sys.time() - start_time
#knn_cv white
if (v <= 3.5) {
  set.seed(1)} else {
    set.seed(1, sample.kind="Rounding")}
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn_cv_white <- train(quality ~ ., method = "knn", data = train_white,tuneGrid = data.frame(k = s
                      trControl = control)
y_hat_knn_cv_white <- predict(train_knn_cv_white, test_white)
knn_cv_white_acc<-confusionMatrix(factor(as.integer(y_hat_knn_cv_white)),
                factor(test_white$quality))$overall["Accuracy"]
```

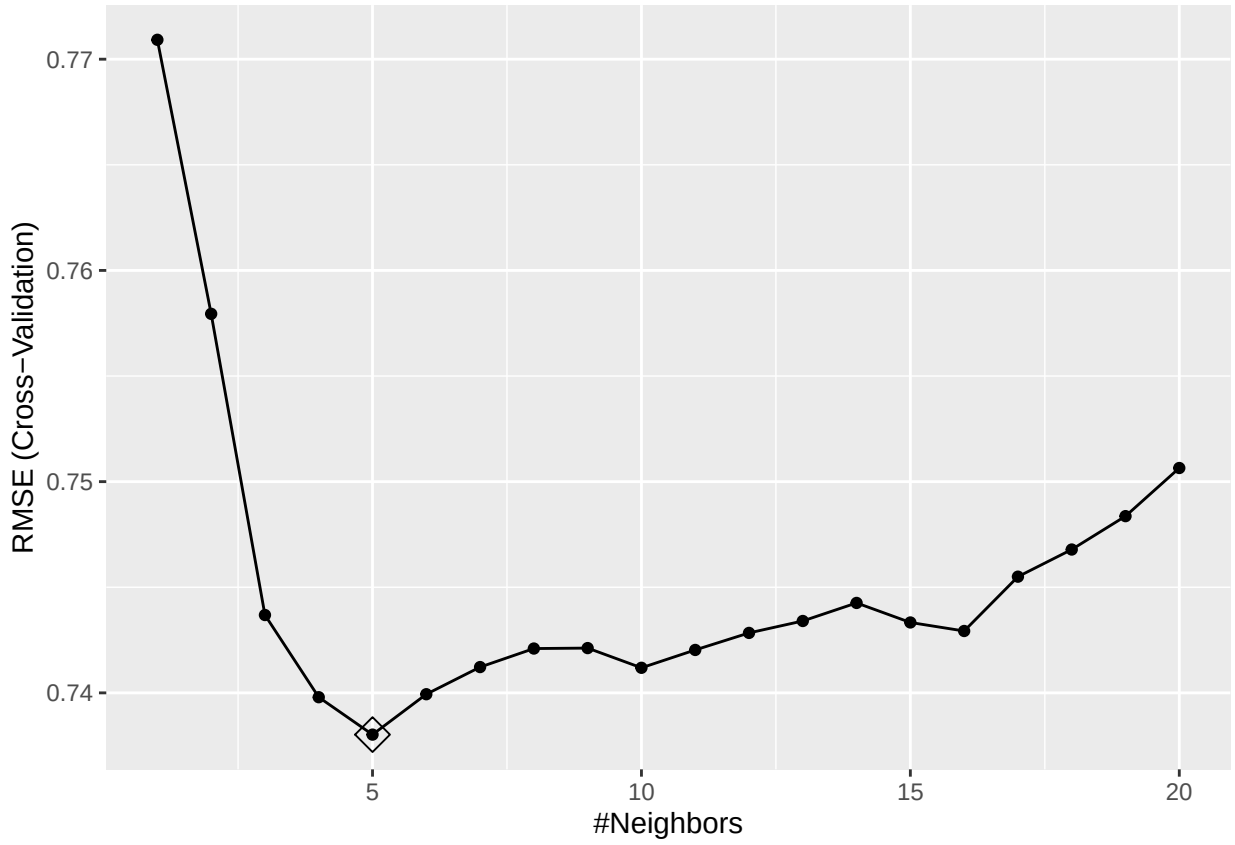
```
## Warning in levels(reference) != levels(data): longer object length is not a
## multiple of shorter object length
```

```
## Warning in confusionMatrix.default(factor(as.integer(y_hat_knn_cv_white)), :
## Levels are not in the same order for reference and data. Refactoring data to
## match.
```

```
knn_cv_white_acc
```

```
## Accuracy
##  0.43878
```

```
ggplot(train_knn_cv_white,highlight=TRUE)
```




```
train_knn_cv_white$bestTune
```

```
##   k
## 5 5
```

```
#Sys.time() - start_time
#finished knn_cv white
```

##Results

```
#summary rows of data
row_total_summary<-data.frame(Wine_set="Combined Complete",RowCount=num_rows_comb)
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="Combined Training Set",RowCount=num
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="Combined Test Set",RowCount=num_rows
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="Red Wine Complete Set",RowCount=num
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="Red Wine Training Set",RowCount=num
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="Red Wine Test Set",RowCount=num_rows
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="White Wine Complete Set",RowCount=nu
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="White Wine Training Set",RowCount=nu
row_total_summary<-bind_rows(row_total_summary,data.frame(Wine_set="White Wine Test Set",RowCount=num_r
```

```
row_total_summary%>%knitr::kable(row.names = NA, caption = "Rows per data frame")
```

Table 1: Rows per data frame

Wine_set	RowCount
Combined Complete	6497
Combined Training Set	5846
Combined Test Set	651
Red Wine Complete Set	1599
Red Wine Training Set	1438
Red Wine Test Set	161
White Wine Complete Set	1599
White Wine Training Set	4408
White Wine Test Set	490

```
#Summary accuracy output
accuracy_summary<-data.frame(Algorithm="RPart",Wine_set="Combined",Accuracy=rpart_comb_acc)
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="randomForest",Wine_set="Combined",Ac
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="knn",Wine_set="Combined",Accuracy=kn
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="knn cv",Wine_set="Combined",Accuracy
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="RPart",Wine_set="Red",Accuracy=rpart
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="randomForest",Wine_set="Red",Accurac
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="knn",Wine_set="Red",Accuracy=knn_red
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="knn cv",Wine_set="Red",Accuracy=knn_
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="RPart",Wine_set="White",Accuracy=rpa
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="randomForest",Wine_set="White",Accura
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="knn",Wine_set="White",Accuracy=knn_wh
accuracy_summary<-bind_rows(accuracy_summary,data.frame(Algorithm="knn cv",Wine_set="White",Accuracy=kn
accuracy_summary%>%knitr::kable(row.names = NA, caption = "Accuracies per algorithmic model")
```

Table 2: Accuracies per algorithmic model

	Algorithm	Wine_set	Accuracy
Accuracy...1	RPart	Combined	0.48387
Accuracy...2	randomForest	Combined	0.46083
Accuracy...3	knn	Combined	0.48694
Accuracy...4	knn cv	Combined	0.49770
Accuracy...5	RPart	Red	0.51553
Accuracy...6	randomForest	Red	0.48447
Accuracy...7	knn	Red	0.46584
Accuracy...8	knn cv	Red	0.49068
Accuracy...9	RPart	White	0.41020
Accuracy...10	randomForest	White	0.45306
Accuracy...11	knn	White	0.41429
Accuracy...12	knn cv	White	0.43878

```
#
imp_combined
```

```
## rf variable importance
##
```

```
##   only 20 most important variables shown (out of 33)
##
##                                 Overall
## fact_alcohol(11.3,14.9]          100.00
## fact_alcohol(10.3,11.3]           24.01
## fact_volatileAcidity(0.4,1.58]    22.22
## fact_volatileAcidity(0.29,0.4]    13.45
## fact_sulphates(0.6,2]             13.08
## fact_fixedAcidity(7,7.7]          12.48
## fact_volatileAcidity(0.23,0.29]   11.29
## fact_CitricAcid(0.25,0.31]        10.96
## fact_pH(3.32,4.01]                10.84
## fact_residualSugar(3,8.1]         10.42
## fact_pH(3.21,3.32]                 9.79
## fact_freeSulphurDioxide(29,41]     9.46
## fact_pH(3.11,3.21]                 9.41
## fact_fixedAcidity(6.4,7]           9.32
## fact_freeSulphurDioxide(17,29]     8.81
## fact_sulphates(0.43,0.51]          8.46
## fact_CitricAcid(0.31,0.39]         7.98
## fact_residualSugar(1.8,3]          7.59
## fact_sulphates(0.51,0.6]           7.51
## fact_Chlorides(0.038,0.047]        7.45
```

```
#
imp_rf_red
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                Overall
## fact_alcohol(11.1,14.9]         100.00
## fact_sulphates(0.73,2]           38.47
## fact_volatileAcidity(0.64,1.58]  19.66
## fact_alcohol(10.2,11.1]          14.59
## fact_sulphates(0.62,0.73]        13.42
## fact_totalSulphurDioxide(62,289] 12.43
## fact_Chlorides(0.09,0.611]        8.66
## fact_pH(3.4,4.01]                 8.54
## fact_freeSulphurDioxide(7,14]     8.23
## fact_sulphates(0.55,0.62]         8.21
## fact_residualSugar(1.9,2.2]       7.74
## fact_totalSulphurDioxide(22,38]   7.02
## fact_CitricAcid(0.09,0.26]        6.31
## fact_fixedAcidity(7.1,7.9]        6.18
## fact_volatileAcidity(0.39,0.52]   5.98
## fact_residualSugar(2.2,2.6]       5.96
## fact_volatileAcidity(0.52,0.64]   5.33
## fact_Chlorides(0.07,0.079]        4.72
## fact_alcohol(9.5,10.2]            4.15
## fact_density(0.997,0.998]         3.81
```

```
#
imp_rf_white
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                    Overall
## fact_alcohol(11.4,14.2]             100.00
## fact_volatileAcidity(0.32,1.1]       33.37
## fact_alcohol(10.4,11.4]              18.60
## fact_CitricAcid(0.27,0.32]           15.68
## fact_volatileAcidity(0.26,0.32]      14.30
## fact_Chlorides(0.05,0.346]           13.25
## fact_pH(3.28,3.82]                   12.31
## fact_freeSulphurDioxide(23,34]       11.20
## fact_freeSulphurDioxide(34,46]       10.77
## fact_totalSulphurDioxide(167,440]    10.45
## fact_fixedAcidity(6.8,7.3]           10.31
## fact_fixedAcidity(7.3,14.2]          10.29
## fact_residualSugar(1.7,5.2]           9.69
## fact_CitricAcid(0.32,0.39]            9.65
## fact_volatileAcidity(0.21,0.26]       9.31
## fact_sulphates(0.41,0.47]             9.01
## fact_sulphates(0.55,1.08]             8.59
## fact_totalSulphurDioxide(108,134]     8.31
## fact_sulphates(0.47,0.55]             8.01
## fact_CitricAcid(0.39,1.66]            7.84
#
```

It can be seen from the numbers of rows in each of the dataframes that the number of observations be used in the models is low.

The accuracies obtained were in the approximate range 40-50% and not as high as required for a reasonable model (possibly 80% or more). More data should also be gathered. The process run time was high (approximately 2 hours).

The Rpart decision trees yielded some possible predictors to consider:-

1. for the combined set - alcohol, density, volatile acidity and sulphates.
2. for red wines - alcohol, sulphates and pH
3. for white wines - alcohol, volatile acidity, fixed acidity, pH, residual sugars and citric acid. Alcohol is the only common predictor hence it may be better to treat red and white wines separately.

The randomForest variable importances with scores above 10% were:-

1. for the combined set - alcohol, volatile acidity, sulphates, fixed acidity, citric acid, pH and residual sugars.
2. for red wine - alcohol, sulphates, volatile acidity, sulphates and total sulphur dioxide.
3. for white wine - alcohol, volatile acidity, citric acid, chlorides, pH, free sulphur dioxide, fixed sulphur dioxide and fixed acidity. Alcohol and volatile acidity are the only common predictors hence treating red and white wines separately is indicated again.

In all cases the Confusion Matrix calls generated warnings that the number of levels in the data did not match the number of levels in the reference. These warnings could not be fixed despite reference to internet forums and there was not sufficient time available to pursue this issue.

##Conclusion and Recommendations

The Vino Verde wine data supplied by Cortez et al. (2009) was downloaded and imported into R dataframes for analysis. The data for red wines and white wines were investigated separately and as a combined set.

The algorithmic models used for the three sets of data were: RPart, random forests, knn and knn-cv. None were found to have an accuracy greater than about 50%. The data sets were however small compared to for example the Movielens dataset. Also the warning message concerning the mismatch in the number of levels in data and reference sets should be heeded. There was unfortunately insufficient time to resolve this issue. It was however noted on inspecting the variable importance output from the random forest analysis that red and white wines should be treated separately. Further work is therefore required possibly using larger datasets. Other algorithmic techniques could also be investigated or possibly using Python instead of R.

```
## [1] "2022-03-02 14:24:26 GMT"
```

```
## [1] "2022-03-02 16:27:20 GMT"
```

```
## Time difference of 2.0483 hours
```