Wine dataset analysis

BROINE Thomas, BRONSIN Baptiste, TCHILINGUIRIAN Théo

2025-02-17

Dataset analysis

We observe that every variable is quantitative.

- **fixed acidity** g(tartaric acid)/dmˆ3
- **volatile acidity** g(acetic acid)/dmˆ3
- **Citric acid** g/dmˆ3
- **Residual sugar** g/dmˆ3
- **Chlorides** g(sodium chloride)/dmˆ3
- **Free sulfur dioxide** mg/dmˆ3
- **Total sulfur dioxide** mg/dmˆ3
- **Density** g/cmˆ3
- **pH**
- **Sulphates** g(potassium sulphate)/dmˆ3
- **Alcohol** vol.%

Quick summary of the data

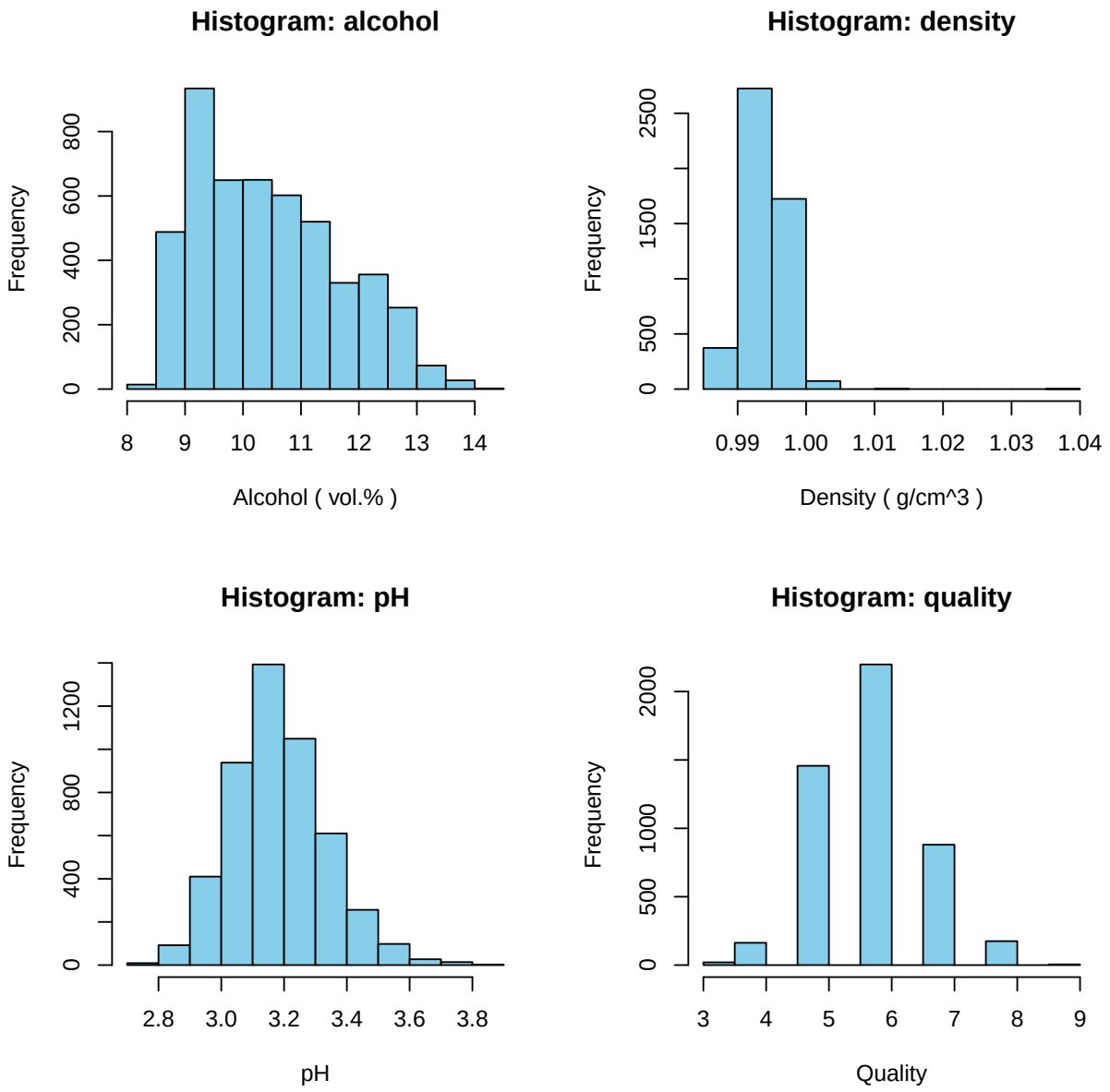
fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sulphates	alcohol	quality
6.5	0.23	0.38	1.3	0.032	29	112	0.99298	3.29	0.54	9.7	5
6.2	0.21	0.29	1.6	0.039	24	92	0.99114	3.27	0.50	11.2	6
6.6	0.32	0.36	8.0	0.047	57	168	0.99490	3.15	0.46	9.6	5
6.5	0.24	0.19	1.2	0.041	30	111	0.99254	2.99	0.46	9.4	6
5.5	0.29	0.30	1.1	0.022	20	110	0.98869	3.34	0.38	12.8	7
6.0	0.21	0.38	0.8	0.020	22	98	0.98941	3.26	0.32	11.8	6

```
##  fixed.acidity    volatile.acidity  citric.acid     residual.sugar
##  Min.   : 3.800   Min.   :0.0800   Min.   :0.0000   Min.   : 0.600
##  1st Qu.: 6.300   1st Qu.:0.2100   1st Qu.:0.2700   1st Qu.: 1.700
##  Median : 6.800   Median :0.2600   Median :0.3200   Median : 5.200
##  Mean   : 6.855   Mean   :0.2782   Mean   :0.3342   Mean   : 6.391
##  3rd Qu.: 7.300   3rd Qu.:0.3200   3rd Qu.:0.3900   3rd Qu.: 9.900
##  Max.   :14.200   Max.   :1.1000   Max.   :1.6600   Max.   :65.800
##    chlorides       free.sulfur.dioxide total.sulfur.dioxide    density
##  Min.   :0.00900   Min.   :  2.00      Min.   :  9.0        Min.   :0.9871
##  1st Qu.:0.03600   1st Qu.: 23.00      1st Qu.:108.0        1st Qu.:0.9917
##  Median :0.04300   Median : 34.00      Median :134.0        Median :0.9937
##  Mean   :0.04577   Mean   : 35.31      Mean   :138.4        Mean   :0.9940
##  3rd Qu.:0.05000   3rd Qu.: 46.00      3rd Qu.:167.0        3rd Qu.:0.9961
```

```
##  Max.   :0.34600   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.:3.090   1st Qu.:0.4100   1st Qu.: 9.50   1st Qu.:5.000
##  Median :3.180   Median :0.4700   Median :10.40   Median :6.000
##  Mean   :3.188   Mean   :0.4898   Mean   :10.51   Mean   :5.878
##  3rd Qu.:3.280   3rd Qu.:0.5500   3rd Qu.:11.40   3rd Qu.:6.000
##  Max.   :3.820   Max.   :1.0800   Max.   :14.20   Max.   :9.000
```

Histograms on important variables

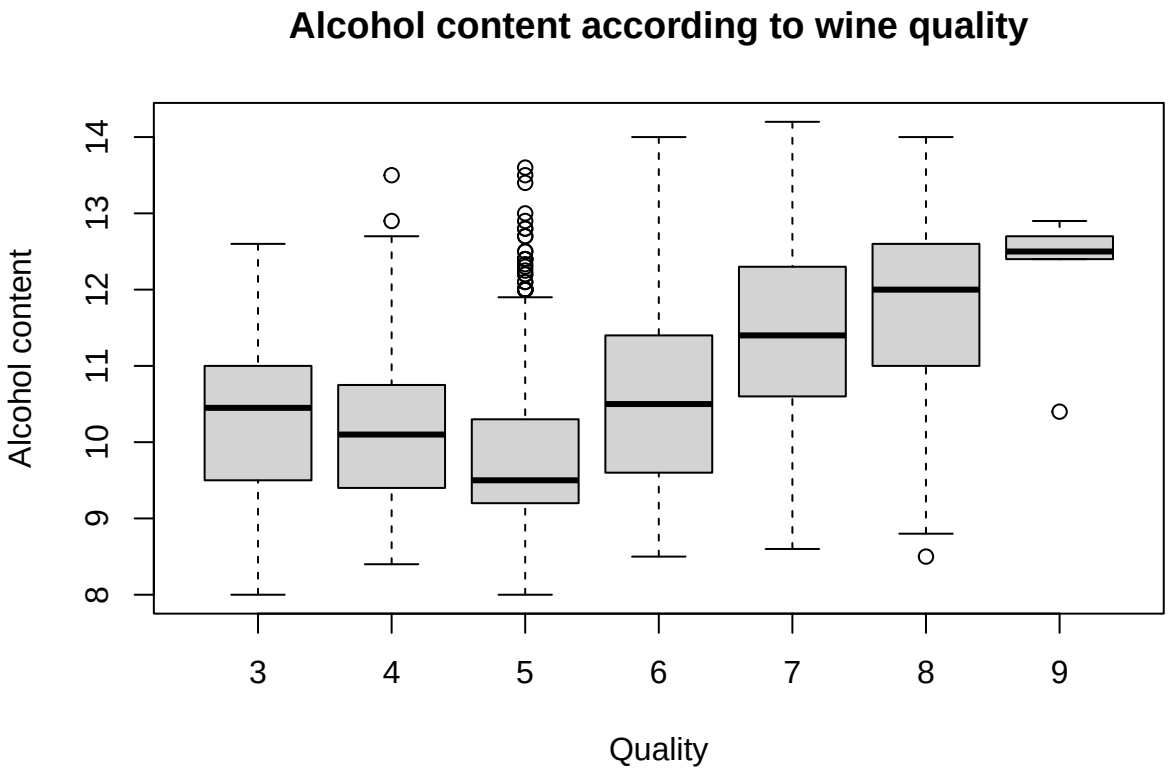
Here is an overview of some important variables that will be used for the analysis.



These histograms display the quantities of some important variables, per their values.

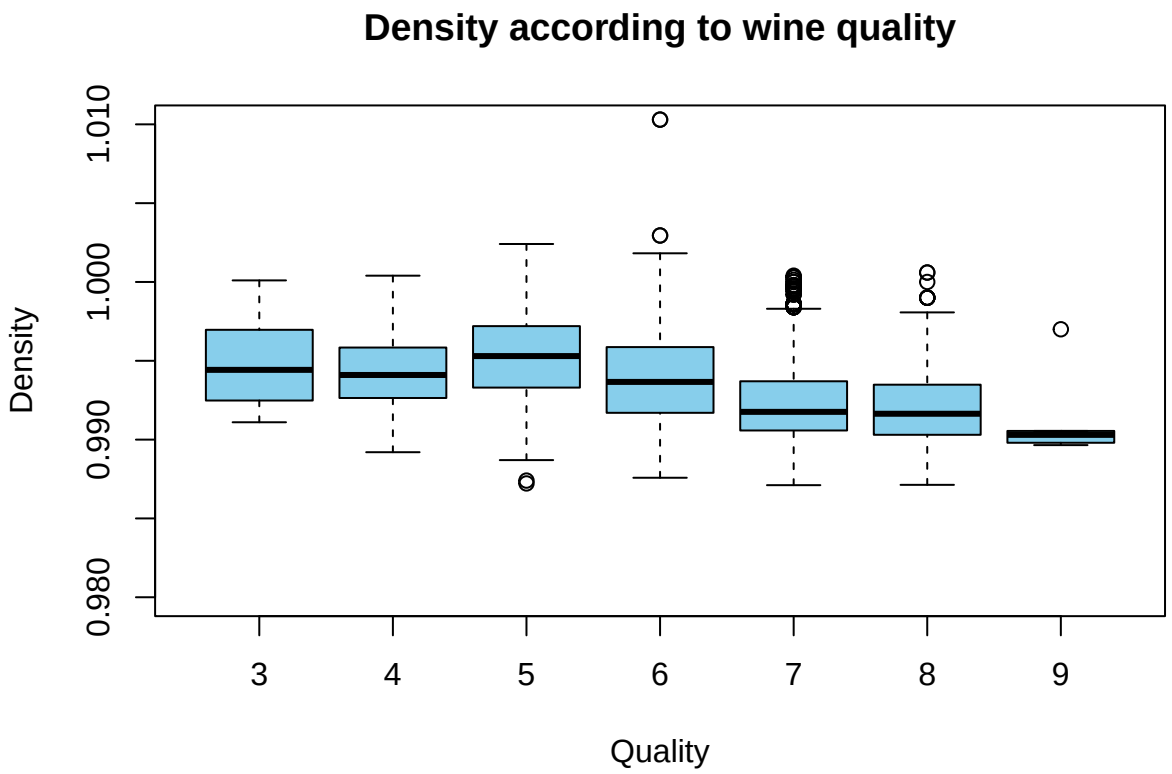
Boxplots on important variables

The following boxplots show the distribution of some important variables, per the wine quality.



Alcohol content

This boxplot shows that generally, higher wine qualities have higher alcohol content. The best quality of wine has a consistant high value in alcohol content.
However, it is important to note that there are very few data on higher quality wine (as can be seen in the above histograms).

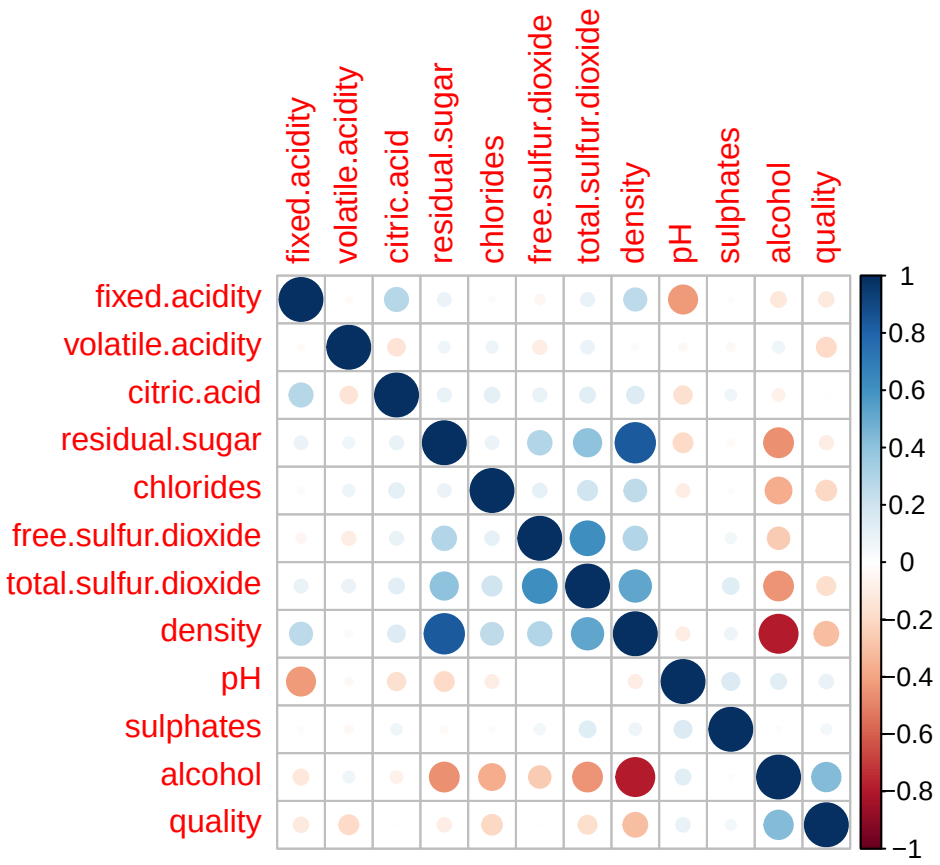


Density

This boxplot shows that higher wine qualities have lower density (that is, different contents such as sugars, alcohol, and other chemicals). The best quality of wine has a consistant low density.

Correlation

This correlation plot shows the correlation between variables.



Interpreting the results: This correlation plot shows clearly the correlations between variables.

- A positive correlation (shades of blue) means that if one the variables' value increases, the other increases.
- A negative correlation (shades of red) means that if one of the variables' value increases, the other decreases.

Obviously, wine density is inversely correlated to alcohol content. This can be easily understood when comparing the density of ethanol to that of water:

- Ethanol density : 0.7892 g/mL at 20°C.
- Water density : 0.9982 g/mL at 20°C.

And indeed, if alcohol content is higher, the density lowers, which explains the strong negative correlation.

Other obvious correlations include acidity and pH, density and residual sugar content

Basically, the density is explained by the wine's contents (alcohol, sugars, sulphates, chlorides). Thankfully for this small dataset, the correlation between these variables will not influence the wine quality prediction. For this small dataset, we don't need to eliminate variables for predictions.

Logistic binary regression

Now, we will do a binary logistic regression for predictive analysis.

First, we classify the quality as binary.

```r
winequality_white$quality_binary <- ifelse(winequality_white$quality <= 5, "bad", "good")

# We want a factor (category)
winequality_white$quality_binary <- as.factor(winequality_white$quality_binary)
```

```r
# Separate the data into training and test sets
set.seed(123)
trainIndex <- createDataPartition(winequality_white$quality, p = .8,
                                  list = FALSE,
                                  times = 1)
wineTrain <- winequality_white[ trainIndex,]
wineTest  <- winequality_white[-trainIndex,]

wineTrain_withoutQuality <- wineTrain[, !(names(wineTrain) %in% c("quality"))]
wineTrain_withoutQualityBinary <- wineTrain[, !(names(wineTrain) %in% c("quality_binary"))]

wineTest_withoutQuality <- wineTest[, !(names(wineTest) %in% c("quality"))]
wineTest_withoutQualityBinary <- wineTest[, !(names(wineTest) %in% c("quality_binary"))]
wineTest_WithoutAllQuality <- wineTest[, !(names(wineTest) %in% c("quality", "quality_binary"))]

dim(wineTrain)
```

```
## [1] 3919   13
```

```r
dim(wineTest)
```

```
## [1] 979  13
```

```r
# Binary logistic regression
logit_model <- glm(quality_binary ~ fixed.acidity + volatile.acidity +
                                    citric.acid + residual.sugar + chlorides +
                                    free.sulfur.dioxide + total.sulfur.dioxide +
                                    density + pH + sulphates + alcohol,
                   data = wineTrain_withoutQuality, # We need here to predict the binary quality
                   family = binomial)

# Résumé du modèle
summary(logit_model)
```

```
##
## Call:
## glm(formula = quality_binary ~ fixed.acidity + volatile.acidity +
##     citric.acid + residual.sugar + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + density + pH + sulphates + alcohol,
##     family = binomial, data = wineTrain_withoutQuality)
##
## Coefficients:
##                        Estimate Std. Error z value Pr(>|z|)
## (Intercept)           2.566e+02  7.969e+01   3.220 0.001282 **
## fixed.acidity         7.632e-03  8.039e-02   0.095 0.924368
## volatile.acidity     -6.485e+00  4.621e-01 -14.034  < 2e-16 ***
## citric.acid           2.536e-01  3.424e-01   0.741 0.458867
```

```
## residual.sugar        1.645e-01  3.019e-02   5.447 5.11e-08 ***
## chlorides            -1.240e-01  1.875e+00  -0.066 0.947274
## free.sulfur.dioxide   9.199e-03  3.095e-03   2.972 0.002956 **
## total.sulfur.dioxide -2.155e-03  1.358e-03  -1.587 0.112414
## density              -2.686e+02  8.077e+01  -3.326 0.000881 ***
## pH                    1.090e+00  4.062e-01   2.684 0.007275 **
## sulphates             1.967e+00  4.054e-01   4.852 1.22e-06 ***
## alcohol               7.158e-01  1.051e-01   6.808 9.92e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4996.8  on 3918  degrees of freedom
## Residual deviance: 3945.4  on 3907  degrees of freedom
## AIC: 3969.4
##
## Number of Fisher Scoring iterations: 5
```

```r
# Prédiction sur l'ensemble de test (probabilité d'être "bon")
predict_prob_logit <- predict(logit_model, newdata = wineTest_WithoutAllQuality, type = "response") # W

# Pour obtenir la classe prédite (0/1) selon un cutoff=0.5
predict_class_logit <- ifelse(predict_prob_logit <= 0.5, "bad", "good")
predict_class_logit <- factor(predict_class_logit, levels = c("bad", "good"))

# Table de confusion
conf_mat_logit <- table(Predicted = predict_class_logit,
                        Actual = wineTest$quality_binary)
conf_mat_logit
```

```
##          Actual
## Predicted bad good
##      bad  147   78
##      good 181  573
```

```r
# Accuracy
accuracy_logit <- sum(diag(conf_mat_logit)) / sum(conf_mat_logit)
accuracy_logit
```

```
## [1] 0.7354443
```

Decision Tree Model

```r
# Entraîner le modèle avec un seul arbre de décision
tree_model <- rpart(quality ~ ., data = wineTrain_withoutQualityBinary, method = "class") # We need her

# Afficher l'arbre de décision
print(tree_model)
```

```
## n= 3919
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 3919 2160 6 (0.0046 0.032 0.3 0.45 0.18 0.035 0.001)
##    2) alcohol< 10.625 2299 1296 6 (0.0048 0.041 0.43 0.44 0.079 0.012 0.00043)
##      4) volatile.acidity>=0.2375 1455  673 5 (0.0048 0.056 0.54 0.36 0.038 0.0027 0.00069)
##        8) alcohol< 9.85 992  384 5 (0.005 0.053 0.61 0.3 0.027 0.001 0) *
##        9) alcohol>=9.85 463  237 6 (0.0043 0.06 0.38 0.49 0.063 0.0065 0.0022)
##         18) free.sulfur.dioxide< 24.5 150   78 5 (0 0.13 0.48 0.33 0.047 0.0067 0) *
##         19) free.sulfur.dioxide>=24.5 313  137 6 (0.0064 0.026 0.33 0.56 0.07 0.0064 0.0032) *
##      5) volatile.acidity< 0.2375 844  365 6 (0.0047 0.017 0.24 0.57 0.15 0.027 0) *
##    3) alcohol>=10.625 1620  864 6 (0.0043 0.019 0.12 0.47 0.33 0.067 0.0019)
##      6) alcohol< 12.55 1334  670 6 (0.0045 0.022 0.13 0.5 0.29 0.054 0.0015) *
##      7) alcohol>=12.55 286  142 7 (0.0035 0.007 0.031 0.32 0.5 0.13 0.0035) *
```

```
# Faire des prédictions sur l'ensemble de test
tree_predictions <- predict(tree_model, wineTest_WithoutAllQuality, type = "class") # We remove all qua

# Afficher les prédictions
head(tree_predictions)
```

```
## 1 2 3 4 5 6
## 6 5 6 6 5 6
## Levels: 3 4 5 6 7 8 9
```

```
# Calculer la précision
tree_accuracy <- sum(tree_predictions == wineTest$quality) / nrow(wineTest)

# Afficher la précision
print(paste("Précision :", round(tree_accuracy * 100, 2), "%"))
```

```
## [1] "Précision : 51.89 %"
```

There are two types of log loss calculations: one for binary classification problems and one for multi-class classification problems. In this case, we have a binary classification problem, so we will use the log loss formula for binary classification.

```
y_true <- ifelse(wineTest$quality_binary == "good", 1, 0)

# Convertir les facteurs en numériques
tree_predictions_numeric <- as.numeric(as.character(tree_predictions))

# Convertir les scores de qualité en probabilités
probabilities <- tree_predictions_numeric / 10

# Fonction pour calculer le log loss
log_loss <- function(y_true, probabilities, eps=1e-15){
  probabilities <- pmin(pmax(probabilities, eps), 1 - eps)
  -mean(y_true * log(probabilities) + (1 - y_true) * log(1 - probabilities))
}
```

```r
# Calcul de la log loss
ll_rf <- log_loss(y_true, probabilities)
print(paste("Log loss :", ll_rf))
```

```
## [1] "Log loss : 0.615091454941158"
```

We also calculate the Brier score of the model.

```r
# Calculer Brier score pour le modèle decision tree
tree_probabilities <- predict(tree_model, wineTest_WithoutAllQuality)
brier_score_tree <- mean((tree_probabilities[, 2] - y_true)^2)
print(paste("Brier Score (Decision Tree):", round(brier_score_tree, 4)))
```

```
## [1] "Brier Score (Decision Tree): 0.632"
```

Random Forest Model

```r
# Entraîner le modèle de forêts aléatoires
randomForestModel <- randomForest(quality ~ ., data = wineTrain_withoutQualityBinary)

# Prédire sur l'ensemble de test
predictions <- predict(randomForestModel, wineTest_WithoutAllQuality)

# Arrondir les prédictions pour obtenir des nombres entiers
rounded_predictions <- round(predictions)

# Afficher les prédictions
head(rounded_predictions)
```

```
## 1 2 3 4 5 6
## 6 6 6 6 6 6
```

```r
# Évaluer le modèle
eval <- postResample(pred = rounded_predictions, obs = wineTest$quality)

# Afficher les métriques d'évaluation
eval
```

```
##      RMSE  Rsquared       MAE
## 0.6580995 0.4600931 0.3677222
```

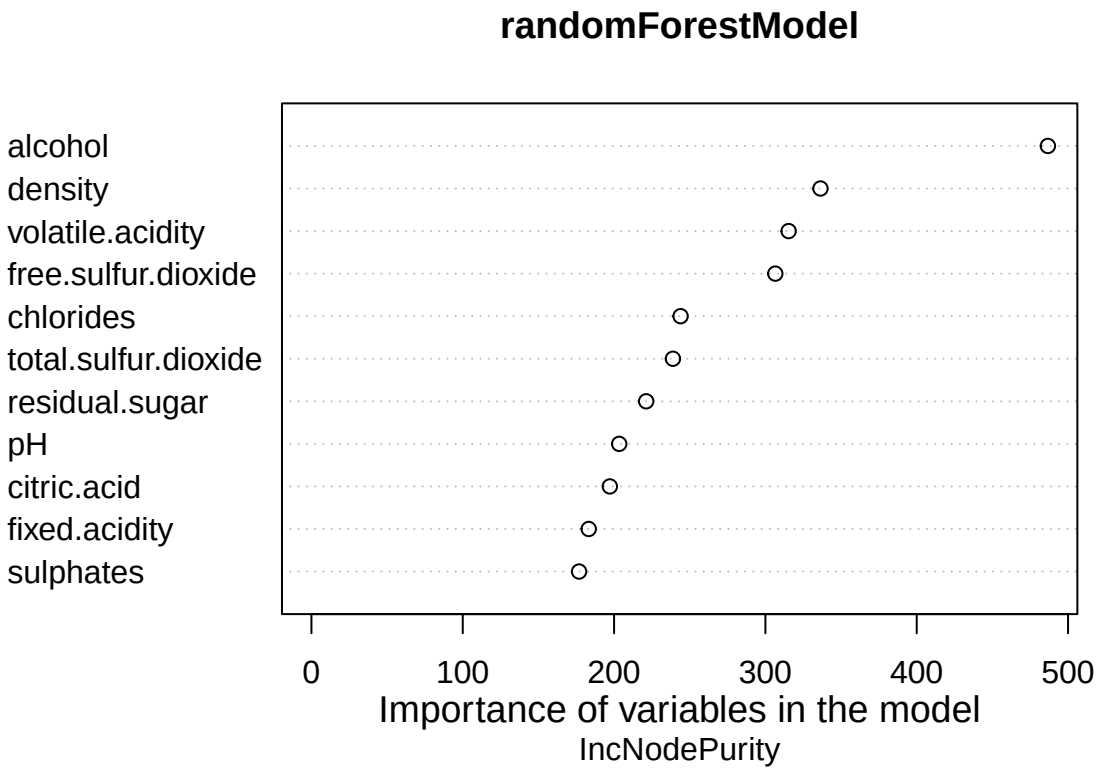
```r
# Calculer la précision
forest_accuracy <- sum(rounded_predictions == wineTest$quality) / nrow(wineTest)

# Afficher la précision
print(paste("Précision :", round(forest_accuracy * 100, 2), "%"))
```

```
## [1] "Précision : 66.29 %"
```

```r
# Importer les variables importantes
importance(randomForestModel)
```

```
##                      IncNodePurity
## fixed.acidity             183.2601
## volatile.acidity          315.3653
## citric.acid               197.2069
## residual.sugar            221.2184
## chlorides                 243.9526
## free.sulfur.dioxide       306.5367
## total.sulfur.dioxide      238.8863
## density                   336.3713
## pH                        203.3838
## sulphates                 176.8777
## alcohol                   486.6842
```

```r
varImpPlot(randomForestModel)
mtext("Importance of variables in the model", side = 1, line = 2, cex = 1.2)
```



Dans cette section, nous avons entraîné un modèle de forêts aléatoires pour prédire la qualité du vin.

```r
# Convertir les scores de qualité en probabilités
probabilities <- rounded_predictions / 10

# Calcul de la log loss
```

```r
ll_rf_forest <- log_loss(y_true, probabilities)
print(paste("Log loss :", ll_rf_forest))
```

```
## [1] "Log loss : 0.580885162206191"
```

Le log loss binaire est une mesure de performance utilisée pour évaluer la précision des prédictions probabilistes d'un modèle de classification binaire, en comparant les probabilités prédites à la véritable classe (0 ou 1) de chaque échantillon.

And we calculate the Brier score as well.

```r
# Calculer Brier score pour le modèle random forest
#...
```

Interpretation and Comparison

First, we examined the dataset, its variables and their relative importance for our predictions.
We have done predictions using two models: a single decision tree, and a random forest model.
To conclude this report, we will now compare their accuracy metrics to determine which model was the better prediction model.

These metrics help us evaluate the performance of our models, by comparing the accuracy, log loss, and Brier score between the two models.

Accuracy Comparison

The accuracy metric represents the proportion of correctly classified wine quality labels in our test set. We learnt that decision trees are a simple model that is able to capture basic patterns in the data, but are more prone to overfitting when compared to the random forest model, which aggregates multiple trees, and as such improves accuracy by reducing variance.

From our results:

- Decision tree model accuracy: 0.5188968
- Random forest model accuracy: 0.6629213

As we can see, the random forest outperforms the single decision tree in accuracy. In the forest, multiple trees contribute to the final prediction, and in this prediction, this led to a better result in accuracy.

Log Loss Comparison

The log loss is another performance metric. It measures the uncertainty of a probabilistic classification model.

From our results:

- Decision tree model log loss: 0.6150915
- Random forest model log loss: 0.5808852

A lower log loss value indicates that the predicted probabilities are closer to the true values, which is better (the model is better calibrated).

Brier Score Comparison

The Brier score measures the mean squared error between predicted probabilities and actual outcomes. A lower Brier score indicates better probabilistic predictions.

From our results:

- Decision tree model Brier score: 0.6320091
- Random forest model Brier score: Not done.

Final results

Comparing the performance metrics we were able to calculate shows that the random forest model seemingly outperforms the single decision tree, across multiple evaluation metrics.

We can conclude that the decision tree is a simple model which we can understand more easily, but by combining multiple trees, the random forest model mitigates overfitting and provides more accurate predictions.

We tried to predict wine quality using the given dataset, comparing two prediction models on multiple accuracy metrics, and we demonstrated that the random forest model comes out on top in performance.