UNIVERSITÀ DEGLI STUDI DI CATANIA
DIPARTIMENTO DI ECONOMIA E IMPRESA
CORSO DI LAUREA IN DATA SCIENCE FOR MANAGEMENT

REPORT ON "RED WINE QUALITY" DATASET

Campisi Daniela
1000011488

Casella Bruno
1000014143

Statistical Learning

ACADEMIC YEAR 2019 – 2020

"RED WINE QUALITY" DATASET

The dataset used in this report is the "Red Wine Quality" available at the following link:
https://archive.ics.uci.edu/ml/datasets/wine+quality
Two datasets are included, related to red and white variants of the Portuguese "Vinho Verde" wine, from north Portugal.
For more details, consult: Cortez et al., 2009
Due to privacy and logistic issues, only physicochemical (inputs) and sensory (output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.)
This dataset can be viewed as a classification or regression task. The classes are ordered and not balanced (e.g. there are much more normal wines than excellent or poor ones).
The target variable of the dataset is "quality". In the original data there are six values for the target variable, integer numbers ranging from 3 to 8. The target variable has been reclassified in four values: the values 3 and 4 have been put together in class "4" and the values 7 and 8 have been put together in class "7".
The dataset has been split for predictive analysis in training set (about 60% of the units of the original dataset), validation and test sets (about 20% of the units of the original dataset each).
The test set does not contain the values of the target variable.

Content of the dataset:

- 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:
    12. Quality

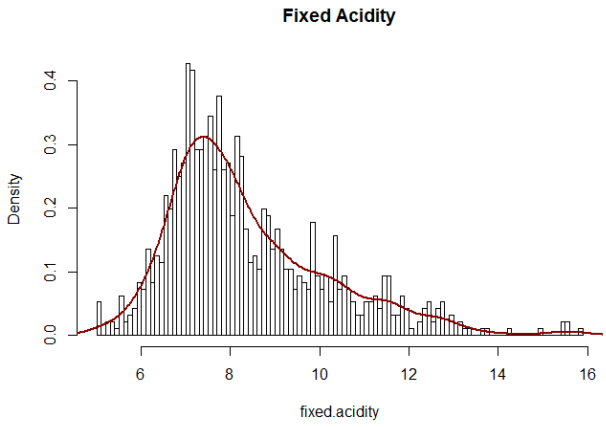
In this analysis we started with the classification of wines on the basis of their quality. In order to do this, we transformed the quality variable into a binary one, and built models such as logistic regression, support vector machines and neural vectors. At the end we compared all the models and chose the best one and then applied it to the validation data.

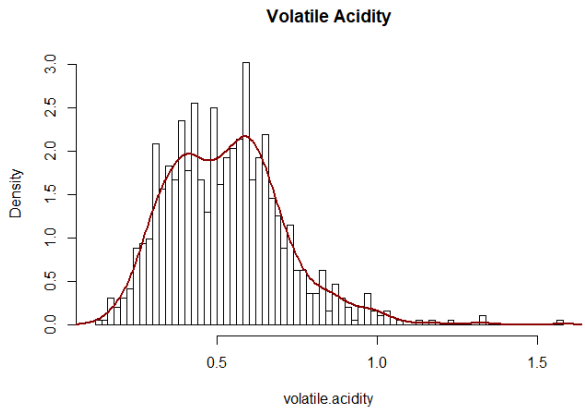
DESCRIPTIVE ANALYSIS

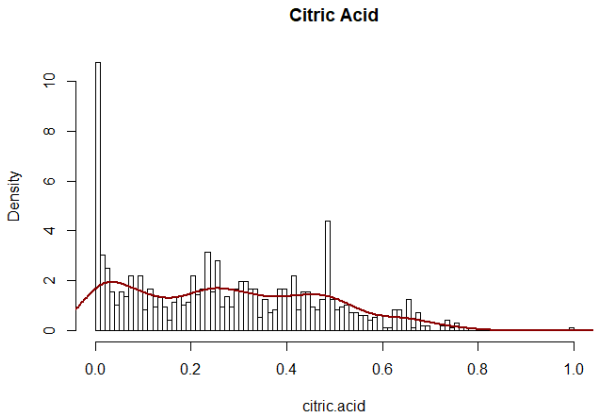
The training set is composed by 957 observations and 12 variables, and each of them is continuous. There are not NULL values. Let's have a look at the structure of the dataset:

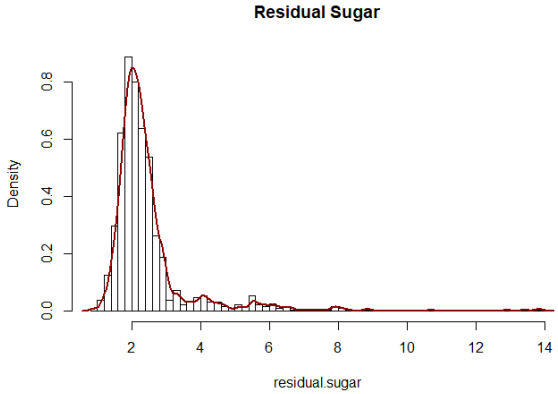
```
> str(data_train)
'data.frame':   957 obs. of  12 variables:
 $ fixed.acidity       : num  7.9 7.3 7 6.9 8.1 8.4 7.4 6.4 6.7 7.2 ...
 $ volatile.acidity    : num  0.885 0.98 0.975 0.39 0.87 0.67 0.59 0.53 0.76 0.58 ...
 $ citric.acid         : num  0.03 0.05 0.04 0.24 0 0.19 0.08 0.09 0.02 0.54 ...
 $ residual.sugar      : num  1.8 2.1 2 2.1 3.3 2.2 4.4 3.9 1.8 2.1 ...
 $ chlorides           : num  0.058 0.061 0.087 0.102 0.096 0.093 0.086 0.123 0.078 0.114 ...
 $ free.sulfur.dioxide : num  4 20 12 4 26 11 6 14 6 3 ...
 $ total.sulfur.dioxide: int  8 49 67 7 61 75 29 31 12 9 ...
 $ density             : num  0.997 0.997 0.996 0.995 1 ...
 $ pH                  : num  3.36 3.31 3.35 3.44 3.6 3.2 3.38 3.5 3.55 3.33 ...
 $ sulphates           : num  0.33 0.55 0.6 0.58 0.72 0.59 0.5 0.67 0.63 0.57 ...
 $ alcohol             : num  9.1 9.7 9.4 11.4 9.8 9.2 9 11 9.95 10.3 ...
 $ quality             : int  4 4 4 4 4 4 4 4 4 4 ...
> sum(!complete.cases(data_train)) #number of NA values = 0
[1] 0
> plot_missing(data_train)
```

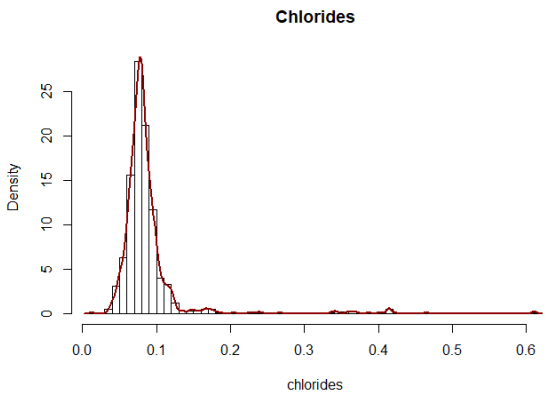
Let's have a look at the single variables:

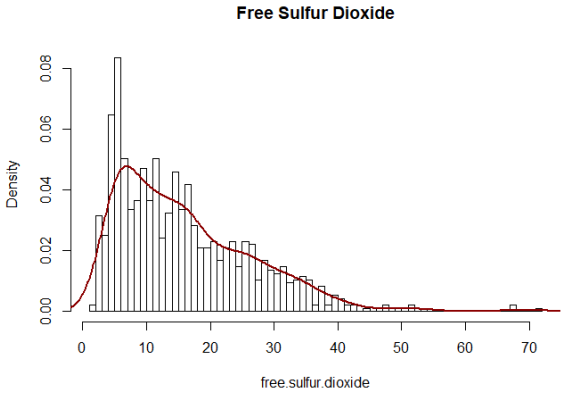


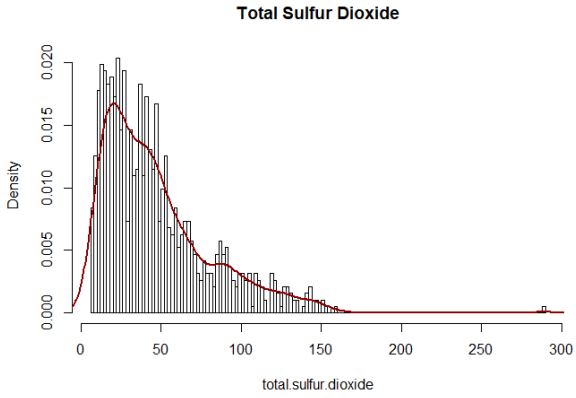


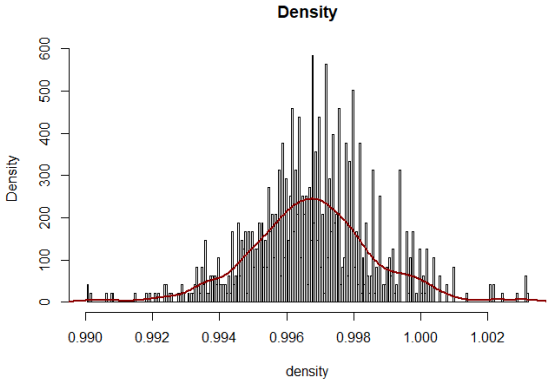


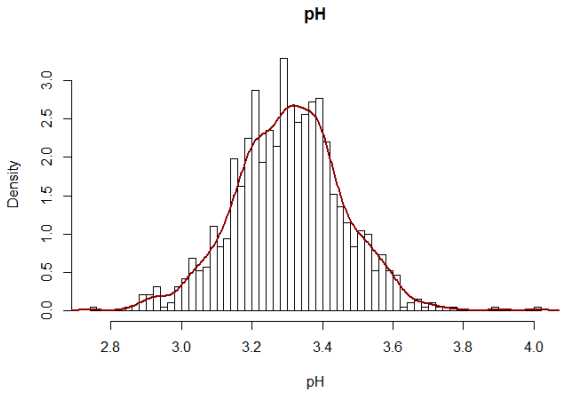


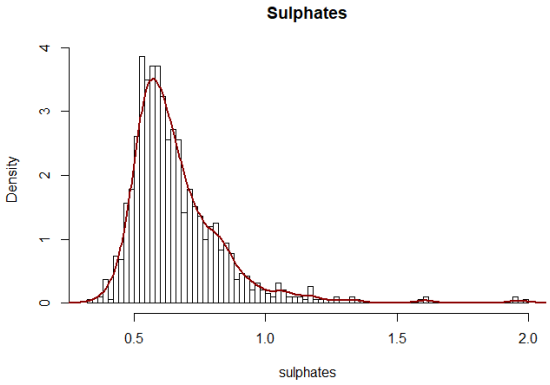

Chlorides
Density
0
5
10
15
20
25
0.0
0.1
0.2
0.3
0.4
0.5
0.6
chlorides

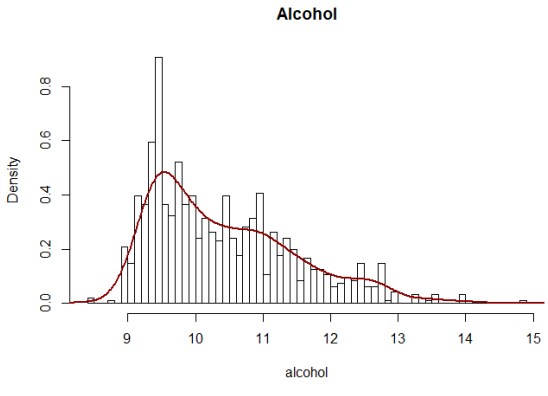

Free Sulfur Dioxide
Density
0.00
0.02
0.04
0.06
0.08
0
10
20
30
40
50
60
70
free.sulfur.dioxide

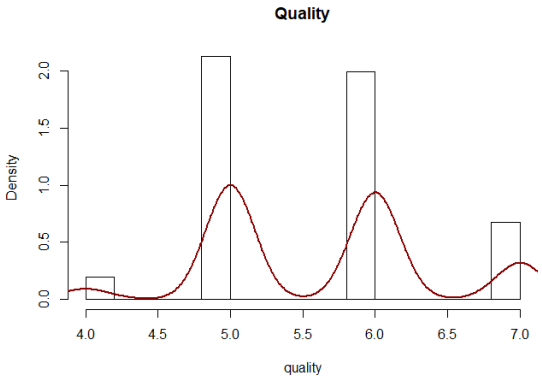

Total Sulfur Dioxide
Density
0.000
0.005
0.010
0.015
0.020
0
50
100
150
200
250
300
total.sulfur.dioxide

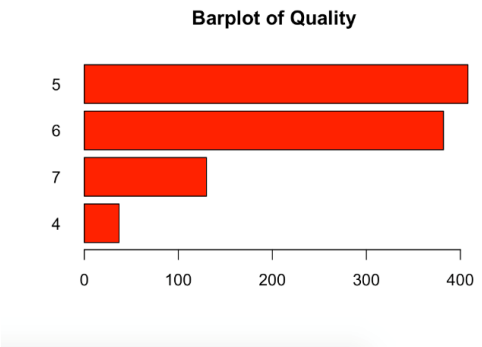

Density
Density
0
100
200
300
400
500
600
0.990
0.992
0.994
0.996
0.998
1.000
1.002
density


pH
Density
0.0
0.5
1.0
1.5
2.0
2.5
3.0
2.8
3.0
3.2
3.4
3.6
3.8
4.0
pH


Sulphates
Density
0
1
2
3
4
0.5
1.0
1.5
2.0
sulphates


Alcohol
Density
0.0
0.2
0.4
0.6
0.8
9
10
11
12
13
14
15
alcohol

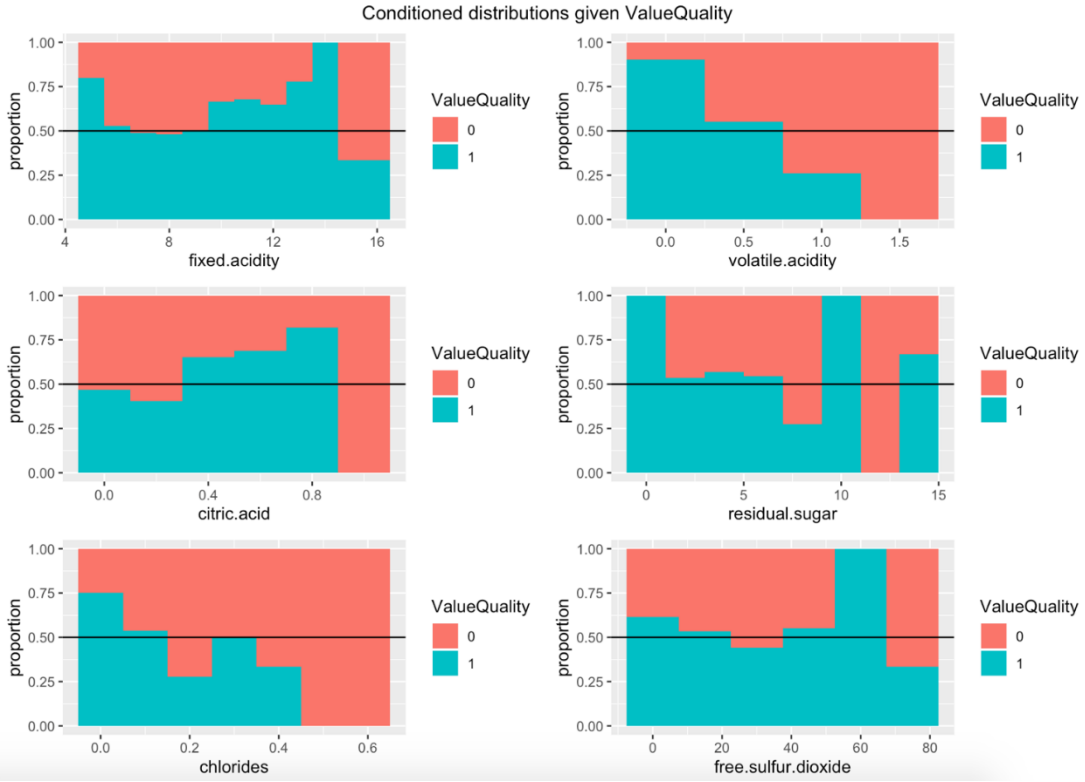

Quality
Density
0.0
0.5
1.0
1.5
2.0
4.0
4.5
5.0
5.5
6.0
6.5
7.0
quality

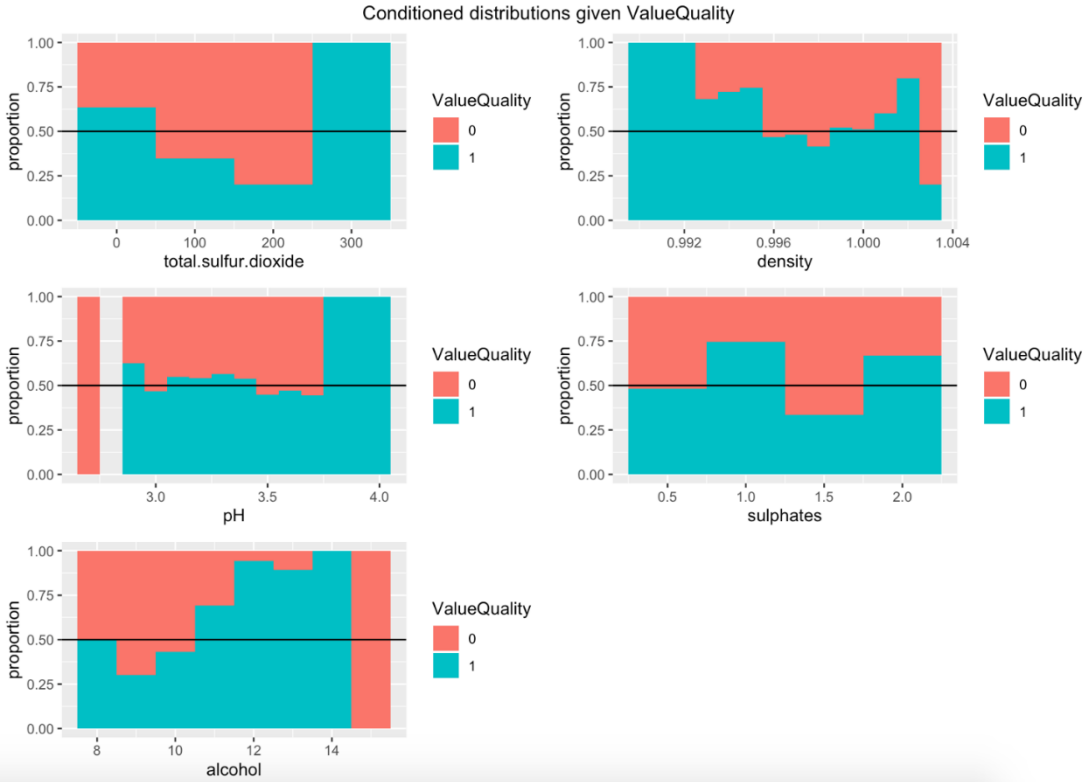


The distribution of **fixed acidity** is right skewed and leptokurtic, in fact it has a positive value for skewness, 1.11, and kurtosis greater than 3 (4.34). Moreover, it is a unimodal distribution, this means that this variable present only one class.
The distribution of **volatile acidity** is positively skewed and leptokurtic, with skewness=0.80 and kurtosis=4.74. Moreover, it is a bimodal distribution, this means that this variable could have two classes.
The distribution of **citric Acid** is right skewed and platykurtic, with skewness=0.33 and kurtosis less than 3 (2.21), and it is a unimodal distribution, this means that this variable could have one class.
The distribution of **residual sugar** is positively skewed (skewness=4.23) and leptokurtic (kurtosis=27.99), as for the distribution of **chlorides** (skewness=5.81 and kurtosis=44.06), **free sulfur dioxide** (skewness=1.16 and kurtosis=4.86), **total sulfur dioxide** (skewness=1.44 and kurtosis=6.28), **density** (skewness=0.04 and kurtosis=4.03), and **pH** (skewness=0.05 and kurtosis=3.67).
The distribution of **sulphates** is bimodal (two potential classes), right skewed (skewness=2.46) and leptokurtic (kurtosis=14.30).
The distribution of **alcohol** is unimodal (one potential class), right skewed (skewness=0.86) and leptokurtic (kurtosis=3.24).
The last variable, **quality**, unlike the others, which are numerical and continuous, is numerical and discrete, as it can assume values between 4 and 7. Furthermore, its distribution is unimodal, as mode assumes only one value, which is equal to 5, and is positively skewed (skewness=0.21) and platykurtic (kurtosis=2.44).

Bivariate Analysis with Quality:

Here we can see 11 charts that represents the conditioned distributions of each independent variable given the output variable quality:





We can see that the quality is excellent (values 6 or 7) in these cases:

- With Fixed Acidity lower than 6 and between 10 and 15.
- With Volatile Acidity lower than 0.75.
- With Citric Acid between 0.3 and 0.9.
- For all values of Residual Sugar but values between 7 and 9 and between 11 and 13.
- With Chlorides lower than 0.15.
- For all values of Free Sulfur Dioxide but values between 20 and 40 and greater than 70.
- For all values of Total Sulfur Dioxide but values between 50 and 250.
- With Density lower than 0.995 and between 1.001 and 1.003.
- With pH greater than 3.7.
- With Sulphates between 0.75 and 1.25 and greater than 1.75.
- With Alcohol greater than 10.5.

So, in the logistic regression we will use all the variables of the dataset as potential predictors.

MODELING TRAINING DATA

A) LOGISTIC REGRESSION

We will start modeling training data using multiple logistic regression, reason why we will use all the variables of the dataset as potential predictors.

The variable quality has already been transformed into a binary one before doing the bivariate analysis. It takes value 1 if it its value is 6 or 7 (excellent wine), or 0 if its value is 4 or 5 (bad wine).

We perform logistic regression using the **mixed selection** (stepwise regression) that is a combination of forward and backward selection. Firstly, the model has no variable; then we add the variable that provides the best fit, which is chosen by means of AIC, *Akaike Information Criterion* (e.g. if AIC decreases, we add that variable).

Then, we continue to add variable one by one.

Of course, the p-values for variables can become larger as new predictors are added to the model. Hence, if at any point the p-value of the variables in the model rises above a certain threshold, we remove that variable from the model.

We continue to perform these forward and backward steps until all variables in the model have a sufficiently low p-value, and all variables outside the model would have a large p-value if added to the model.

```
> logR = step(glm(ValueQuality~.-quality, data = data_train1, family = binomial),
+             direction = "both")
Start:  AIC=991.65
ValueQuality ~ (fixed.acidity + volatile.acidity + citric.acid +
    residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol + quality) - quality

                      Df Deviance     AIC
- citric.acid          1   967.66  989.66
- pH                   1   967.74  989.74
- fixed.acidity        1   969.48  991.48
<none>                     967.65  991.65
- residual.sugar       1   970.27  992.27
- free.sulfur.dioxide  1   972.26  994.26
- density              1   972.71  994.71
- chlorides            1   979.16 1001.16
- alcohol              1   984.57 1006.57
- sulphates            1   995.17 1017.17
- volatile.acidity     1   995.53 1017.53
- total.sulfur.dioxide 1  1003.04 1025.04
```

We can see that citric acid is the variable with the smallest AIC.

```
Step:  AIC=989.66
ValueQuality ~ fixed.acidity + volatile.acidity + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + pH + sulphates + alcohol

                      Df Deviance     AIC
- pH                   1   967.76  987.76
<none>                     967.66  989.66
- fixed.acidity        1   969.89  989.89
- residual.sugar       1   970.35  990.35
+ citric.acid          1   967.65  991.65
- free.sulfur.dioxide  1   972.37  992.37
- density              1   972.72  992.72
- chlorides            1   980.34 1000.34
- alcohol              1   985.07 1005.07
- sulphates            1   995.26 1015.26
- total.sulfur.dioxide 1  1006.35 1026.35
- volatile.acidity     1  1009.65 1029.65
```

Here we can see that if we remove citric acid from the model, its AIC increases and this means that it is not a good predictor.
Now the pH variable is the one with the smallest AIC.

```
Step:  AIC=987.76
ValueQuality ~ fixed.acidity + volatile.acidity + residual.sugar +
    chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + sulphates + alcohol

                      Df Deviance     AIC
<none>                     967.76  987.76
- residual.sugar       1   970.39  988.39
+ pH                   1   967.66  989.66
- fixed.acidity        1   971.68  989.68
+ citric.acid          1   967.74  989.74
- free.sulfur.dioxide  1   972.90  990.90
- density              1   974.08  992.08
- chlorides            1   982.02 1000.02
- alcohol              1   993.34 1011.34
- sulphates            1   995.60 1013.60
- total.sulfur.dioxide 1  1009.27 1027.27
- volatile.acidity     1  1009.68 1027.68
```

Here we can see that if we remove pH from the model, its AIC increases and this means that it is not a good predictor.
Hence, all the variables of the dataset, except Citric Acid and pH, are chosen as predictors in the final model.

Here we can see the details of our logistic regression model:

```
> summary(logR)

Call:
glm(formula = ValueQuality ~ fixed.acidity + volatile.acidity +
    residual.sugar + chlorides + free.sulfur.dioxide + total.sulfur.dioxide +
    density + sulphates + alcohol, family = binomial, data = data_train1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1302  -0.8391   0.2978   0.8034   2.2719

Coefficients:
                       Estimate Std. Error z value Pr(>|z|)
(Intercept)           2.185e+02  8.988e+01   2.431 0.015064 *
fixed.acidity         1.526e-01  7.757e-02   1.968 0.049119 *
volatile.acidity     -3.181e+00  5.175e-01  -6.146 7.94e-10 ***
residual.sugar        1.184e-01  7.310e-02   1.619 0.105379
chlorides            -6.540e+00  1.848e+00  -3.539 0.000402 ***
free.sulfur.dioxide   2.345e-02  1.043e-02   2.248 0.024602 *
total.sulfur.dioxide -2.252e-02  3.755e-03  -5.998 2.00e-09 ***
density              -2.254e+02  9.017e+01  -2.499 0.012444 *
sulphates             2.817e+00  5.622e-01   5.011 5.41e-07 ***
alcohol               5.646e-01  1.146e-01   4.926 8.39e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1321.99  on 956  degrees of freedom
Residual deviance:  967.76  on 947  degrees of freedom
AIC: 987.76

Number of Fisher Scoring iterations: 4
```

The column “Estimate” indicates the estimated increase of the log-odds (logit), given a one-unit increase in the predictor value and keeping all other predictors constant. The log-odds represent the logarithm of the ratio between the probability that the quality of the wine is excellent, and the probability that the wine is bad.
The estimated intercept is typically not of interest; its main purpose is to adjust the average fitted probabilities to the proportion of ones in the data.
We know that the estimates give the average change in Y associated with a one-unit increase in X. For example, the Estimate of Fixed Acidity is about 0.15; this indicates that an increase in Fixed Acidity is associated with an increase in the probability of an excellent wine. To be precise, a one-unit increase in Fixed Acidity is associated with an increase in the log-odds by 0.15 units.
Many aspects of the logistic regression output shown are similar to the linear regression output. For example, we can measure the accuracy of the coefficient estimates by computing their standard errors using the z value and the p-value. A small p-value indicates that it is unlikely to observe such substantial association between the predictor and the response due to the chance, in absence of any real association between the predictor and the response (i.e. under the null hypothesis). In other words, if we see a small p-

value, then we can infer that there is an association between the predictor and the response. Typical p-value cutoffs for rejecting the null hypothesis are 5% or 1%. Since the p-values of all the variables except Residual Sugar are very low, we can reject the null hypothesis.
So, we can conclude that there is indeed an association between the other variables and the probability of having an excellent wine.
Moreover, since the estimate of Volatile Acidity, for example, is negative, we can say that if this chemical substance increases, then the probability of having a good wine decreases.

The **goodness of fit** of the logistic regression model can be assessed by the deviance, which is a measure based on the log-likelihood.
The deviance is a measure of goodness of fit of a generalized linear model. Or rather, it is a measure of badness of fit, where higher numbers indicate worse fit. R reports two forms of deviance: the *Null Deviance*, that shows how well the response variable is predicted by a model that includes only the intercept, and the *Residual Deviance*, that shows how well the response is predicted by the model when the predictors are included. If the Null Deviance is really small, it means that the Null Model explains the data pretty well. Likewise with the Residual Deviance.
In this case, we have a value of 1321.99 on 956 degrees of freedom for Null Deviance. Including the independent variables decreased the deviance to 967.76 on 947 degrees of freedom, which is a significant reduction.
Hence, the Residual Deviance has reduced by 354.23 with a loss of 9 degrees of freedom.
The log-likelihood function is maximized by the Fisher scoring algorithm, that in this case needed 4 iterations to perform the fit.

Let's have a look at the **Confusion Matrix** and at the **Misclassification rate**:

```
> confMat1

glm.pred1   0   1 Sum
      0   326 124 450
      1   119 388 507
      Sum 445 512 957

> delta1
[1] 25.39185
```

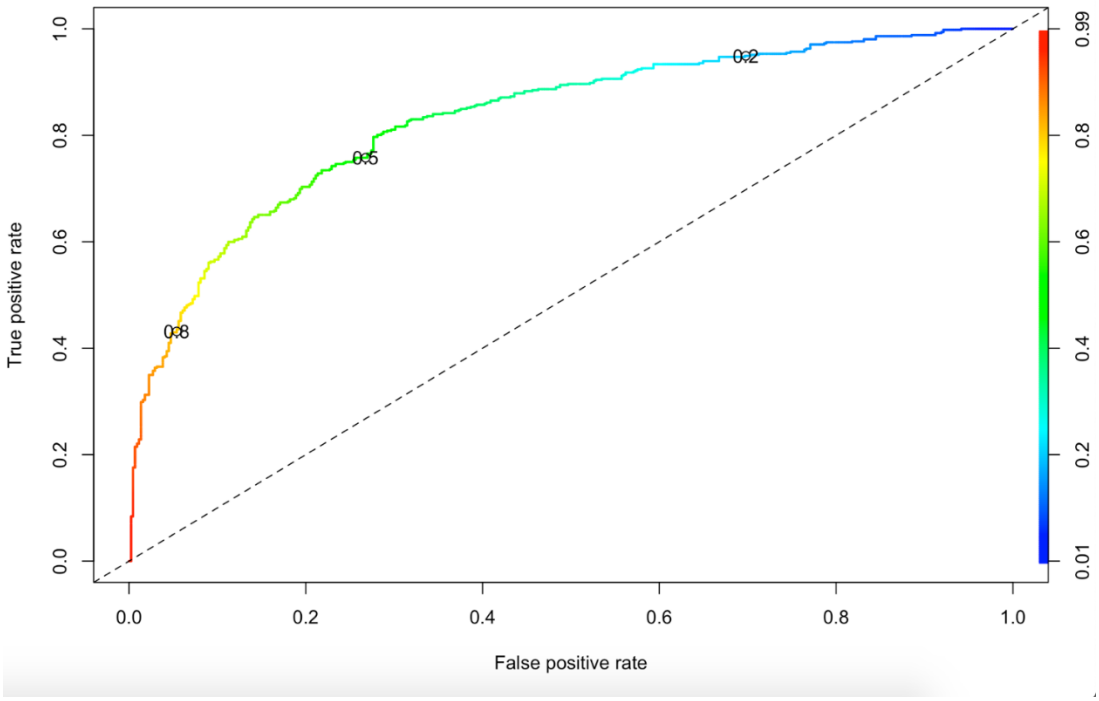
Elements on the main diagonal of the confusion matrix represent wines whose quality statuses were correctly predicted, while off-diagonal elements represent statuses that were misclassified. The relative percentage of the latter is given by the misclassification error, which is equal to 25.39%.

```
> tpr
[1] 75.78
> fpr
[1] 26.74
```

Here we show that for a 0.5 threshold: the **TPR** (true positive rate) – it is the percentage of true excellent wines that have been identified and it is equal to the sensitivity: - is equal to 75.78%; while the **FPR** (false positive rate) – it is the percentage of bad wines that have been classified as excellent ones and it is equal to 1 minus specificity - is equal to 26.74%.

ROC curve:

To assess the accuracy of a continuous measurement for predicting a binary outcome we use the Receiver Operating Characteristic curve (**ROC**).
The overall performance of a classifier is given by the area under the ROC curve (**AUC**).
An ideal ROC curve will hug the top left corner, so the larger the AUC the better the classifier.
A good ROC curve is close to the ideal ROC curve, indicating a high true positive rate and a low false positive rate.




```
> AUC@y.values[[1]]
[1] 0.8309384
```

Since the ideal case is AUC=1 and we have obtained AUC=0.83, we can conclude that we have a good result, which indicates a high true positive rate and a low false positive rate.

The model has been trained using the data train dataset. Now we will use the **data validation set** to sum up, using the misclassification error computation: smaller the error, more the model has learned the generating process of the data.
The data validation set contains 324 observations of 12 variables, and 0% of unknown values:

```
> str(data_validation)
'data.frame':   324 obs. of  12 variables:
 $ fixed.acidity       : num  5 12.5 11.6 10.5 9.9 7.4 7.5 12 10.2 8.5 ...
 $ volatile.acidity    : num  1.02 0.46 0.58 0.59 0.5 ...
 $ citric.acid         : num  0.04 0.49 0.66 0.49 0.24 0 0.48 0.5 0.37 0.4 ...
 $ residual.sugar      : num  1.4 4.5 2.2 2.1 2.3 4.25 2.6 1.4 2.2 6.3 ...
 $ chlorides           : num  0.045 0.07 0.074 0.07 0.103 0.097 0.073 0.071 0.057 0.05 ...
 $ free.sulfur.dioxide : num  41 26 10 14 6 5 22 6 14 3 ...
 $ total.sulfur.dioxide: num  85 49 47 47 14 14 84 26 36 10 ...
 $ density             : num  0.994 0.998 1.001 0.999 0.998 ...
 $ pH                  : num  3.75 3.05 3.25 3.3 3.34 3.63 3.32 3.07 3.23 3.28 ...
 $ sulphates           : num  0.48 0.57 0.57 0.56 0.52 0.54 0.7 0.6 0.49 0.56 ...
 $ alcohol             : num  10.5 9.6 9 9.6 10 10.7 9.6 10.4 9.3 12 ...
 $ quality             : int  4 4 4 4 4 4 4 4 4 4 ...
```

Now, let's calculate the misclassification error of the model on the validation set:

```
> confMat1val

glm.pred1val   0   1 Sum
         0   110  48 158
         1    41 125 166
         Sum 151 173 324
> delta1val
[1] 9.299896
```

The misclassification error on the validation set is about 9.3%, with a threshold of 0.5, and it is lower than the one computed on the training set, which is 25.39%.

B) SUPPORT VECTOR MACHINES

The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using *kernels*. A kernel is a function that quantifies the similarity of two observations and it could be linear, polynomial or radial.
Let's start with **linear kernel**, which quantifies the similarity of a pair of observations using Pearson (standard) correlation:

```
> dsvc = data.frame(model = names_svc, cost = format(cost, scientific=F), misc.error = error_svc)

  model    cost misc.error
1  svc1   0.001  0.4660494
2  svc2   0.010  0.4660494
3  svc3   0.100  0.4506173
4  svc4   1.000  0.3796296
5  svc5   5.000  0.3858025
6  svc6  10.000  0.3827160
7  svc7 100.000  0.3734568


> bmsvc = dsvc[which.min(dsvc$misc.error), ]
> bmsvc$misc.error = bmsvc$misc.error*100
> names(bmsvc)= c("Best SVC model", "cost", "Misclassification error on validation set (%)")

  Best SVC model    cost
7           svc7  100.000
  Misclassification error on validation set (%)
7                                      37.34568
```

Here we can see that the best model is svc7, indeed it has the smaller misclassification error (37.35%).
This model has the cost parameter equal to 100.000. The cost argument allows us to specify the cost of a violation to the margin. When the cost argument is small, then the margins will be wide and many support vectors will be on the margin or will violate the margin. When the cost argument is large, then the margins will be narrow and there will be few support vectors on the margin or violating the margin.

Let's continue with **polynomial kernel**. Using such a kernel instead of the standard linear one in the support vector classifier algorithm leads to a much more flexible decision boundary. It amounts to fitting a support vector classifier in a higher-dimensional space involving polynomials of degree d, rather than in the original feature space. When the support vector classifier is combined with a non-linear kernel, the resulting classifier is known as a support vector machine:

```
> dpsvm = data.frame(model = names_psvm, cost = rep(format(cost, scientific = F ), 9), degree = r
ep(degree, each = 7), misc.error = error_psvm)
> dpsvm
    model    cost degree misc.error
1   psvm1   0.001      2  0.4660494
2   psvm2   0.010      2  0.4660494
3   psvm3   0.100      2  0.4660494
4   psvm4   1.000      2  0.4660494
5   psvm5   5.000      2  0.4660494
6   psvm6  10.000      2  0.4691358
7   psvm7 100.000      2  0.4104938
8   psvm8   0.001      3  0.4660494
```

```
9   psvm9   0.010      3  0.4660494
10 psvm10   0.100      3  0.4660494
11 psvm11   1.000      3  0.4660494
12 psvm12   5.000      3  0.4660494
13 psvm13  10.000      3  0.4660494
14 psvm14 100.000      3  0.4660494
15 psvm15   0.001      4  0.4660494
16 psvm16   0.010      4  0.4660494
17 psvm17   0.100      4  0.4660494
18 psvm18   1.000      4  0.4660494
19 psvm19   5.000      4  0.4660494
20 psvm20  10.000      4  0.4660494
21 psvm21 100.000      4  0.4660494
22 psvm22   0.001      5  0.4660494
23 psvm23   0.010      5  0.4660494
24 psvm24   0.100      5  0.4660494
25 psvm25   1.000      5  0.4660494
26 psvm26   5.000      5  0.4660494
27 psvm27  10.000      5  0.4660494
28 psvm28 100.000      5  0.4660494
29 psvm29   0.001      6  0.4660494
30 psvm30   0.010      6  0.4660494
31 psvm31   0.100      6  0.4660494
32 psvm32   1.000      6  0.4660494
33 psvm33   5.000      6  0.4660494
34 psvm34  10.000      6  0.4660494
35 psvm35 100.000      6  0.4660494
36 psvm36   0.001      7  0.4660494
37 psvm37   0.010      7  0.4660494
38 psvm38   0.100      7  0.4660494
39 psvm39   1.000      7  0.4660494
40 psvm40   5.000      7  0.4660494
41 psvm41  10.000      7  0.4660494
42 psvm42 100.000      7  0.4660494
43 psvm43   0.001      8  0.4660494
44 psvm44   0.010      8  0.4660494
45 psvm45   0.100      8  0.4660494
46 psvm46   1.000      8  0.4660494
47 psvm47   5.000      8  0.4660494
48 psvm48  10.000      8  0.4660494
49 psvm49 100.000      8  0.4660494
50 psvm50   0.001      9  0.4660494
51 psvm51   0.010      9  0.4660494
52 psvm52   0.100      9  0.4660494
53 psvm53   1.000      9  0.4660494
54 psvm54   5.000      9  0.4660494
55 psvm55  10.000      9  0.4660494
56 psvm56 100.000      9  0.4660494
57 psvm57   0.001     10  0.4660494
58 psvm58   0.010     10  0.4660494
59 psvm59   0.100     10  0.4660494
60 psvm60   1.000     10  0.4660494
61 psvm61   5.000     10  0.4660494
62 psvm62  10.000     10  0.4660494
63 psvm63 100.000     10  0.4660494
```

```
> bmpsvm
  Best polynomial SVM model     cost      degree
7                     psvm7    100.000      2
  Misclassification error on validation set (%)
7                                   41.04938
```

In this case the best polynomial model is **psvm7** with a misclassification error of 41%.

Now, let's try **radial kernel**, which is another popular choice:

```
> drsvm = data.frame(model = names_rsvm, cost = rep(format(cost, scientific = F), 5), g
amma = rep(gamma, each = 7), misc.error = error_rsvm)
> drsvm
    model    cost gamma misc.error
1   rsvm1   0.001   0.5  0.4660494
2   rsvm2   0.010   0.5  0.4660494
3   rsvm3   0.100   0.5  0.4537037
4   rsvm4   1.000   0.5  0.3858025
5   rsvm5   5.000   0.5  0.3827160
6   rsvm6  10.000   0.5  0.3827160
7   rsvm7 100.000   0.5  0.3611111
8   rsvm8   0.001   1.0  0.4660494
9   rsvm9   0.010   1.0  0.4660494
10 rsvm10   0.100   1.0  0.4012346
11 rsvm11   1.000   1.0  0.3888889
12 rsvm12   5.000   1.0  0.3796296
13 rsvm13  10.000   1.0  0.3765432
14 rsvm14 100.000   1.0  0.3209877
15 rsvm15   0.001   2.0  0.4660494
16 rsvm16   0.010   2.0  0.4660494
17 rsvm17   0.100   2.0  0.3888889
18 rsvm18   1.000   2.0  0.3888889
19 rsvm19   5.000   2.0  0.3796296
20 rsvm20  10.000   2.0  0.3734568
21 rsvm21 100.000   2.0  0.3271605
22 rsvm22   0.001   3.0  0.4660494
23 rsvm23   0.010   3.0  0.4660494
24 rsvm24   0.100   3.0  0.3888889
25 rsvm25   1.000   3.0  0.3919753
26 rsvm26   5.000   3.0  0.3827160
27 rsvm27  10.000   3.0  0.3703704
28 rsvm28 100.000   3.0  0.3055556
29 rsvm29   0.001   4.0  0.4660494
30 rsvm30   0.010   4.0  0.4660494
31 rsvm31   0.100   4.0  0.3919753
32 rsvm32   1.000   4.0  0.3858025
33 rsvm33   5.000   4.0  0.3858025
34 rsvm34  10.000   4.0  0.3703704
35 rsvm35 100.000   4.0  0.2901235
```

```
> bmrsvm
   Best radial SVM model    cost gamma Misclassification error on validation set (%)
35               rsvm35 100.000     4                                        29.01235
```

In this case the best model is **rsvm35** with a misclassification error of 29%.

Now we can compare the models and choose the best one:

```
               Best_model    Misc_error
LinearK                 7       37.34568
PolynomialK             7       41.04938
RadialK                35       29.01235
```

The best model is **rsvm35** with the smallest misclassification error of 29%. Let's see it in detail:

```
> summary(RSVM$rsvm35)

Call:
svm(formula = as.factor(data_train_svm$ValueQuality) ~ fixed.acidity +
    volatile.acidity + residual.sugar + chlorides + free.sulfur.dioxide +
    total.sulfur.dioxide + density + sulphates + alcohol, data = data_train_svm,
    kernel = "radial", cost = c, gamma = g, scale = F)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  100

Number of Support Vectors:  676

 ( 337 339 )


Number of Classes:  2
```

C) NEURAL NETWORKS:

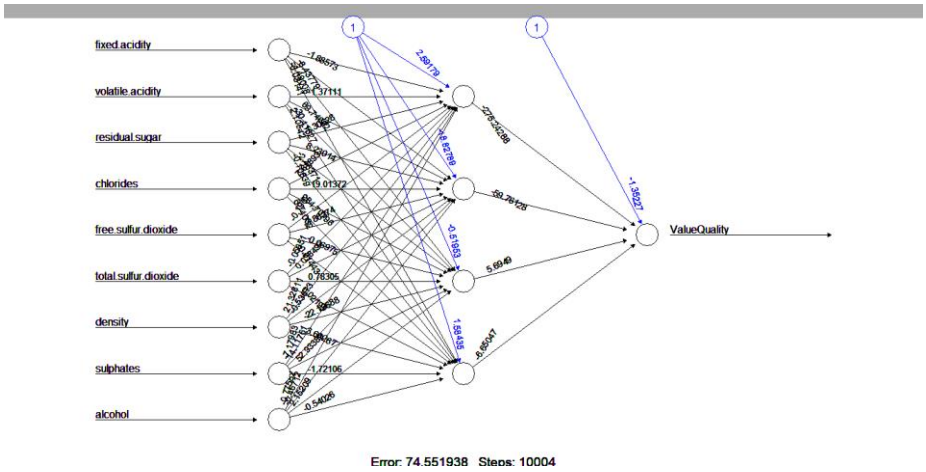
The last approach we use to model the train data is the neural network. We constructed 7 neural networks: the first 5 neural networks have only one hidden layer with a number of neurons ranging from 1 to 5; the sixth has two hidden layers: the first with 2 neurons and the second with only one; the seventh neural network has 2 hidden layers: 3 neurons the first and 2 the second.
We increased the threshold value from 0.01 to 0.1, in order to speed up the computation; we increased also the stepmax value, in order to avoid convergence errors; finally we used 3 repetition for training our neural network.

```
> names_nn
[1] "nn1" "nn2" "nn3" "nn4" "nn5" "nn6" "nn7"
> hidden
[1] "1"     "2"     "3"     "4"     "5"     "(2,1)" "(3,2)"
> head(error_nn, 10)
[1] 0.2654321 0.2654321 0.2901235 0.2438272 0.2870370 0.2685185 0.2561728
```

```
> bmnn
  Best nn model hidden Misclassification error on valida tion set (%)
4           nn4      4                                          24.38272
```

As we can see from the last code, the best model is nn4 with smallest misclassification error of 24.38%.

```
> plot(nn4)
```



This is the neural network 4, the best one according to our analysis. This is a neural network with one hidden layer with 4 neurons.

Comparison between models:

Logistic regression:

```
> delta1
[1] 25.39185
```

Support Vector machines:

```
> bmsvc
  Best SVC model    cost Misclassification error on validation set (%)
7           svc7 100.000                                       37.34568

> bmpsvm
  Best polynomial SVM model    cost degree
7                     psvm7 100.000      2
  Misclassification error on validation set (%)
7                                    41.04938

> bmrsvm
   Best radial SVM model    cost gamma
35                rsvm35 100.000     4
   Misclassification error on validation set (%)
35                                      29.01235
```

Neural Networks:

```
> bmnn
  Best nn model hidden Misclassification error on validation set (%)
4           nn4      4                                        24.38272
```

We can conclude that the best model for our data is the **neural network**, in particular the nn4 with the smallest misclassification error of 24.38%. So, we can apply this model to the test set.

PREDICT TARGET VALUES

Let's have a look at the structure of our test set:

```
> str(data_test)
'data.frame':    318 obs. of  11 variables:
 $ fixed.acidity       : num  8.2 8.1 8.8 6.5 4.6 6.9 10.4 8.3 8.2 7.5 ...
 $ volatile.acidity    : num  0.915 0.73 0.61 0.67 0.52 1.09 0.61 1.02 0.78 0.755 ...
 $ citric.acid         : num  0.27 0 0.3 0 0.15 0.06 0.49 0.02 0 0 ...
 $ residual.sugar      : num  2.1 2.5 2.8 4.3 2.1 2.1 2.1 3.4 2.2 1.9 ...
 $ chlorides           : num  0.088 0.081 0.088 0.057 0.054 0.061 0.2 0.084 0.089 0.084
...
 $ free.sulfur.dioxide : num  7 12 17 11 8 12 5 6 13 6 ...
 $ total.sulfur.dioxide: int  23 24 46 20 65 31 16 11 26 12 ...
 $ density             : num  0.996 0.998 0.998 0.995 0.993 ...
 $ pH                  : num  3.26 3.38 3.26 3.45 3.9 3.51 3.16 3.48 3.37 3.34 ...
 $ sulphates           : num  0.47 0.46 0.51 0.56 0.56 0.43 0.63 0.49 0.46 0.49 ...
 $ alcohol             : num  10 9.6 9.3 11.8 13.1 11.4 8.4 11 9.6 9.7 ...
```

We predicted 318 values according to the convention that 0 stands for bad quality wines, while 1 stands for excellent quality wines:

```
> predicted_values
  [1] "0" "0" "0" "1" "1" "0" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "1" "0" "0" "0"
 [21] "0" "1" "1" "0" "0" "0" "1" "0" "0" "0" "0" "1" "0" "0" "0" "1" "0" "0" "0" "0"
 [41] "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "1" "1" "0" "0" "0" "1" "0"
 [61] "0" "0" "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "1" "0" "0" "1" "0" "0" "0" "0"
 [81] "0" "0" "1" "0" "0" "0" "0" "0" "0" "0" "0" "0" "1" "1" "0" "0" "0" "1" "0" "1"
[101] "0" "0" "0" "0" "0" "1" "0" "1" "1" "0" "1" "0" "1" "1" "0" "0" "0" "0" "0" "0"
[121] "0" "0" "0" "1" "0" "0" "0" "1" "0" "0" "0" "0" "0" "1" "0" "0" "1" "1" "0" "0"
[141] "0" "1" "1" "0" "0" "1" "1" "0" "0" "1" "0" "1" "0" "1" "1" "1" "0" "1" "0" "0"
[161] "1" "1" "0" "0" "0" "1" "0" "0" "1" "0" "1" "1" "1" "1" "0" "0" "1" "1" "1" "1"
[181] "1" "1" "0" "1" "1" "0" "1" "1" "1" "0" "1" "1" "0" "1" "1" "1" "1" "1" "0" "1"
[201] "1" "1" "1" "1" "0" "1" "0" "0" "1" "0" "1" "1" "0" "0" "1" "1" "0" "0" "1" "1"
[221] "0" "1" "0" "1" "1" "0" "1" "1" "0" "1" "1" "1" "1" "1" "0" "1" "0" "0" "1" "0"
[241] "0" "1" "0" "1" "0" "1" "0" "1" "0" "1" "0" "1" "0" "0" "1" "1" "0" "1" "1" "1"
[261] "1" "0" "1" "1" "0" "0" "1" "1" "0" "1" "0" "0" "1" "1" "1" "1" "1" "1" "1" "1"
[281] "1" "1" "0" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1"
[301] "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1"
```

```
> table(predicted_values)
predicted_values
  0   1
161 157
```

Here we can see that there are 161 bad wines and 157 excellent wines.