Additional task. Report

Contents

Introduction

The data set that will be introduced and analyzed below is about the two types of wines: white and red. Using 11 physicochemical characteristics of the liquid, the wine will be decided whether it is white or red.

The classification methods will be operated to find the model to divide the wine into two groups. The following methods will be used:

1. Logistic regression;
2. Linear discriminant analysis
3. Classification trees;
4. Random forest
5. Gradient Boosting

Also, the methods will be compared to find the most appropriate model for classification types of wines.

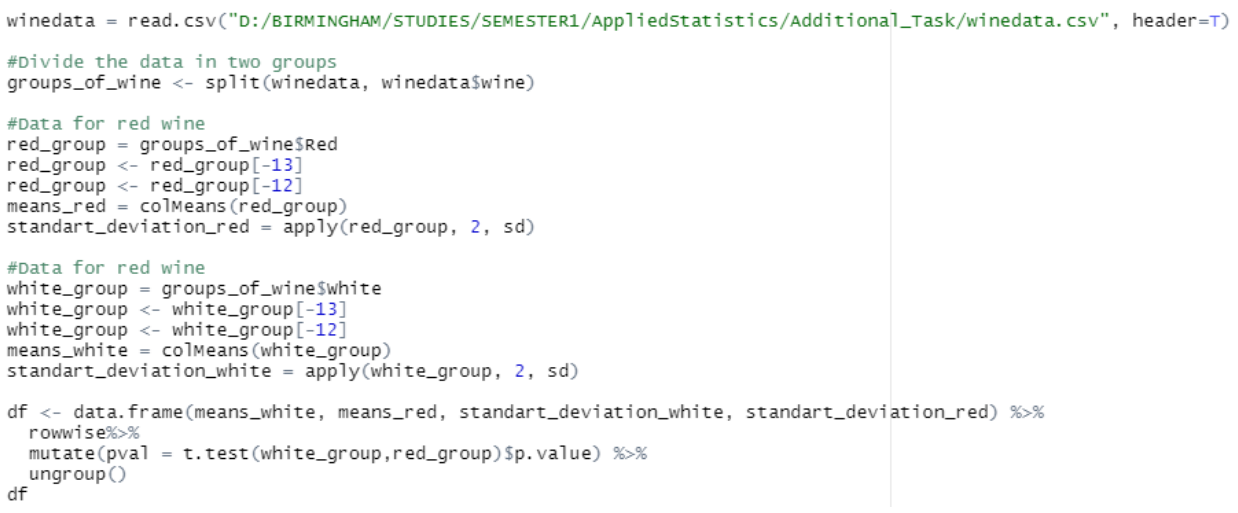
Methods and Analysis

Initial Data

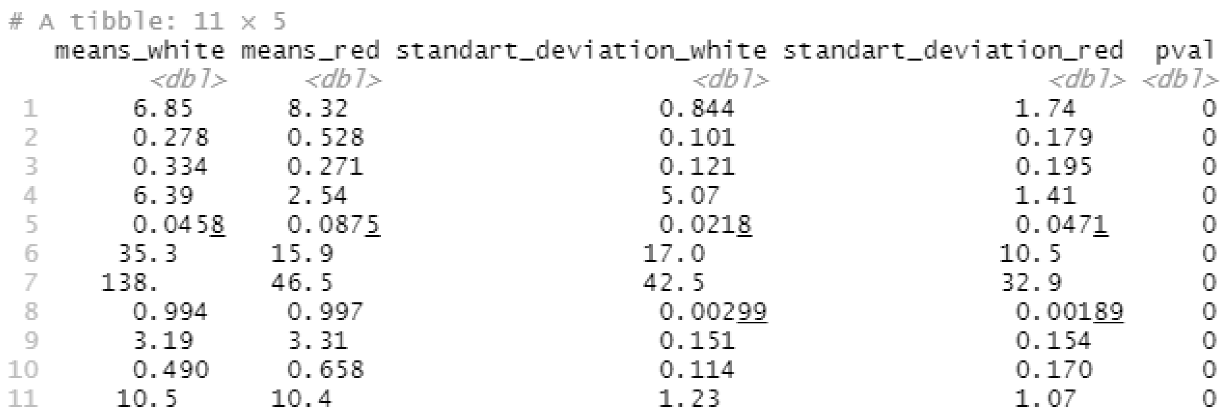
Let us start by looking at the data we have.

Consider 11 physicochemical characteristics as predictors. To analyze each predictor, calculate each group's mean and standard deviation.

R code:



```
winedata = read.csv("D:/BIRMINGHAM/STUDIES/SEMESTER1/AppliedStatistics/Additional_Task/winedata.csv", header=T)

#Divide the data in two groups
groups_of_wine <- split(winedata, winedata$wine)

#Data for red wine
red_group = groups_of_wine$Red
red_group <- red_group[-13]
red_group <- red_group[-12]
means_red = colMeans(red_group)
standart_deviation_red = apply(red_group, 2, sd)

#Data for red wine
white_group = groups_of_wine$White
white_group <- white_group[-13]
white_group <- white_group[-12]
means_white = colMeans(white_group)
standart_deviation_white = apply(white_group, 2, sd)

df <- data.frame(means_white, means_red, standart_deviation_white, standart_deviation_red) %>%
  rowwise%>%
  mutate(pval = t.test(white_group,red_group)$p.value) %>%
  ungroup()
df
```

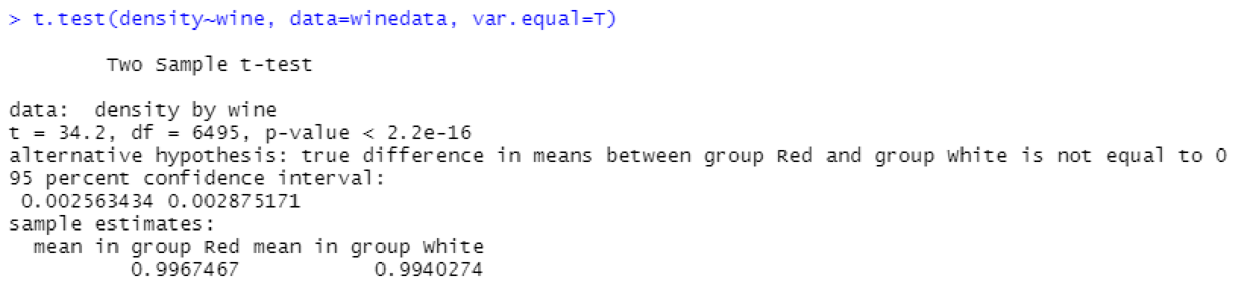


```
# A tibble: 11 × 5
   means_white means_red standart_deviation_white standart_deviation_red  pval
         <dbl>     <dbl>                    <dbl>                  <dbl> <dbl>
 1      6.85      8.32                    0.844                  1.74      0
 2      0.278     0.528                   0.101                  0.179     0
 3      0.334     0.271                   0.121                  0.195     0
 4      6.39      2.54                    5.07                   1.41      0
 5      0.0458    0.0875                  0.0218                 0.0471    0
 6     35.3      15.9                    17.0                   10.5       0
 7    138.       46.5                    42.5                   32.9       0
 8      0.994     0.997                   0.00299                0.00189   0
 9      3.19      3.31                    0.151                  0.154     0
10      0.490     0.658                   0.114                  0.170     0
11     10.5      10.4                     1.23                   1.07      0
```

Figure 1. Table with mean, standard deviation for two groups of wines, and p-value of the means of the predictors

Figure 1 shows a table where all predictors are represented by order from 1 to 11, as in the initial data frame. The first two columns show the values of the mean of two groups, the next two are the standard deviation, and the last one introduces the p-value. This table gives us only the first tree's significant value. The values of the p-value for each predictor are equal to zero in figure 1. That means that all values are a lot less than 0.01. And the differences between means are significant enough to reject the zero hypotheses, i.e., the means are not equal to each other.

Let us check it. Consider the mean for the predictor "density."



```
> t.test(density~wine, data=winedata, var.equal=T)

	Two Sample t-test

data:  density by wine
t = 34.2, df = 6495, p-value < 2.2e-16
alternative hypothesis: true difference in means between group Red and group White is not equal to 0
95 percent confidence interval:
 0.002563434 0.002875171
sample estimates:
  mean in group Red mean in group White
          0.9967467           0.9940274
```

The p-value is very small. So, we consent to the alternative hypothesis: the true difference in means between the two groups is not 0.

Training and test data set

Make preparation for the fitting classification model. We have to create our test and training test using our initial data.

Divide the initial data randomly with seed 102 for the opportunity to reproduce the code. R code:

```
#Divide data randomly
set.seed(102)
r = rnorm(1)
test = sample(nrow(winedata), nrow(winedata)*r)
winedata.train = winedata[-test,]
winedata.test = winedata[test,]

nrow(winedata.test) #the number of red and white wine samples in my test set
winedata.test_groups <- split(winedata.test, winedata.test$wine)
nrow(winedata.test_groups$Red)
nrow(winedata.test_groups$White)

nrow(winedata.train) #the number of red and white wine samples in my training set
winedata.train_groups <- split(winedata.train, winedata.train$wine)
nrow(winedata.train_groups$Red)
nrow(winedata.train_groups$White)
```

```
> nrow(winedata.test) #the number of red and white wine samples in my test set
[1] 1172
> winedata.test_groups <- split(winedata.test, winedata.test$wine)
> nrow(winedata.test_groups$Red)
[1] 288
> nrow(winedata.test_groups$White)
[1] 884
> nrow(winedata.train) #the number of red and white wine samples in my training set
[1] 5325
> winedata.train_groups <- split(winedata.train, winedata.train$wine)
> nrow(winedata.train_groups$Red)
[1] 1311
> nrow(winedata.train_groups$White)
[1] 4014
```

According to the output of the R code above, we have:

The number of samples in the test set is 1172, where 288 are red wine and 884 are white wine.

The number of samples in the training set is 5325, where 1311 are red wine, and 4014 are white wine.

Classification methods

Use the train data set to build the model and test the data set to check if it works properly. We will rely on the errors we get from fitting the models to the test data set.

The first classification method that we will consider is the logistic regression method.

Logistic regression

Write R code to build the model:

```
#Logistic regression
#Fit a logistic regression model to predict Direction
glm.fit = glm(as.factor(wine)~fixed.acidity+volatile.acidity+citric.acid+residual.sugar+
                chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
                density+pH+sulphates+alcohol, data=winedata.train, family = binomial)
summary(glm.fit)
```

```
> summary(glm.fit)

Call:
glm(formula = as.factor(wine) ~ fixed.acidity + volatile.acidity +
    citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + density + pH + sulphates + alcohol,
    family = binomial, data = winedata.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.5800   0.0009   0.0194   0.0591   6.1147

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           1.700e+03  1.917e+02   8.866  < 2e-16 ***
fixed.acidity         2.578e-01  2.475e-01   1.042   0.2974
volatile.acidity     -6.480e+00  1.123e+00  -5.768 8.01e-09 ***
citric.acid           1.732e+00  1.266e+00   1.367   0.1715
residual.sugar        9.121e-01  1.058e-01   8.620  < 2e-16 ***
chlorides            -2.597e+01  4.690e+00  -5.536 3.09e-08 ***
free.sulfur.dioxide  -6.380e-02  1.349e-02  -4.730 2.25e-06 ***
total.sulfur.dioxide  5.029e-02  5.103e-03   9.855  < 2e-16 ***
density              -1.693e+03  1.956e+02  -8.658  < 2e-16 ***
pH                    2.469e-01  1.517e+00   0.163   0.8707
sulphates            -3.205e+00  1.275e+00  -2.514   0.0119 *
alcohol              -1.763e+00  2.865e-01  -6.155 7.50e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5943.96  on 5324  degrees of freedom
Residual deviance:  375.17  on 5313  degrees of freedom
AIC: 399.17

Number of Fisher Scoring iterations: 9
```

The model shows us that according to the z-value of the predictors, the contribution of density, fixed.acidity and citric. acid is not significant.

R code to check how well the model works:

```
class_pred = predict(glm.fit, winedata.test, type = 'response')

class_pred = ifelse(class_pred > 0.5, "White", "Red")

table(class_pred, winedata.test$wine)

error1 = mean(class_pred != winedata.test$wine)
error1
```

```
> table(class_pred, winedata.test$wine)

class_pred Red White
     Red   287     6
     White   1   878
> error1 = mean(class_pred != winedata.test$wine)
> error1
[1] 0.005972696
```

The error is tiny, so the prediction is accurate, and the model is good. So the method works well.

Linear discriminant analysis

R code to build the model:

```
#LDA
lda.fit=lda(as.factor(wine)~fixed.acidity+volatile.acidity+citric.acid+residual.sugar+
                 chlorides+free.sulfur.dioxide+total.sulfur.dioxide+
                 density+pH+sulphates+alcohol, data=winedata.train)

lda.fit
```

```
> lda.fit
Call:
lda(as.factor(wine) ~ fixed.acidity + volatile.acidity + citric.acid +
    residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol, data = winedata.train)

Prior probabilities of groups:
      Red     White
0.2461972 0.7538028

Group means:
      fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides free.sulfur.dioxide total.sulfur.dioxide   density
Red        8.331960        0.5261670   0.2743555       2.535660 0.08676430            15.93059             46.55454 0.9967516
White      6.849626        0.2782337   0.3340533       6.337369 0.04548057            35.31751            138.05282 0.9939890
            pH sulphates  alcohol
Red   3.309840 0.6581159 10.42635
White 3.187409 0.4887444 10.52399

Coefficients of linear discriminants:
                               LD1
fixed.acidity           0.28958955
volatile.acidity       -3.07115416
citric.acid             0.83904481
residual.sugar          0.34266036
chlorides              -5.57639870
free.sulfur.dioxide    -0.01880831
total.sulfur.dioxide    0.01972106
density              -875.76036638
pH                      0.93422002
sulphates              -0.89015591
alcohol                -0.78833570
```

LDA gives us the group means and computes, for each individual, the prior probability of belonging to the different groups. The prior probability for the red group is 0.2462, and for the white – 0.7538.

We also got the model with the high coefficient of density predictor.

R code to check how well the model works:

```
lda.pred=predict(lda.fit, winedata.test)


lda.class =lda.pred$class
error3 = mean(lda.class != winedata.test$wine)
error3
```

```
> error3
[1] 0.004266212
```

The error is tiny, so the prediction is accurate, and the model is good. So, the method works well.

Classification trees

R code to build the model:

```
#Classification trees
#Build a model
model = rpart(wine~., data = winedata.train, method = 'class')
rpart.plot(model)
```

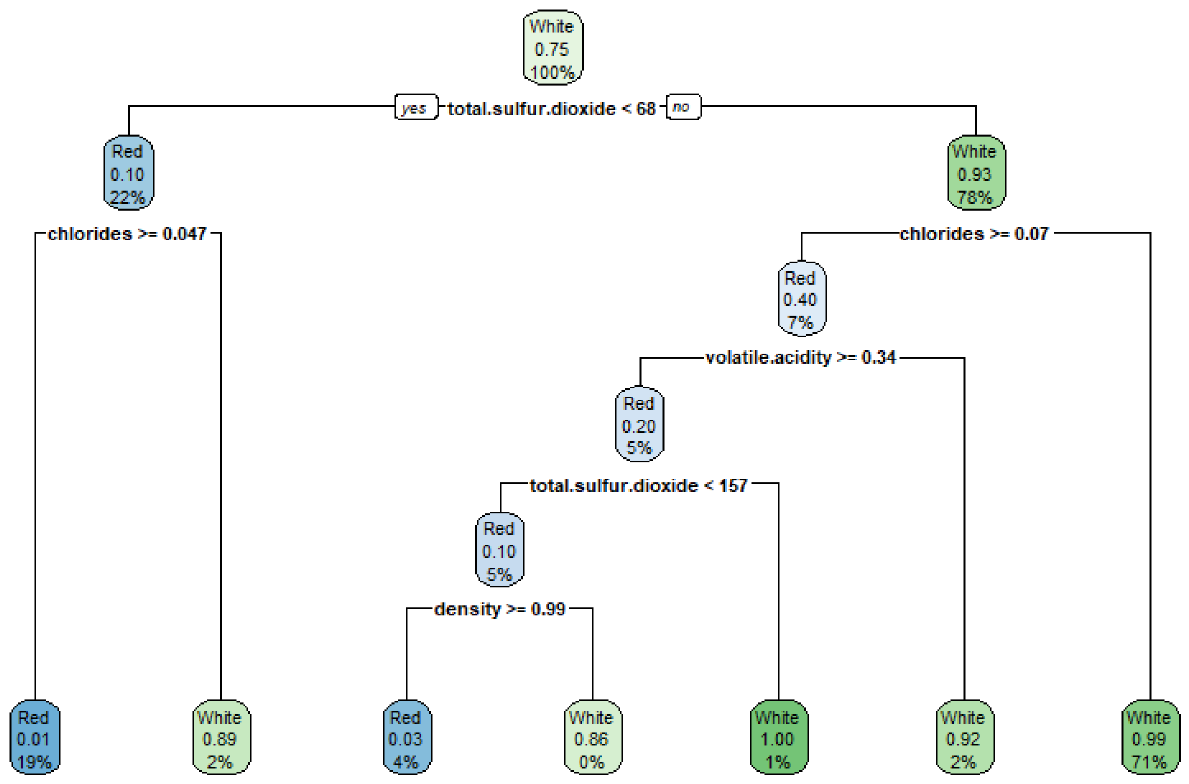


Figure 2. Classification tree model

Figure 2 illustrates the classification tree model. It has 6 nodes and consists of 4 predictors: total.sulfur.dioxide, chlorides, volatile.acidity, and density.

R code to check how well the model works:

```
#Predict
predict_tree = predict(model, winedata.test, type = "class")

table(winedata.test$wine, predict_tree)

error5 = mean(winedata.test$wine != predict_tree)
error5

> error5
[1] 0.01450512
```

The error is tiny, so the prediction is still accurate. However, the error is less accurate than LDA and Logistic regression errors.

Random forest

R code to build the model:

```
#Random forest
rf_model = randomForest(as.factor(wine)~., data = winedata.train)

summary(rf_model)
print(rf_model)
```

```
> summary(rf_model)
                Length Class  Mode
call                3  -none- call
type                1  -none- character
predicted        5325  factor numeric
err.rate         1500  -none- numeric
confusion           6  -none- numeric
votes           10650  matrix numeric
oob.times        5325  -none- numeric
classes             2  -none- character
importance         12  -none- numeric
importanceSD        0  -none- NULL
localImportance     0  -none- NULL
proximity           0  -none- NULL
ntree               1  -none- numeric
mtry                1  -none- numeric
forest             14  -none- list
y                5325  factor numeric
test                0  -none- NULL
inbag               0  -none- NULL
terms               3  terms  call
> print(rf_model)

Call:
 randomForest(formula = as.factor(wine) ~ ., data = winedata.train)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 3

        OOB estimate of  error rate: 0.49%
Confusion matrix:
       Red White class.error
Red   1291    20 0.015255530
White    6  4008 0.001494768
```

The method used 500 trees. The number of variables tried at each split is 3.

R code to check how well the model works:

```
pred_test = predict(rf_model, winedata.test)
error8 = mean(winedata.test$wine != pred_test)
error8
```

```
> error8
[1] 0.003412969
```

The error is tiny, so the prediction is accurate. And more accurate than the Classification tree error.

Gradient Boosting

R code to build the model:

```
#Gradient Boosting
model_gbm = gbm(wine~., data = winedata.train, distribution = "multinomial")

summary(model_gbm)
```

```
> summary(model_gbm)
                                      var      rel.inf
chlorides                       chlorides 47.352502306
total.sulfur.dioxide total.sulfur.dioxide 41.182212426
volatile.acidity         volatile.acidity  7.751514603
sulphates                       sulphates  1.717203442
pH                                     pH  0.622878410
fixed.acidity               fixed.acidity  0.573565590
density                           density  0.374645430
residual.sugar             residual.sugar  0.226878117
quality                           quality  0.102005171
alcohol                           alcohol  0.088896926
citric.acid                   citric.acid  0.007697578
free.sulfur.dioxide   free.sulfur.dioxide  0.000000000
```

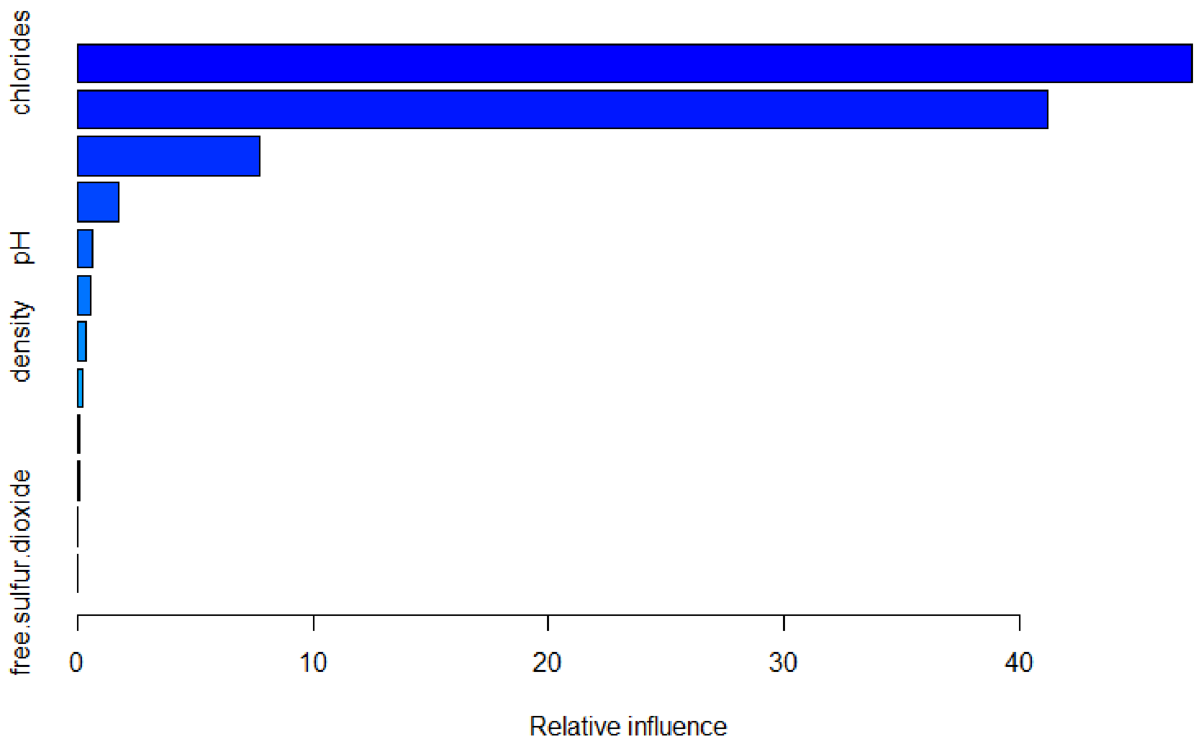


Figure 3. Relative influences of the predictors

In this model, the main influence on the model has two predictors: total.sulfur.dioxide, chlorides.

Tune parameters by choosing the number of trees, splits, and shrinkage parameters.

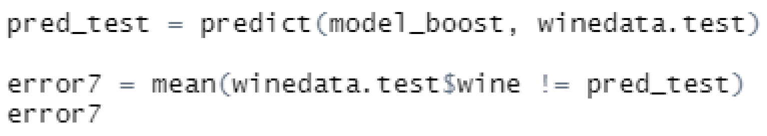
```
#Boosting with tuning
#cross-validation for tuning parameters
#choose 5 blocks for cross-validation
train_control = trainControl(method = "cv", number = 5)

#The tunning grid for choosing number of trees, splits and shrinkage parameter
gbmGrid <- expand.grid(max_depth = c(1, 2, 3),
                       nrounds = (1:10)*50,
                       eta = c(.1, .4),
                       gamma = 0,
                       colsample_bytree = 0.6,
                       min_child_weight = 1,
                       subsample = 1)

model_boost = train(wine~., data = winedata.train, method = "xgbTree", trControl = train_control, tuneGrid = gbmGrid)

summary(model_boost)
print(model_boost)
```

```
> summary(model_boost)
              Length Class              Mode
handle            1  xgb.Booster.handle externalptr
raw           92419  -none-             raw
niter             1  -none-             numeric
call              5  -none-             call
params            8  -none-             list
callbacks         1  -none-             list
feature_names    12  -none-             character
nfeatures         1  -none-             numeric
xNames           12  -none-             character
problemType       1  -none-             character
tuneValue         7  data.frame         list
obsLevels         2  -none-             character
param             0  -none-             list
> print(model_boost)
eXtreme Gradient Boosting

5325 samples
  12 predictor
   2 classes: 'Red', 'White'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 4260, 4259, 4260, 4260, 4261
Resampling results across tuning parameters:
```

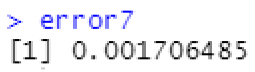
```
  eta  max_depth  nrounds  Accuracy   Kappa
  0.1  1           50      0.9810329  0.9481698
  0.1  1          100      0.9892958  0.9709958
  0.1  1          150      0.9902349  0.9735803
  0.1  1          200      0.9919256  0.9781962
  0.1  1          250      0.9926768  0.9802223
  0.1  1          300      0.9936157  0.9827488
  0.1  1          350      0.9938033  0.9832490
  0.1  1          400      0.9941791  0.9842628
  0.1  1          450      0.9943669  0.9847750
  0.1  1          500      0.9947423  0.9858003
  0.1  2           50      0.9883563  0.9684151
  0.1  2          100      0.9928640  0.9806934
  0.1  2          150      0.9936152  0.9827419
  0.1  2          200      0.9941786  0.9842612
  0.1  2          250      0.9947421  0.9857977
  0.1  2          300      0.9953055  0.9873287
  0.1  2          350      0.9954931  0.9878458
  0.1  2          400      0.9953053  0.9873442
  0.1  2          450      0.9954931  0.9878563
  0.1  2          500      0.9954931  0.9878563
  0.1  3           50      0.9915491  0.9770966
  0.1  3          100      0.9938030  0.9832462
  0.1  3          150      0.9941786  0.9842638
  0.1  3          200      0.9953053  0.9873233
  0.1  3          250      0.9953053  0.9873324
  0.1  3          300      0.9954931  0.9878432
  0.1  3          350      0.9956807  0.9883525
  0.1  3          400      0.9956807  0.9883525
  0.1  3          450      0.9956807  0.9883525
  0.1  3          500      0.9956807  0.9883525
  0.4  1           50      0.9921125  0.9786775
  0.4  1          100      0.9945542  0.9852983
  0.4  1          150      0.9947416  0.9858271
  0.4  1          200      0.9943658  0.9848210
  0.4  1          250      0.9943658  0.9848210
  0.4  1          300      0.9941779  0.9843218
  0.4  1          350      0.9941779  0.9843218
  0.4  1          400      0.9939903  0.9838125
  0.4  1          450      0.9943657  0.9848170
  0.4  1          500      0.9941779  0.9843153
  0.4  2           50      0.9947425  0.9857954
  0.4  2          100      0.9960569  0.9893693
  0.4  2          150      0.9958692  0.9888704
  0.4  2          200      0.9954935  0.9878604
  0.4  2          250      0.9954935  0.9878604
  0.4  2          300      0.9954931  0.9878640
  0.4  2          350      0.9951177  0.9868544
  0.4  2          400      0.9951177  0.9868544
  0.4  2          450      0.9951177  0.9868544
  0.4  2          500      0.9951177  0.9868544
  0.4  3           50      0.9958685  0.9888451
  0.4  3          100      0.9956807  0.9883447
  0.4  3          150      0.9956807  0.9883447
  0.4  3          200      0.9956807  0.9883447
  0.4  3          250      0.9958687  0.9888557
 0.4  3          300      0.9958687  0.9888557
 0.4  3          350      0.9958687  0.9888557
 0.4  3          400      0.9958687  0.9888557
 0.4  3          450      0.9958687  0.9888557
 0.4  3          500      0.9958687  0.9888557

Tuning parameter 'gamma' was held constant at a value of 0
Tuning parameter 'colsample_bytree' was held constant at a value
 of 0.6
Tuning parameter 'min_child_weight' was held constant at a value of 1
Tuning parameter 'subsample' was held constant
 at a value of 1
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nrounds = 100, max_depth = 2, eta = 0.4, gamma = 0, colsample_bytree =
 0.6, min_child_weight = 1 and subsample = 1.
```

Using cross-validation, we chose tunning parameters:

- number of trees = 100
- number of splits = 2
- shrinkage parameter = 0.4



```r
pred_test = predict(model_boost, winedata.test)

error7 = mean(winedata.test$wine != pred_test)
error7
```



```
> error7
[1] 0.001706485
```

We got the least error among all that we got before.

Conclusion

All methods classify wines highly accurately using their physicochemical characteristics. The classification tree method produces the biggest error = 0.0145, so the method is less accurate than others. The least error has the gradient boosting method with tuning parameters.

According to Gradient Boosting and Classification tree methods, the variables total.sulfur.dioxide and chlorides are more important than the others.