

Wine Quality Prediction

Amine Agrane & Lydia Khelfane

10/02/2021

```
## Loading required package: lattice
## Loading required package: ggplot2
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

Project and Dataset Presentation

In this notebook we will use the data from the Kaggle Repository (<https://www.kaggle.com/rajyellow46/wine-quality>) by P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. The data include examples of red and white wines from Portugal-one of the world's leading wine-producing countries.

For each wine, a laboratory analysis measured characteristics such as acidity, sugar content, chlorides, sulfur, alcohol, pH, and density. The samples were then rated in a blind tasting by panels of no less than three judges on a quality scale ranging from zero (very bad) to 10 (excellent). In the case of judges disagreeing on the rating, the median value was used.

Objective of this notebook The objective that we want to achieve throw this notebook is to build a machine learning model of classification type, that will predict if a wine is considered as good or note. The model takes as input some wine characteristics (alcohol content, acidity, sugar proportion, etc), and gives as output a binary variable that describes the quality of the wine (“Good” or “Bad”). We'll use a decision tree as our classification model.

Dataset description The wine dataset is composed by a total of 14 different variables :

1- fixed acidity (tartaric acid - g/dm3) : most acids involved with wine or fixed or nonvolatile (do not evaporate readily)

2- volatile acidity (acetic acid - g/dm3): the amount of acetic acid in wine, which at too high of levels can lead to an unpleasant, vinegar taste

3- citric acid (g/dm3) : found in small quantities, citric acid can add ‘freshness’ and flavor to wines

4- residual sugar (g/dm3) : the amount of sugar remaining after fermentation stops, it's rare to find wines with less than 1 gram/liter and wines with greater than 45 grams/liter are considered sweet

5- chlorides (sodium chloride - g/dm3) : the amount of salt in the wine

6- free sulfur dioxide (mg/dm3) : the free form of SO₂ exists in equilibrium between molecular SO₂ (as a dissolved gas) and bisulfite ion; it prevents microbial growth and the oxidation of wine

7- total sulfur dioxide (mg/dm3) : amount of free and bound forms of S₀₂; in low concentrations, SO₂ is mostly undetectable in wine, but at free SO₂ concentrations over 50 ppm, SO₂ becomes evident in the nose and taste of wine

8- density (g/cm3) : the density of water is close to that of water depending on the percent alcohol and sugar content

9- pH : describes how acidic or basic a wine is on a scale from 0 (very acidic) to 14 (very basic); most wines are between 3-4 on the pH scale

10- sulphates (potassium sulphate - g/dm3) : a wine additive which can contribute to sulfur dioxide gas (S₀₂) levels, wich acts as an antimicrobial and antioxidant

11- alcohol (% by volume) : the percent alcohol content of the wine Output variable (based on sensory data):

12- quality: score between 0 and 10 that describes the quality of the wine.

13- color: color of the wine. There are two type wine, red wine and white wine.

Exploratory Data Analysis on the wine dataset

Load the wine dataset We start by reading the data which is stored in the csv file `winequality.csv`. The file is loaded using the `read.csv` command, along with the `as.data.frame` command which store our data inside a structure of dataframe type.

```
# Load the data
df_wine <- as.data.frame( read.csv(file = './data/winequality.csv', sep=',', stringsAsFactors=F))
head(df_wine)
```

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4           0.70           0.00           1.9           0.076
## 2           7.8           0.88           0.00           2.6           0.098
## 3           7.8           0.76           0.04           2.3           0.092
## 4          11.2           0.28           0.56           1.9           0.075
## 5           7.4           0.70           0.00           1.9           0.076
## 6           7.4           0.66           0.00           1.8           0.075
##      free.sulfur.dioxide total.sulfur.dioxide density    pH sulphates alcohol
## 1                   11                   34 0.9978 3.51      0.56      9.4
## 2                   25                   67 0.9968 3.20      0.68      9.8
## 3                   15                   54 0.9970 3.26      0.65      9.8
## 4                   17                   60 0.9980 3.16      0.58      9.8
## 5                   11                   34 0.9978 3.51      0.56      9.4
## 6                   13                   40 0.9978 3.51      0.56      9.4
##      quality color
## 1          5    red
## 2          5    red
## 3          5    red
## 4          6    red
## 5          5    red
## 6          5    red
```

```
tail(df_wine)
```

```
##      fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 6492           6.5           0.23           0.38           1.3           0.032
```

```
## 6493      6.2      0.21      0.29      1.6      0.039
## 6494      6.6      0.32      0.36      8.0      0.047
## 6495      6.5      0.24      0.19      1.2      0.041
## 6496      5.5      0.29      0.30      1.1      0.022
## 6497      6.0      0.21      0.38      0.8      0.020
##      free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 6492      29      112 0.99298 3.29      0.54      9.7
## 6493      24      92 0.99114 3.27      0.50     11.2
## 6494      57     168 0.99490 3.15      0.46      9.6
## 6495      30     111 0.99254 2.99      0.46      9.4
## 6496      20     110 0.98869 3.34      0.38     12.8
## 6497      22      98 0.98941 3.26      0.32     11.8
##      quality color
## 6492      5 white
## 6493      6 white
## 6494      5 white
## 6495      6 white
## 6496      7 white
## 6497      6 white
```

Basic analysis on the dataset

Now we're gonna achieve some basic analysis on our wine dataset to get a better understanding of its contents, size and structure.

Dimension of our wine Dataset :

```
cat('The number of rows inside the dataset is : ', dim(df_wine)[1])
```

```
## The number of rows inside the dataset is : 6497
```

```
cat('The number of columns/features inside the dataset is : ', dim(df_wine)[2])
```

```
## The number of columns/features inside the dataset is : 13
```

Structure and distribution of the variables :

We use the `str` and `summary` functions to display some basic statistics about our dataset. The `str` function shows the nature (type) of each variable (feature) of our dataset, and some values that the feature can take. The `summary` function in other hand give us some statistics metrics (min, max, median, etc) for each feature of the dataset.

```
str(df_wine)
```

```
## 'data.frame': 6497 obs. of 13 variables:
## $ fixed.acidity : num 7.4 7.8 7.8 11.2 7.4 7.4 7.9 7.3 7.8 7.5 ...
## $ volatile.acidity : num 0.7 0.88 0.76 0.28 0.7 0.66 0.6 0.65 0.58 0.5 ...
## $ citric.acid : num 0 0 0.04 0.56 0 0 0.06 0 0.02 0.36 ...
## $ residual.sugar : num 1.9 2.6 2.3 1.9 1.9 1.8 1.6 1.2 2 6.1 ...
## $ chlorides : num 0.076 0.098 0.092 0.075 0.076 0.075 0.069 0.065 0.073 0.071 ...
## $ free.sulfur.dioxide : num 11 25 15 17 11 13 15 15 9 17 ...
## $ total.sulfur.dioxide: num 34 67 54 60 34 40 59 21 18 102 ...
## $ density : num 0.998 0.997 0.997 0.998 0.998 ...
## $ pH : num 3.51 3.2 3.26 3.16 3.51 3.51 3.3 3.39 3.36 3.35 ...
## $ sulphates : num 0.56 0.68 0.65 0.58 0.56 0.56 0.46 0.47 0.57 0.8 ...
## $ alcohol : num 9.4 9.8 9.8 9.8 9.4 9.4 9.4 10 9.5 10.5 ...
## $ quality : int 5 5 5 6 5 5 5 7 7 5 ...
## $ color : chr "red" "red" "red" "red" ...
```

```
summary(df_wine)
```

```
## fixed.acidity    volatile.acidity    citric.acid    residual.sugar
## Min.   : 3.800    Min.   :0.0800    Min.   :0.0000    Min.   : 0.600
## 1st Qu.: 6.400    1st Qu.:0.2300    1st Qu.:0.2500    1st Qu.: 1.800
## Median : 7.000    Median :0.2900    Median :0.3100    Median : 3.000
## Mean   : 7.215    Mean   :0.3397    Mean   :0.3186    Mean   : 5.443
## 3rd Qu.: 7.700    3rd Qu.:0.4000    3rd Qu.:0.3900    3rd Qu.: 8.100
## Max.   :15.900    Max.   :1.5800    Max.   :1.6600    Max.   :65.800
## chlorides        free.sulfur.dioxide    total.sulfur.dioxide    density
## Min.   :0.00900    Min.   : 1.00    Min.   : 6.0    Min.   :0.9871
## 1st Qu.:0.03800    1st Qu.: 17.00    1st Qu.: 77.0    1st Qu.:0.9923
## Median :0.04700    Median : 29.00    Median :118.0    Median :0.9949
## Mean   :0.05603    Mean   : 30.53    Mean   :115.7    Mean   :0.9947
## 3rd Qu.:0.06500    3rd Qu.: 41.00    3rd Qu.:156.0    3rd Qu.:0.9970
## Max.   :0.61100    Max.   :289.00    Max.   :440.0    Max.   :1.0390
## pH              sulphates              alcohol              quality
## Min.   :2.720    Min.   :0.2200    Min.   : 8.00    Min.   :3.000
## 1st Qu.:3.110    1st Qu.:0.4300    1st Qu.: 9.50    1st Qu.:5.000
## Median :3.210    Median :0.5100    Median :10.30    Median :6.000
## Mean   :3.219    Mean   :0.5313    Mean   :10.49    Mean   :5.818
## 3rd Qu.:3.320    3rd Qu.:0.6000    3rd Qu.:11.30    3rd Qu.:6.000
## Max.   :4.010    Max.   :2.0000    Max.   :14.90    Max.   :9.000
## color
## Length:6497
## Class :character
## Mode :character
##
##
##
```

Correlation between the variables :

```
as.data.frame( cor(df_wine[c(-13, -14)]))
```

```
##              fixed.acidity    volatile.acidity    citric.acid    residual.sugar
## fixed.acidity      1.00000000      0.21900826    0.32443573    -0.11198128
## volatile.acidity    0.21900826      1.00000000   -0.37798132    -0.19601117
## citric.acid         0.32443573    -0.37798132    1.00000000     0.14245123
## residual.sugar     -0.11198128    -0.19601117    0.14245123     1.00000000
## chlorides          0.29819477     0.37712428    0.03899801    -0.12894050
## free.sulfur.dioxide -0.28273543    -0.35255731    0.13312581     0.40287064
## total.sulfur.dioxide -0.32905390    -0.41447619    0.19524198     0.49548159
## density            0.45890998     0.27129565    0.09615393     0.55251695
## pH                 -0.25270047     0.26145440   -0.32980819    -0.26731984
## sulphates          0.29956774     0.22598368    0.05619730    -0.18592741
## alcohol            -0.09545152    -0.03764039   -0.01049349    -0.35941477
## quality            -0.07674321    -0.26569948    0.08553172    -0.03698048
## chlorides free.sulfur.dioxide    total.sulfur.dioxide
## fixed.acidity      0.29819477     -0.28273543    -0.32905390
## volatile.acidity    0.37712428     -0.35255731    -0.41447619
## citric.acid         0.03899801     0.13312581     0.19524198
## residual.sugar     -0.12894050     0.40287064     0.49548159
## chlorides          1.00000000     -0.19504479    -0.27963045
## free.sulfur.dioxide -0.19504479     1.00000000     0.72093408
```

```
## total.sulfur.dioxide -0.27963045      0.72093408      1.00000000
## density              0.36261466      0.02571684      0.03239451
## pH                   0.04470798     -0.14585390     -0.23841310
## sulphates            0.39559331     -0.18845725     -0.27572682
## alcohol              -0.25691558     -0.17983843     -0.26573964
## quality              -0.20066550      0.05546306     -0.04138545
##              density      pH      sulphates      alcohol
## fixed.acidity      0.45890998 -0.25270047  0.299567744 -0.095451523
## volatile.acidity    0.27129565  0.26145440  0.225983680 -0.037640386
## citric.acid         0.09615393 -0.32980819  0.056197300 -0.010493492
## residual.sugar      0.55251695 -0.26731984 -0.185927405 -0.359414771
## chlorides           0.36261466  0.04470798  0.395593307 -0.256915580
## free.sulfur.dioxide 0.02571684 -0.14585390 -0.188457249 -0.179838435
## total.sulfur.dioxide 0.03239451 -0.23841310 -0.275726820 -0.265739639
## density             1.00000000  0.01168608  0.259478495 -0.686745422
## pH                  0.01168608  1.00000000  0.192123407  0.121248467
## sulphates           0.25947850  0.19212341  1.000000000 -0.003029195
## alcohol             -0.68674542  0.12124847 -0.003029195  1.000000000
## quality             -0.30585791  0.01950570  0.038485446  0.444318520
##              quality
## fixed.acidity      -0.07674321
## volatile.acidity    -0.26569948
## citric.acid         0.08553172
## residual.sugar      -0.03698048
## chlorides           -0.20066550
## free.sulfur.dioxide 0.05546306
## total.sulfur.dioxide -0.04138545
## density            -0.30585791
## pH                  0.01950570
## sulphates           0.03848545
## alcohol             0.44431852
## quality             1.00000000
```

Number of NAN values for each variables :

```
cat('Number of NAN values for each variable :', colSums(is.na(df_wine)))
```

```
## Number of NAN values for each variable : 0 0 0 0 0 0 0 0 0 0 0 0 0
```

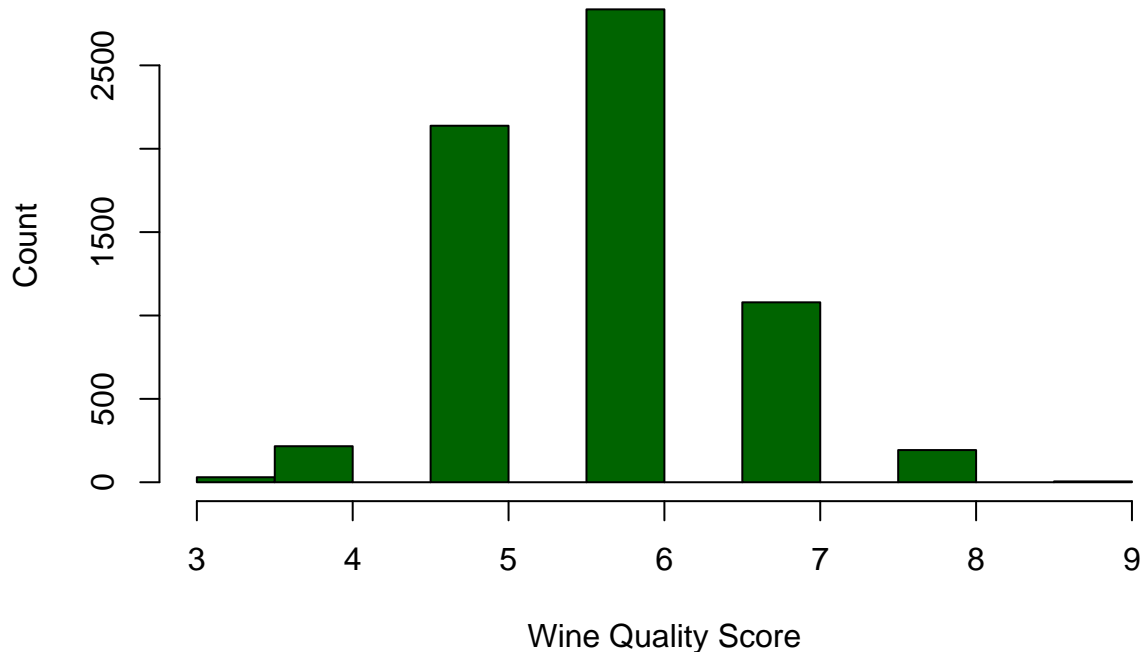
Preparing the data

We want to predict the quality of the wine (score between 1 and 10) by using the rest of the feature variables. Here, wine quality is the variable we want to predict (Y variable).

Inspect the wine quality score feature Let's take a look at the distribution of the wine quality score in our dataset. We see that Wine quality appear to follow a bell shaped distribution (Gaussian/Normal distribution). This implies most wines are of average quality and few are good or bad

```
# Simple Bar Plot
hist(df_wine$quality, main="Wine Quality Score Distribution",ylab="Count", xlab="Wine Quality Score", b
```

Wine Quality Score Distribution



Now let's check the frequency of each wine quality score. From the above table, we see that an approximate 45% of the wine quality scores have a 6 score value.

```
table(df_wine$quality)
```

```
##
##    3    4    5    6    7    8    9
##  30  216 2138 2836 1079  193   5
```

Binning the Wine Quality Score Variable We want to predict the quality of the wine based on the rest of the variables present in our dataset. We see from the above table that the wine quality score variable takes 7 different values (3, 4, 5, 6, 7, 8, 9) inside our dataset. If we want to build a machine learning model to predict this variable, we'll use a classification model that will achieve a multiclass classification, in our case we have 7 distinct classes (7 possible values).

The question we want to answer through this notebook is whatever a wine is good or bad ? In our case, the output variable (Y) that we want to get is a binary variable True => "Good" and False => "Bad".

We transform our wine quality score variable to a binary variable by achieving a binning on its values, i.e. we fix a specific threshold and associate each score to a unique class, the class "Bad" for the lower quality wines and the class "Good" for the best wines.

Threshold : - Good : scores from 7 to 9. - Bad : scores from 1 to 6.

```
cat('Threshold distribution : ', table(ifelse(df_wine$quality<=6, 'Bad', ifelse(df_wine$quality>=7, "Good", "Bad"))))
```

```
## Threshold distribution : 5220 1277
```

```
# We drop the quality column and replace it with the new qualityClass variable.
qualityClass <- as.factor(ifelse(df_wine$quality<=6, 'Bad', ifelse(df_wine$quality>=7, 'Good', '')))
df_wine <- data.frame(subset(df_wine, select = -quality), qualityClass)
```

Build a Decision Tree Model to predict wine quality class.

Splitting the data into training and testing sets We split our data into two different sets :

- Training set for the model fitting (learning the patterns)
- Testing set for estimating the model's accuracy

```
# we set the seed to get reproducible results
set.seed(10)

# get the training dataset indexes
dataset_size <- dim(df_wine)[1]
train_set_size <- round(0.8*dataset_size)
train_index <- sample(dataset_size, train_set_size, replace=FALSE)

# split into train and test sets
training_data <- df_wine[train_index, ]
test_data <- df_wine[-train_index,]
```

Instantiate and train the Decision Tree Model We will begin by training a classification tree model. Although almost any implementation of decision trees can be used to perform classification tree modeling, the `rpart` (recursive partitioning) package offers the most faithful implementation of classification trees as they were described by the CART team. As the classic R implementation of CART, the `rpart` package is also well-documented and supported with functions for visualizing and evaluating the `rpart` models.

Using the R formula interface, we can specify `qualityClass` as the outcome variable and use the dot notation to allow all the other columns in the `training_data` data frame to be used as predictors.

```
rpart_model <- rpart(qualityClass~., data=training_data, method="class")

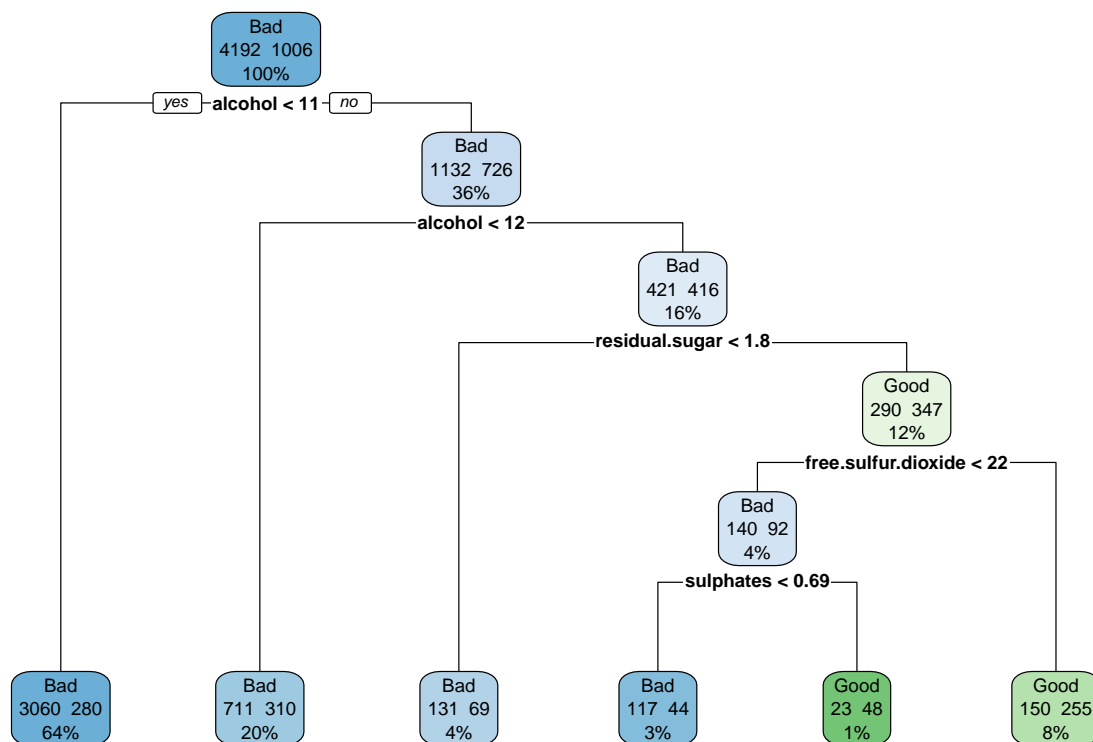
# choosing the best complexity parameter "cp" to prune the tree
cp.optim <- rpart_model$cptable[which.min(rpart_model$cptable[, "xerror"]), "CP"]

# tree pruning using the best complexity parameter. For more in
tree <- prune(rpart_model, cp=cp.optim)
```

Visualization of the decision tree model The tree can be understood using only the preceding output, it is often more readable using visualization. After installing the package using the `install.packages("rpart.plot")` command, the `rpart.plot()` function produces a tree diagram from any `rpart` model object. The following commands plot the classification tree we built earlier which produces a tree diagram as follows:

For each node in the tree, the number of examples reaching the decision point is listed. For instance, all 5198 examples (100% of the examples) begin at the root node, of which 64% have `alcohol < 11`. Because `alcohol` was used first in the tree, it is the single most important predictor of wine quality class.

```
# Visualise the decision tree model graphically
rpart.plot(rpart_model, extra=101)
```



Model Evaluation

The evaluation of machine learning algorithms consists in comparing the predicted value with the actual outcome. The confusion matrix displays the predicted and observed values in a table and provides additional statistics summary. We'll use the `confusionMatrix` function to display the following metric about our classification model :

- **The confusion matrix** : The confusion matrix is a simple table with the cross tabulation of the predicted values with the actual observed values. All values are in absolute numbers of observations and predictions, so for example the True Positive is the number of predicted values that are exactly the same as the actual values. In our case, a True Positive is a Bad wine (in terms of quality) that was correctly classified as Bad.
- **Sensitivity** : It's the proportion of positive values when they are actually positive, i.e. the ability to predict positive values.
- **Specificity** : It's the probability of a predicted negative value conditioned to a negative outcome.

```
# store the predicted values by our decision tree model
predict_rpart <- predict(rpart_model, test_data[, -13], type="class")
confusionMatrix(predict_rpart, test_data[, 13])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##           Bad  987  202
```



```
##      Good  41   69
##
##              Accuracy : 0.8129
##              95% CI : (0.7906, 0.8338)
##      No Information Rate : 0.7914
##      P-Value [Acc > NIR] : 0.02901
##
##              Kappa : 0.2749
##
##      McNemar's Test P-Value : < 2e-16
##
##              Sensitivity : 0.9601
##              Specificity : 0.2546
##      Pos Pred Value : 0.8301
##      Neg Pred Value : 0.6273
##              Prevalence : 0.7914
##      Detection Rate : 0.7598
##      Detection Prevalence : 0.9153
##      Balanced Accuracy : 0.6074
##
##      'Positive' Class : Bad
##
```

Using the `confusionMatrix` function on our decision tree model, we get the following results :

- Accuracy : 0.8129
- Sensitivity : 0.9601
- Specificity : 0.2546

The overall accuracy score is pretty good, but we have a low Specificity score which mean that we globally fail to detect a True Negative (in our case a True Negative -> Correctly classified as Good wine). In order to improve classification score, we're gonna use in the next section a more complex classification model, which is a Random Forest classifier.

Improve model Accuracy by using a Random Forest

While decision trees are easy to interpret, they tend to be rather simplistic and are often outperformed by other algorithms. Random Forests are one way to improve the performance of decision trees. The algorithm starts by building out trees similar to the way a normal decision tree algorithm works. However, every time a split has to made, it uses only a small random subset of features to make the split instead of the full set of features.

From the below output, we instantiate and train a random forest model on our wine dataset. This model will generates 1000 decision trees with 3 variables randomly sampled (approximate of $\sqrt{\text{number of variables}}$ = 4) used for splitting.

```
rf_model <- randomForest(qualityClass~., data=training_data, ntree=1000, mtry=4)
rf_model

##
## Call:
## randomForest(formula = qualityClass ~ ., data = training_data,          ntree = 1000, mtry = 4)
##              Type of random forest: classification
##              Number of trees: 1000
##      No. of variables tried at each split: 4
##
```

```
##           OOB estimate of  error rate: 11.2%
## Confusion matrix:
##           Bad Good class.error
## Bad   4042  150  0.03578244
## Good   432  574  0.42942346
```

Evaluate the Random Forest model Same as for the decision tree model, we use the `confusionMatrix` function to evaluate the performances of our Random Forest Classifier. This step is to check how well the model performs with the test data. We see that we now have a better classification model, as the evaluation metrics have all been improved. From the below confusion matrix, we get the following results for our Random Forest model :

- Accuracy : 0.8876
- Sensitivity : 0.9708
- Specificity : 0.5720

998 wines are classified as low-quality wines and 155 are high-quality wines. The accuracy of the model is 88.75%. That means 11.75% (116+30) of wines are predicted wrong by the model. If a customer picks a high-quality wine there is only 2.3% (30/(998+116+30+155)) chance that wine is low quality wine. In the Random Forest model, the prediction of high-quality wines is more accurate compared to a single decision tree model.

The sensitivity (total high-quality wines predicted by high-quality wines) is the percentage of predicted low-quality wines 96% which is good whereas the specificity gives the percentage of predicted high-quality wines i.e., only 57%. This could be explained because the model may be biased in classifying more low-quality wines due to an unbalanced wine quality dataset.

```
rf_predict <- predict(rf_model, test_data[, -13])
confusionMatrix(rf_predict, test_data[, 13])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Bad Good
##           Bad  995  114
##           Good   33  157
##
##           Accuracy : 0.8868
##           95% CI : (0.8683, 0.9036)
##           No Information Rate : 0.7914
##           P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.6149
##
## Mcnemar's Test P-Value : 4.159e-11
##
##           Sensitivity : 0.9679
##           Specificity : 0.5793
##           Pos Pred Value : 0.8972
##           Neg Pred Value : 0.8263
##           Prevalence : 0.7914
##           Detection Rate : 0.7660
##           Detection Prevalence : 0.8537
##           Balanced Accuracy : 0.7736
##
##           'Positive' Class : Bad
```

##

Conclusion

The objective of this notebook was to build a classification model able to predict the quality class of a given wine (Good or Bad) based on its characteristics. After a first step of exploratory data analysis that helped us to better understand the data we're working on, we performed some transformations on our data to facilitate the construction of classification model. After training a first decision tree model, we we're not totaly satisfied with the model performances, so we decided to try another more complex classification model which is a Random Forest Classifier.

Prediction of high-quality wines with Random Forest model is good with less overfitting problems and with high accuracy (from step4); because of the aggregation of decision trees. To pick a desired quality wine, it is always important to check few parameters (from step5) and it's value ranges to know whether we picked the right one or not. Overall, the correct prediction of wines is necessary because there would be chances that the wine we have picked would be not our interest.