Red Wine

PedroCadilha

04/01/2021

Preface

This report is part of the capstone project of HarvardX's Data Science Professional Certificate1 program.

The R Markdown code used to generate this report and the R script are available on GitHub.

1. Introduction

In this project we are going to analyze the Red Wine dataset which is related to the portuguese wine "Vinho verde tinto". In this dataset several wines were given a quality classification. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

In the following sections we are going to do an overview of the dataset to become familiar with it, setting up the path for the modeling section.

In a first approach we are going to use logistic regression, knn nearest neighbors and random forest models, based on this results we will notice that due to the nature of our data (limited number of good wines) we will have to use techniques to achieve better results. We are going to use methods to subsample it, namely, upsampling, downsampling and hybrid methods. This new datasets are going to be modeled using only the Random Forest model.

We will use the F1 scores to evaluate our results.

By the end of this project we will be able to predict the quality of a wine based on it's characteristics and also point out which of them are the most important to have a good wine.

2. Data preparation and exploration

The dataset was created by Paulo Cortez, University of Minho, Guimarães, Portugal, in 2009, and downloaded from the UCI Machine Learning Repository.

There is some data wrangling we had to do with the delimiters and decimal marks. We changed also the column names.

Our data now looks like this:

```
##   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   0.998 3.51      0.56     9.4
## 2                  25                   67   0.997 3.20      0.68     9.8
## 3                  15                   54   0.997 3.26      0.65     9.8
## 4                  17                   60   0.998 3.16      0.58     9.8
## 5                  11                   34   0.998 3.51      0.56     9.4
## 6                  13                   40   0.998 3.51      0.56     9.4
##   quality
## 1       5
## 2       5
## 3       5
## 4       6
## 5       5
## 6       5
```

We have a dataset with 1599 rows (wine observations) and 12 columns (11 wine features plus the column quality). The features are:

Fixed acidity - Most acids involved with wine or fixed or nonvolatile (do not evaporate readily).

Volatile acidity - The amount of acetic acid in wine, which at too high levels can lead to an unpleasant, vinegar taste.

Citric acid - Found in small quantities, citric acid can add freshness and flavor to wines.

Residual sugar - Quantity of sugar left in the wine after the fermentation process. To summarize, a dry wine contains from 0 to 4 grams of sugar per liter, a semi-dry wine from 4 to 12 grams per liter, a semi-sweet wine from 8 to 45 grams per liter and a sweet wine contains more than 45 grams per liter.

Chlorides - The amount of salt in the wine.

Free sulfur dioxide - The portion of SO_2 that is free in the wine

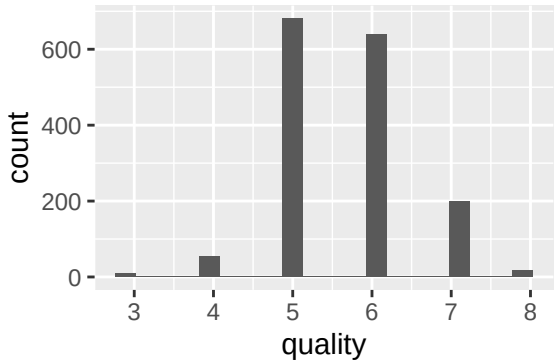
Total sulfur dioxide - Amount of free and bound forms of $S0_2$ in the wine. The sulfur dioxide prevents the wine from reacting with oxygen which can cause browning and off-odors (oxidation), and it inhibits the growth of bacteria and undesirable wild yeasts in the grape juice and wine

Density - Is close to water density, depending on the percent alcohol and sugar content.

ph - Typically, the pH level of a wine ranges from 3 to 4. Red wines with higher acidity are more likely to be a bright ruby color, as the lower pH gives them a red hue.

Sulphates - A wine additive which can contribute to sulfur dioxide gas ($S0_2$) levels, which acts as an antimicrobial and antioxidant. **Alcohol** - Alcohol content range from as little as 5.5% alcohol by volume to as much as around 20%.

The quality of the wine is distributed like this:



Clearly there is a majority of average wines (quality 5 or 6) and a small amount of poor (3 and 4) or good wines (7 or 8).

I remember that we want to identify good wines, so we are going to apply a cutoff on wine quality. The wines with quality equal or above 7 are going to be classified as "good" (category "1") and the wines with quality below 7 are going to be classified as "no good" (category "0").

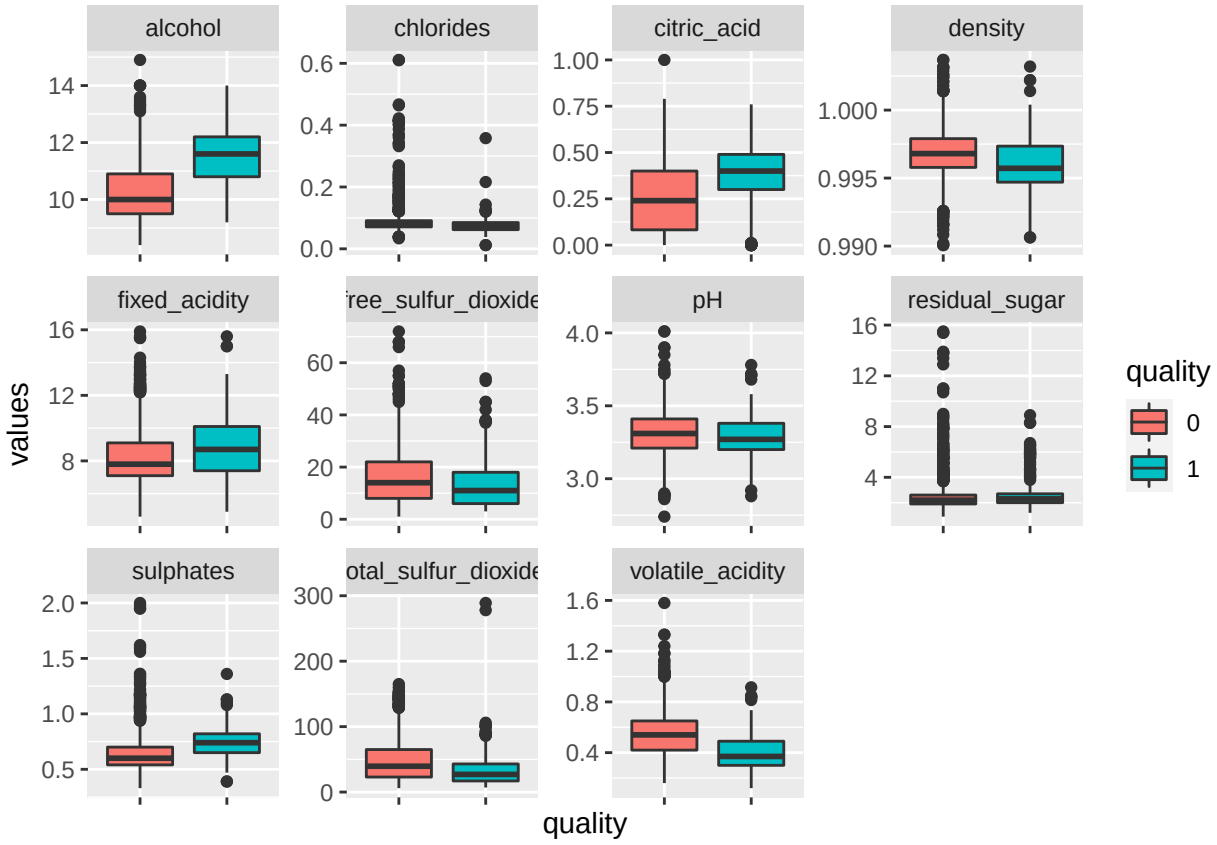
After this cutoff we have now:

```
##
##    0    1
## 1382  217
```

Giving a proportion between wines of:

```
##
##     0     1
## 0.864 0.136
```

We are going to do boxplots of all the features vs. quality to have an overview of the data:/par



Looking at this boxplots we get an idea about the distribution of each feature in the wine. Notice that all the features are present in both wines, with the alcohol and citric acid (freshness and flavor) having a bigger presence in the good wines and the volatile acidity in the bad ones (vinegar taste).

3. Modeling

The data was partitioned in a training and test set.

After this, we have a training set like this:

```
##
##    0    1
## 1105  173
```

and a test set with:

```
##
##   0   1
## 277  44
```

It seem that our classes are unbalanced! There are much less good wines in our data. On a first approach we are going to apply logistic regression, knn-nearest neighbors and random forest models to train our data.

Before we run in to this methods we will clarify some concepts which are going to be used to evaluate them. Overall accuracy can sometimes be a deceptive measure because of unbalanced classes.

In this cases the study of sensitivity, specificity and precision might help:

Sensitivity- True positive rate or recall. Also known as **Recall**. Is the proportion of actual positive overcomes correctly identified as such.

Specificity- True negative rate. Is the proportion of actual negative outcomes that are correctly identified as such.

Precision - Positive predicted value. Is the proportion of correctly predicted positive observations to the total predicted positive observations. The confusion matrix tabulates each combination of prediction and actual value. We can create a confusion matrix in R using the **confusionMatrix()** function from the **caret** package.

So let's start modeling:

3.1 Logistic regression model

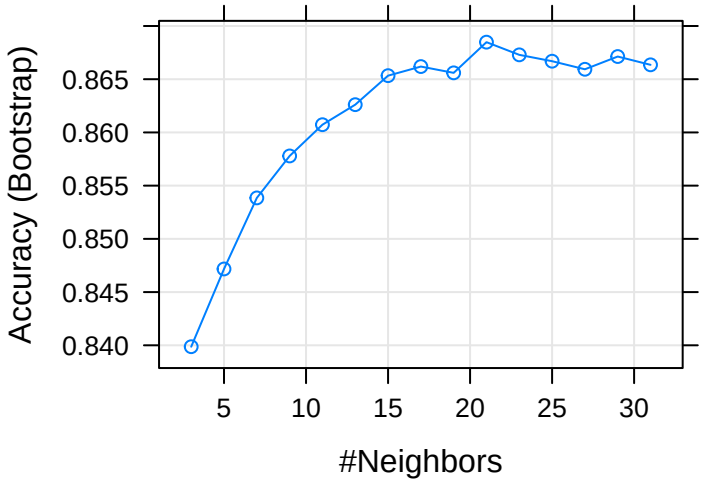
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 270  31
##          1   7  13
##
##                Accuracy : 0.882
##                  95% CI : (0.841, 0.915)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 0.187042
##
##                   Kappa : 0.351
##
##  Mcnemar's Test P-Value : 0.000191
##
##             Sensitivity : 0.2955
```

```
##             Specificity : 0.9747
##          Pos Pred Value : 0.6500
##          Neg Pred Value : 0.8970
##              Prevalence : 0.1371
##          Detection Rate : 0.0405
##    Detection Prevalence : 0.0623
##       Balanced Accuracy : 0.6351
##
##        'Positive' Class : 1
##
```

We have a high accuracy of 0.882 but if we look closer it's only a bit better than simply guess that all wines are weak due to the low prevalence, 0.137, in our data. The most noticeable is the low sensitivity of 0.296, which means that we predicted correctly only 13 good wines in a total of 44 in our test set.

3.2 K-Nearest Neighbours

Running the Knn model, we can plot the accuracy against k to identify the optimal k to use in our model:



From the graph we see that the optimal k is 21 and the confusion matrix:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 274  44
##          1   3   0
##
##                Accuracy : 0.854
##                  95% CI : (0.81, 0.89)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 0.719
##
##                   Kappa : -0.018
##
```

```
##  Mcnemar's Test P-Value : 5.39e-09
##
##             Sensitivity : 0.00000
##             Specificity : 0.98917
##          Pos Pred Value : 0.00000
##          Neg Pred Value : 0.86164
##              Prevalence : 0.13707
##          Detection Rate : 0.00000
##    Detection Prevalence : 0.00935
##       Balanced Accuracy : 0.49458
##
##        'Positive' Class : 1
##
```

This model is clearly oriented to the majority class being unable to predict correctly any good wine! The sensitivity is 0!

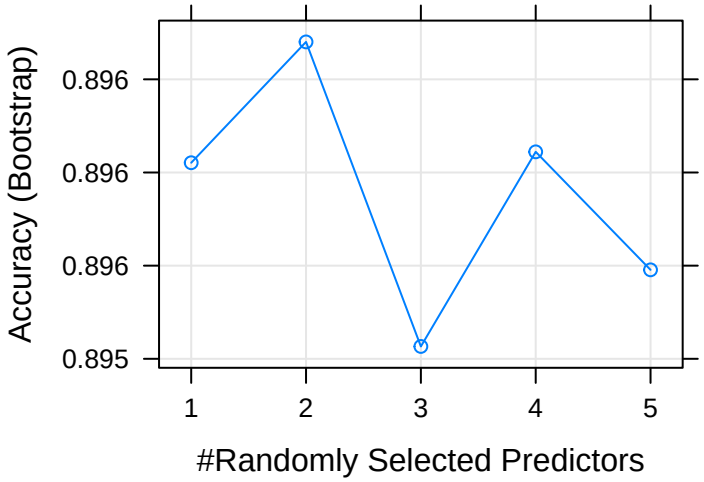
3.3 Random forest model

When applying the random forest model there are two parameters which are the most likely to have the biggest effect on our final accuracy:

mtry - Number of variables randomly sampled as candidates at each split.

ntree - Number of trees to grow

Only the *mtry* parameter is available in caret for tuning, generally, in classification models, the value is about the square root of the number of predictors, since we have 11 predictors, we are going to choose a tuning interval between 1 and 5.



We see from the plot that the best tune for mtry is 2 and the confusion matrix is:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
```

```
##          0 276  20
##          1   1  24
##
##                Accuracy : 0.935
##                  95% CI : (0.902, 0.959)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 3.51e-05
##
##                   Kappa : 0.662
##
##  Mcnemar's Test P-Value : 8.57e-05
##
##             Sensitivity : 0.5455
##             Specificity : 0.9964
##          Pos Pred Value : 0.9600
##          Neg Pred Value : 0.9324
##              Prevalence : 0.1371
##          Detection Rate : 0.0748
##    Detection Prevalence : 0.0779
##       Balanced Accuracy : 0.7709
##
##        'Positive' Class : 1
##
```

Our accuracy of 0.935 is quite good but the sensitivity of 0.5455, despite of it's increase compared to the previous models is not good enough.

In the logistic regression model, from the 44 good wines in the test set, we were able to predict correctly 13, 0 wines with Knn and finally 24 with the Random Forest.

The results from this first approach are not satisfactory, due the unbalanced dataset with a low prevalence of good wines.

4. Creating balanced dataset

The simple technique to reduce the negative impact of this problem is by subsampling the data. We are going to use the following methods:

Upsampling - This method increases the size of the minority class by sampling with replacement so that the classes will have the same size.

Downsampling - Decreases the size of the majority class to be the same or closer to the minority class size by just taking out a random sample.

Hybrid methods - Downsample the majority class and create new artificial points in the minority class. We are going to use *ROSE* (random oversampling examples) and *SMOTE* (Synthetic minority oversampling technique). It's necessary to install, respectively, the libraries ROSE and DMwR to use this techniques.

Remember that our train set currently as the following distribution of wines:

```
##
##    0    1
## 1105  173
```

Let's apply this techniques to create the new training sets.

4.1 Upsampling

The upsampling technique produces a train set like this:

```
set.seed(2020,sample.kind = "Rounding")
train_up<-upSample(x=train[,-ncol(train)],y=train$quality)
table(train_up$Class)
```

```
##
##    0    1
## 1105 1105
```

4.2 Downsampling

The downsampling produces a train set like this:

```
set.seed(2020,sample.kind = "Rounding")
train_down<-downSample(x=train[,-ncol(train)],y=train$quality)
table(train_down$Class)
```

```
##
##   0   1
## 173 173
```

We point out that both **upSample** and **downSample** functions changed the name of the column for our wine quality from **quality** to **Class**.

4.3 ROSE

The ROSE technique creates a more balanced dataset decreasing the majority class and increasing the minority class:

```
set.seed(2020,sample.kind = "Rounding")
train_rose<-ROSE(quality ~ ., data=train)$data
table(train_rose$quality)
```

```
##
##   0   1
## 650 628
```

4.4 SMOTE

Similarly SMOTE technique creates a more balanced dataset decreasing the majority class and increasing the minority class:

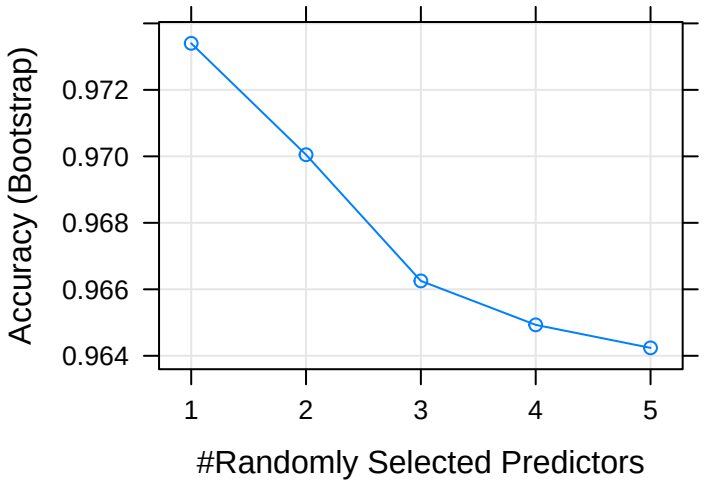
```
set.seed(2020,sample.kind = "Rounding")
train_smote<-SMOTE(quality~., data=train)
table(train_smote$quality)
```

```
##
##   0   1
## 692 519
```

5. Modeling with balanced data

5.1 Random Forest with upsampling

We are ready to start modeling with our new balanced datasets. We are going to use only random forest model since it showed better perfomance with the unbalanced dataset. We are going to use, like before, a tuning interval between 1 and 5 for mtry.



Our best **mtry** is 1.

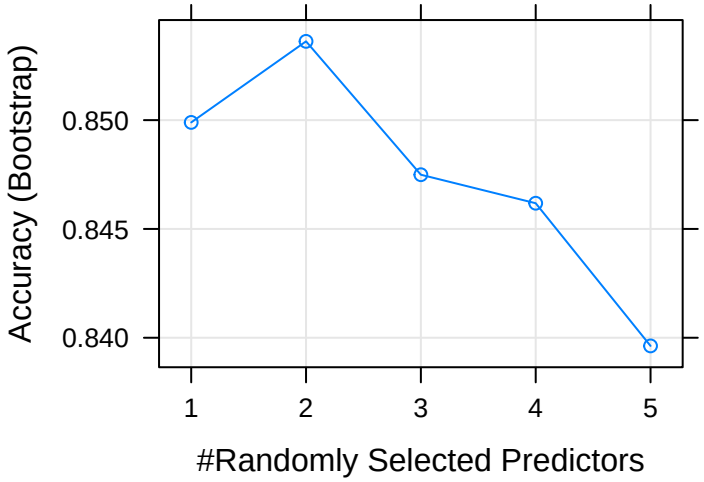
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 271  18
##          1   6  26
##
##                Accuracy : 0.925
##                  95% CI : (0.891, 0.952)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 0.000348
##
##                   Kappa : 0.643
##
##  Mcnemar's Test P-Value : 0.024745
##
##             Sensitivity : 0.5909
##             Specificity : 0.9783
##          Pos Pred Value : 0.8125
##          Neg Pred Value : 0.9377
##              Prevalence : 0.1371
##          Detection Rate : 0.0810
##    Detection Prevalence : 0.0997
##       Balanced Accuracy : 0.7846
##
##        'Positive' Class : 1
```

##

The confusion matrix shows a slight improvement in sensitivity (from 0.54 to 0.59). We were able to predict now 26 good wines correctly, against 24 before in the original data.

5.2 Random Forest with downsampling

we are going to repeat the same steps with the downsample.



Our best **mtry** is 2.

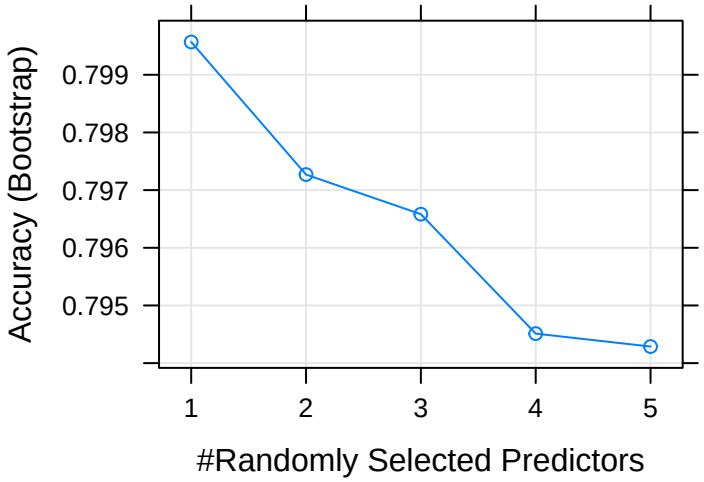
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 212   6
##          1  65  38
##
##                Accuracy : 0.779
##                  95% CI : (0.729, 0.823)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.402
##
##  Mcnemar's Test P-Value : 5.85e-12
##
##             Sensitivity : 0.864
##             Specificity : 0.765
##          Pos Pred Value : 0.369
##          Neg Pred Value : 0.972
##              Prevalence : 0.137
##          Detection Rate : 0.118
##    Detection Prevalence : 0.321
##       Balanced Accuracy : 0.814
```

```
##
##        'Positive' Class : 1
##
```

The confusion matrix shows a big improvement in sensitivity (from 0.54 to 0.86). We were able to predict now 38 good wines correctly, against 24 before in the original data. With this method we lost accuracy, since we lost predictive power on the not good wines.

5.3 Random Forest with ROSE

Repeating the same steps with the ROSE sample:



Our best **mtry** is 1.

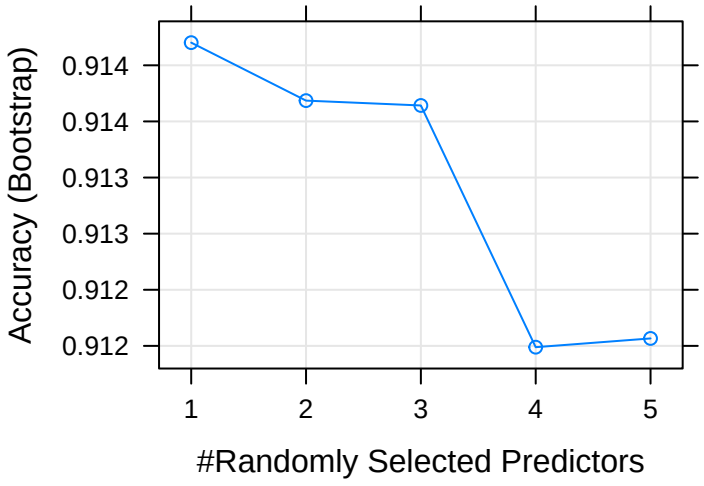
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 224  12
##          1  53  32
##
##                Accuracy : 0.798
##                  95% CI : (0.749, 0.84)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.385
##
##  Mcnemar's Test P-Value : 7e-07
##
##             Sensitivity : 0.7273
##             Specificity : 0.8087
##          Pos Pred Value : 0.3765
##          Neg Pred Value : 0.9492
##              Prevalence : 0.1371
```

```
##          Detection Rate : 0.0997
##    Detection Prevalence : 0.2648
##       Balanced Accuracy : 0.7680
##
##        'Positive' Class : 1
##
```

The confusion matrix shows a big improvement in sensitivity (from 0.54 to 0.809). We were able to predict now 32 good wines correctly, against 24 before in the original data. With this method we lost accuracy, since we lost predictive power on the not good wines, only 224 out of 277 predicted currectly.

5.4 Random Forest with SMOTE

Repeating the same steps with the SMOTE sample:



Our best **mtry** is 1.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 237  11
##          1  40  33
##
##                Accuracy : 0.841
##                  95% CI : (0.796, 0.879)
##     No Information Rate : 0.863
##     P-Value [Acc > NIR] : 0.887
##
##                   Kappa : 0.474
##
##  Mcnemar's Test P-Value : 8.83e-05
##
##             Sensitivity : 0.750
##             Specificity : 0.856
```

```
##          Pos Pred Value : 0.452
##          Neg Pred Value : 0.956
##              Prevalence : 0.137
##          Detection Rate : 0.103
##    Detection Prevalence : 0.227
##       Balanced Accuracy : 0.803
##
##        'Positive' Class : 1
##
```

The confusion matrix shows a big improvement in sensitivity (from 0.54 to 0.75). We were able to predict now 33 good wines correctly, against 24 before in the original data. With this method we lost accuracy, since we lost predictive power on the not good wines, only 237 out of 277 predicted currectly.

6. Results

To evaluate our results we would use the **ROC curve**, *Receiver Operating Characteristic*, since the ROC curve plots **sensitivity** versus **1-specificity**. However in cases, like ours, where the prevalence matters we decided to evaluate our models with the F1 score.

Sometimes it is more useful to have a one number summary, than studying both specificity and sensitivity separately.

The F1-score is the harmonic average of precision and recall. When we want to balance the weight between them we need to choose β.

For a β equal to 1 they have equal weight, when we want to add more weight to recall it should be bigger to 1, and smaller for a big weight to precision.

In our dataset the goal is not only to classify a wine as good when it's really good but also we **don't want to classify a bad wine as a good one**, so we are going to give more weight to **precision** than to **recall**.

When we serve a wine in a restaurant as being good, it's much more important that it's really good (not a bad one served as good), than when we serve one not so good that in fact is better than expected. Therefore we are going to give a value of 0.25 toβ.

The values obtained are resumed in this table:

model	f_meas_values
Random Forest	0.936
Random Forest Upsampling	0.940
Random Forest Downsampling	0.957
Random Forest ROSE	0.940
Random Forest SMOTE	0.949

From this table, the best model would be Random Forest with downsampling, with a value of F-meas of 0.957.

Notice that if we reduce the weight of precision (increase β) our best model measured by f1-meas will move towards the Random Forest original unbalanced dataset.

The results with β equal to 1:

model	f_meas_values
Random Forest	0.963
Random Forest Upsampling	0.958
Random Forest Downsampling	0.857
Random Forest ROSE	0.873
Random Forest SMOTE	0.903

Finally we can also check which features are more important to the wine classification from our chosen model, **Random Forest with downsampling**:

```
## rf variable importance
##
##                       Importance
## alcohol                    100.0
## sulphates                   84.4
## volatile_acidity            46.6
## chlorides                   35.6
## citric_acid                 35.2
## total_sulfur_dioxide        28.3
## fixed_acidity               26.8
## density                     21.9
## pH                          18.2
## residual_sugar               8.5
## free_sulfur_dioxide          0.0
```

The alcohol is the most important feature, followed by sulfates (antimicrobial and antioxidant) and volatile acidity (too high levels lead to vinegar taste).

7. Conclusion

Due to the unbalanced nature of our data, with a reduced number of good wines, we had to use techniques to balance it. The results improved only if we add more weight to the precision of our model. We used F-meas score with weighted β equal to 0.25 to evaluate the models. We used the final model to identify the most important features of red wine to access it's quality. Future results can be improved and also new models and approaches used if we had a bigger dataset with more good wines and also more features of the wine, like region, type of grapes, price or year of production.