Université de Cergy-Pontoise

Rapport Projet

Data Mining

Professeur : Jen Tao yuan
Binome : Mohamed II BAYO - Benoit FAGOT

16 mars 2020

Table des matières

Table des figures

1 Analyse de l'ensemble des données :

Dans l'optique de ce projet Data Mining, le jeu de données a été choisi sur le site web UC Irvine Machine Learning Repository. Le dataset est fourni par 'The Institute of Pharmaceutical and Food Analysis and Technologies', et propose des résultats d'analyses chimiques de vins d'Italie issus de 3 cultivars différents.

Il y a 178 objets, avec pour chacun 13 attributs tous non discrets, et un attribut discret qui représente la classe du vin. Il n'y a pas de valeurs manquantes.

Ce jeu de données est donc apte à être utilisé pour les tests de différents modèles de classifications pour déterminer la classe probable d'un nouveau vin issu de ces cultivars. Nous voulons monter que parmi ces 13 attributs, il y a certains qui permettent de déterminer à quel type de vin on a affaire avec fiabilité (au moins 80% de succès).

La distribution des classes est la suivante :

- class 1 59 instances soit 33%
- class 2 71 instances soit 40%
- class 3 48 instances soit 27%

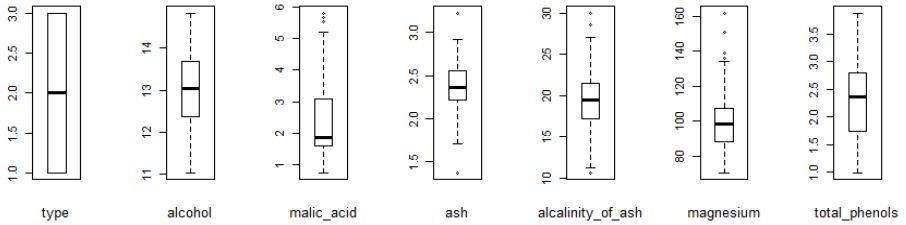
Liste des attributs :

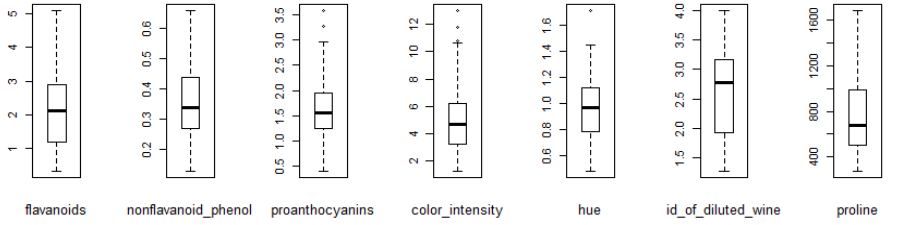
	Attributs	Valeurs Distinctes	valeurs manquantes
1	type	3	0
2	Alcohol	126	0
3	Malic acid	133	0
4	Ash	79	0
5	Alcalinity of ash	63	0
6	Magnesium	53	0
7	Total phenols	97	0
8	Flavanoids	132	0
9	Nonflavanoid phenols	39	0
10	Proanthocyanins	101	0
11	Color intensity	132	0
12	Hue	78	0
13	id of diluted wines	122	0
14	proline	121	0

1.1 Statistique du jeu de données

```
> summary(dat)
 type      alcohol        malic_acid        ash        alcalinity_of_ash
 1:59   Min.   :11.03   Min.   :0.740   Min.   :1.360   Min.   :10.60
 2:71   1st Qu.:12.36   1st Qu.:1.603   1st Qu.:2.210   1st Qu.:17.20
 3:48   Median :13.05   Median :1.865   Median :2.360   Median :19.50
        Mean   :13.00   Mean   :2.336   Mean   :2.367   Mean   :19.49
        3rd Qu.:13.68   3rd Qu.:3.083   3rd Qu.:2.558   3rd Qu.:21.50
        Max.   :14.83   Max.   :5.800   Max.   :3.230   Max.   :30.00
  magnesium      total_phenols     flavanoids    nonflavanoid_phenol proanthocyanins
 Min.   : 70.00   Min.   :0.980   Min.   :0.340   Min.   :0.1300      Min.   :0.410
 1st Qu.: 88.00   1st Qu.:1.742   1st Qu.:1.205   1st Qu.:0.2700      1st Qu.:1.250
 Median : 98.00   Median :2.355   Median :2.135   Median :0.3400      Median :1.555
 Mean   : 99.74   Mean   :2.295   Mean   :2.029   Mean   :0.3619      Mean   :1.591
 3rd Qu.:107.00   3rd Qu.:2.800   3rd Qu.:2.875   3rd Qu.:0.4375      3rd Qu.:1.950
 Max.   :162.00   Max.   :3.880   Max.   :5.080   Max.   :0.6600      Max.   :3.580
 color_intensity       hue         id_of_diluted_wine    proline
 Min.   : 1.280   Min.   :0.4800   Min.   :1.270      Min.   : 278.0
 1st Qu.: 3.220   1st Qu.:0.7825   1st Qu.:1.938      1st Qu.: 500.5
 Median : 4.690   Median :0.9650   Median :2.780      Median : 673.5
 Mean   : 5.058   Mean   :0.9574   Mean   :2.612      Mean   : 746.9
 3rd Qu.: 6.200   3rd Qu.:1.1200   3rd Qu.:3.170      3rd Qu.: 985.0
 Max.   :13.000   Max.   :1.7100   Max.   :4.000      Max.   :1680.0
```

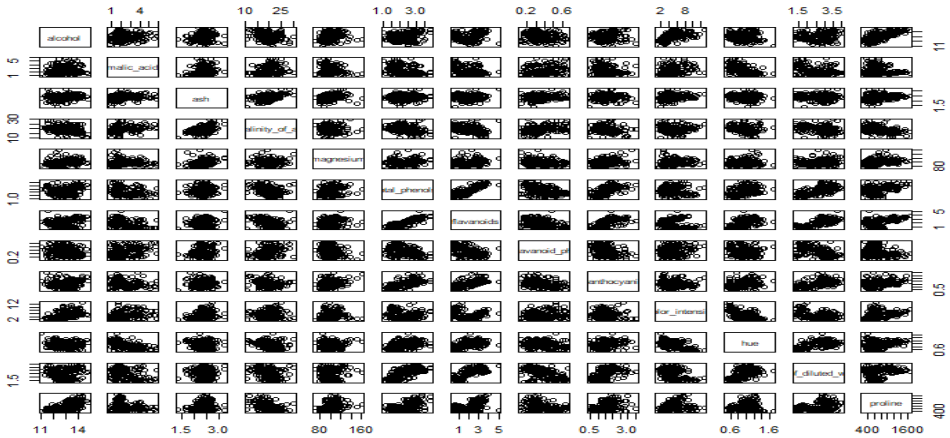
Le résumé des statistiques montre que la plupart des variables ont un large éventail par rapport à l'IQR, ce qui peut indiquer une dispersion dans les données et la présence de valeurs aberrantes. Nous approfondissons nos recherches en produisant des boîtes à moustaches pour chacune des variables



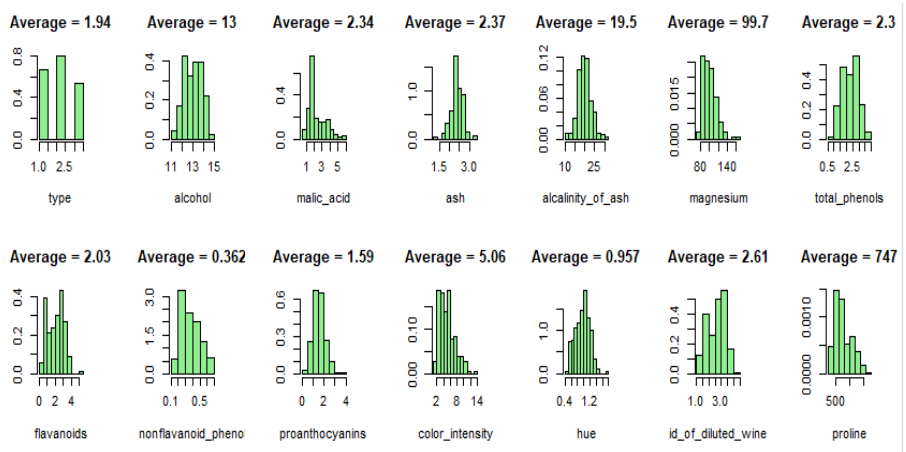


Il démontre que les variables : Acalinity of ash, Magnesium, Ash, Hue, Color intensity, Proanthocyanins, Malic acid contiennent des valeurs aberrantes.

Nous utilisons maintenant une matrice de nuage de points pour obtenir un aperçu des emplacements aberrants :



Maintenant, nous regardons la distribution des valeurs des prédicteurs :



1.2 Analyse des attributs par rapport a l'attribut type

1.2.1 Alcohol

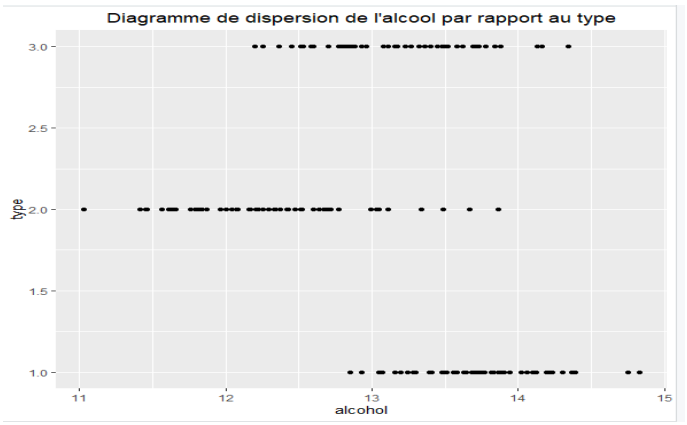


FIGURE 1 – Diagramme de dispersion d'Alcohol par rapport au type

on observe qu'un alcohol inférieur à 12.3 est forcement de type 2, cela pourrait aider l'arbre de décision

1.2.2 Malic acid

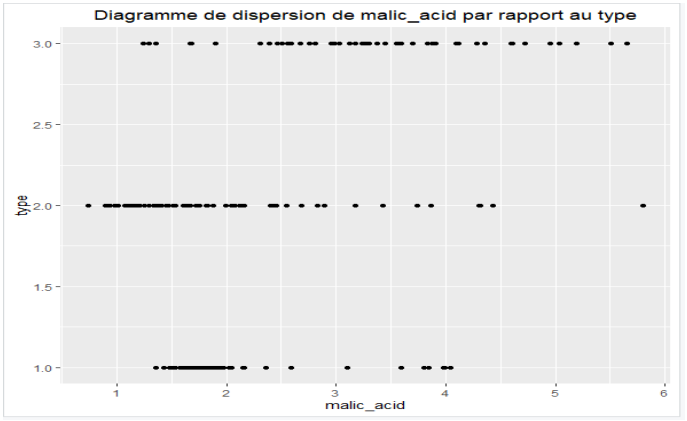


FIGURE 2 – Diagramme de dispersion de Malic acid par rapport au type

Ici on peut conjecturer que les acides maliques supérieur à 4.5 sont de type 3 malgré la présence d'un outlier de type2,on émet plus de réserve sur la pertinence de cet attribut

1.2.3 Ash

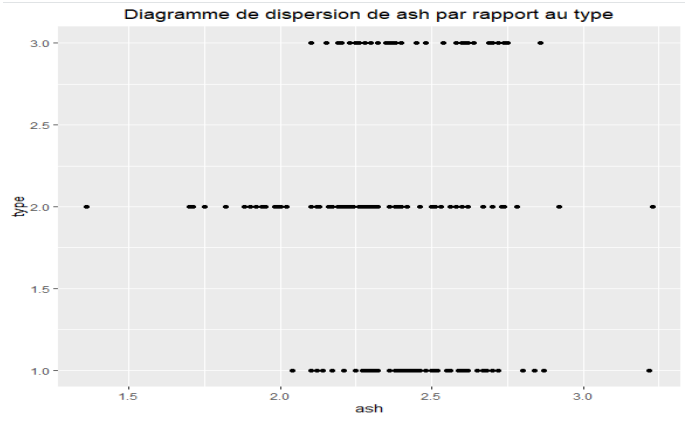


FIGURE 3 – Diagramme de dispersion d'Ash par rapport au type

On peut observer qu'un taux d'ash inférieur à 2.0 permet d'identifier des types 2, peut être utile pour détecter ce type.Cependant il n'est pas un bon indicateur pour les 2 autres types.

1.2.4 Alcalinity of ash

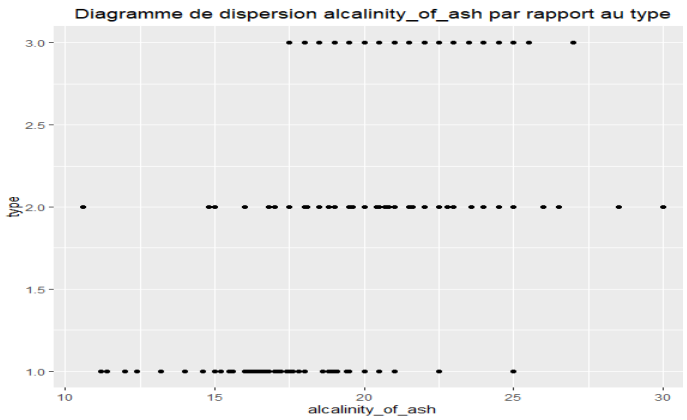


FIGURE 4 – Diagramme de dispersion d'Alcalinity of ash par rapport au type

Alcalinity of ash pourrait nous aider à identifier quelques alcohols du type 1 mais reste trop commun aux autres types.

1.2.5 Magnesium

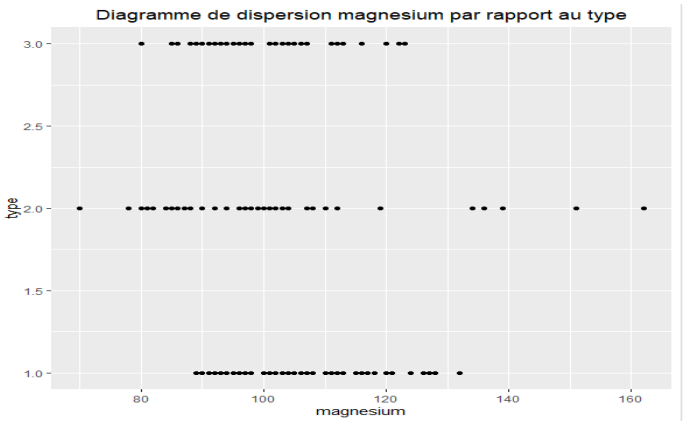


FIGURE 5 – Diagramme de dispersion du Magnesium par rapport au type

La grande majorité des valeurs de magnesium se trouvent dans l'itnervalle [80,130] et ne permet pas de prendre de décision. Cet attribut est très sûrement à écarter

1.2.6 Total phenols

Cet attribut est plutôt bien réparti pour prendre une décision pour les types 1 et 3, et on a vu précédemment que le type 2 était identifiable par 2 autres attributs.

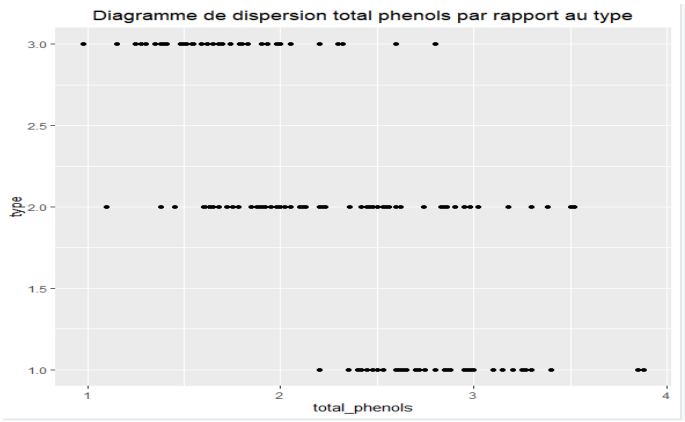


FIGURE 6 – Diagramme de dispersion du Total phenols par rapport au type

1.2.7 Flavanoids

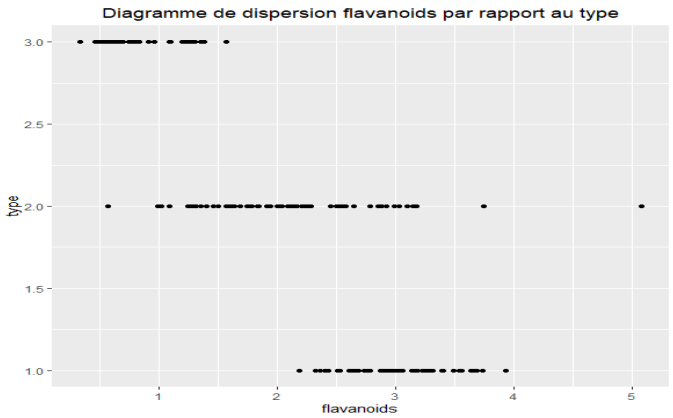


FIGURE 7 – Diagramme de dispersion du Flavanoids par rapport au type

Là encore cet attribut sépare bien les types 1 et 3

1.2.8 Nonflavanoid phenols

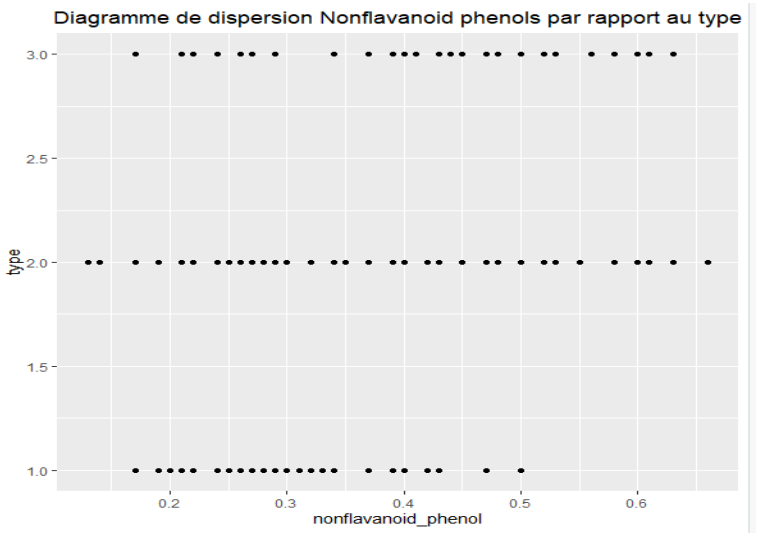


FIGURE 8 – Diagramme de dispersion du Nonflavanoid phenols par rapport au type

nonflavanoid_phenol ne permet pas d'identifier correctement les types, trop commun.

1.2.9 Proanthocyanins

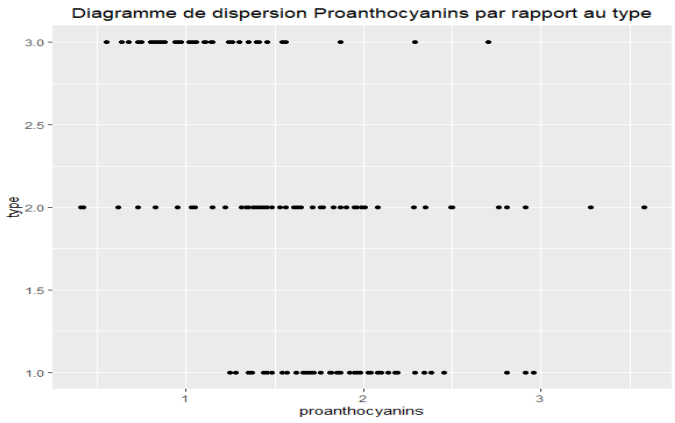


FIGURE 9 – Diagramme de dispersion du Proanthocyanins par rapport au type

Proanthocyanins peut permettre de trancher entre type 1/3.

1.2.10 Color intensity

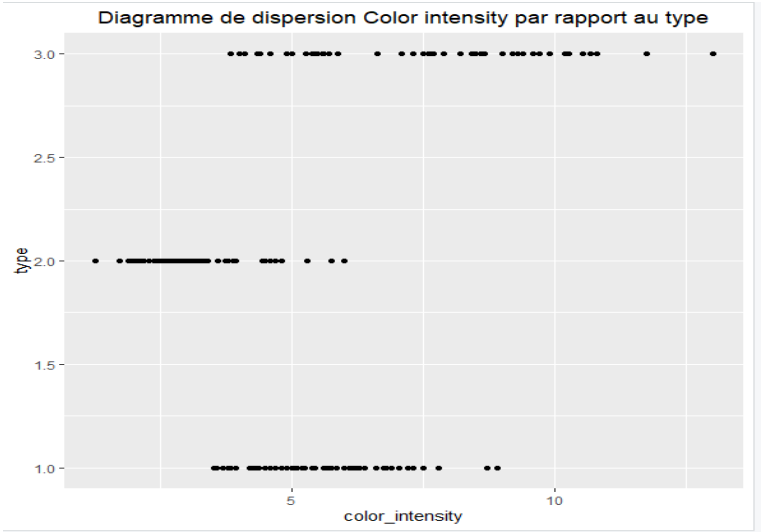


FIGURE 10 – Diagramme de dispersion du Color intensity par rapport au type

Cet attribut donne une bonne indication pour le type 2.

1.2.11 Hue

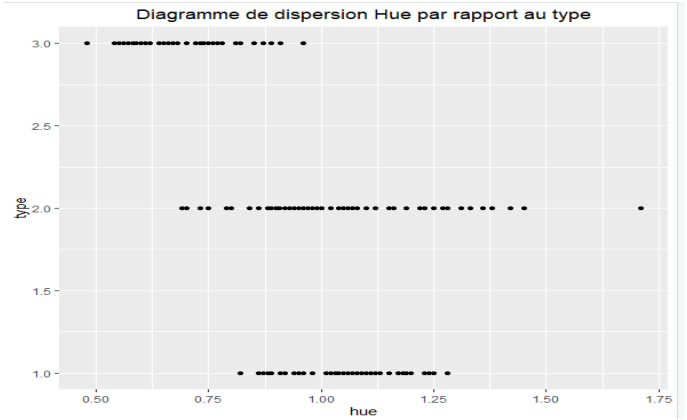


FIGURE 11 – Diagramme de dispersion du Hue par rapport au type

Cet attribut donne une bonne indication pour le type 2

1.2.12 OD280/OD315 of diluted wines

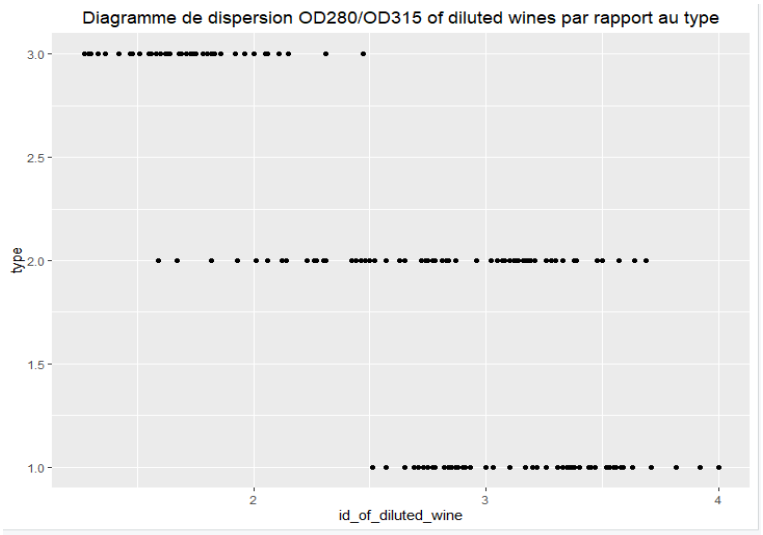


FIGURE 12 – Diagramme de dispersion du OD280/OD315 of diluted wines par rapport au type

Cet attribut découpe parfaitement les type 1 et 3

1.2.13 Proline

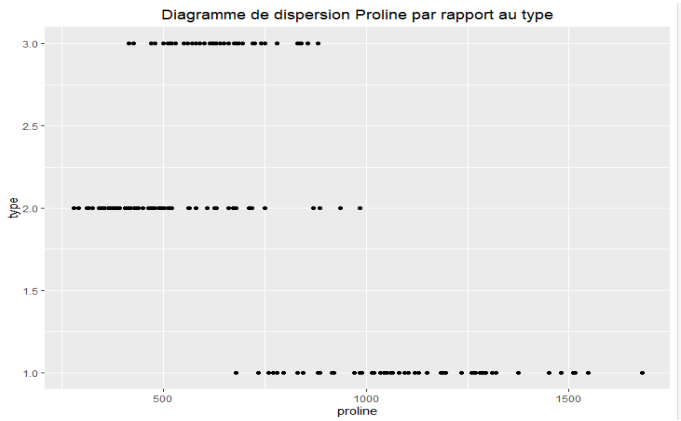


FIGURE 13 – Diagramme de dispersion du Proline par rapport au type

Cet attribut découpe très bien les trois types et peut être un bon attribut pour l'arbre

Après l'analyse sur les différents attributs, nous retenons color_intensity, flavanoids, id_of_diluted_wine et proline comme attributs pour la décision

2 Arbre de décision

Les arbres de décisions permettent de catégoriser un objet en se basant sur des attributs catégoriques ou continus. Ils sont donc objectivement adaptés au jeu de données choisi.

L'objectif va être de trouver les attributs qui sont les plus déterministes puis de chercher les règles qui servent de décision afin de trancher la direction prise dans l'arbre.
On utilise une librairie rpart qui analyse les attributs donnés et détermine ces attributs clés / règles à évaluer, puis génère l'arbre de décision correspondant.
On sépare notre jeu de données avec un ratio 0.7/0.3, respectivement pour le jeu de données d'apprentissage et d'entraînement.

Listing 1 – Méthode rpart - arbre de décision

```
set.seed(12345)
ind <- sample(2, nrow(data_test), replace=TRUE, prob=c(0.8, 0.2))
data_train <- dat2[ind==1,]
data_test <- dat2[ind==2,]

library(rpart)
library(rpart.plot)
fit <- rpart(type~., data = data_train, method = 'class')
rpart.plot(fit, extra = 104)
```

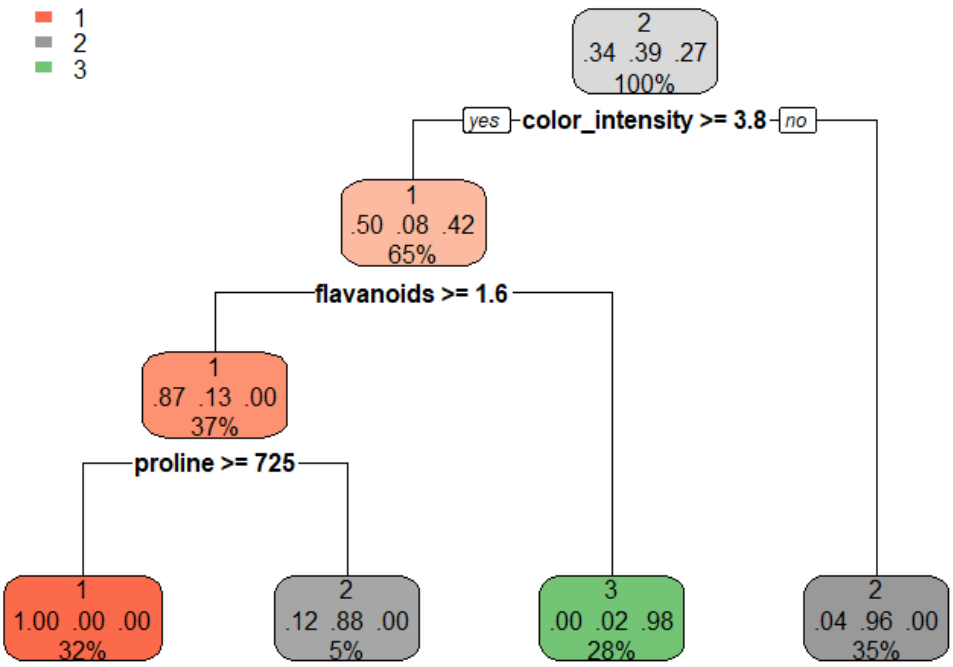


FIGURE 14 – Graphe arbre de décision

Pour évaluer la performance de notre arbre, nous calculons la matrice de confusion.

Listing 2 – Calcul de la matrice de confusion

```
predict_unseen <-predict(fit, data_test, type = 'class')
table_mat <- table(data_test$type, predict_unseen)
table_mat
accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)
print(paste('Accuracy for test', accuracy_Test))l
```

```
> table_mat <- table(data_test$type, predict_unseen)

> table_mat
   predict_unseen
     1  2  3
  1 10  0  0
  2  0 16  1
  3  0  1 14

> accuracy_Test <- sum(diag(table_mat)) / sum(table_mat)

> print(paste('Accuracy for test', accuracy_Test))
[1] "Accuracy for test 0.952380952380952"
```

On obtient une précision globale de 95%.

- Type 1 : Précision 100%, Rappel 100%
- Type 2 : Précision 94%, Rappel 94%
- Type 3 : Précision 93%, Rappel 93%

Ces résultats confirment que le modèle de données est parfaitement adapté au jeu de données car nous avons une précision presque parfaite, et l'arbre a bien utilisé les données que nous avions analysé comme déterministes pour la classification.

3 K-NearestNeighbors (K-NN)

Nous avons construire un classificateur kNN (k-Nearest Neighbour) pour prédire le type de vin.

k -Nearest Neighbors identifie le nombre k d'observations les plus proches de l'échantillon de test, tel que défini par une mesure de distance, par exemple euclidienne. À partir de cet ensemble de k-voisins, la règle de la majorité est utilisée pour prédire la classe. Si k = 3 et le type des vins les plus proches voisins sont 1, 1, 2 , alors nous classerions l'échantillon d'essai comme un vin de type 1. La même approche est étendue aux échantillons de test ultérieurs.

Pseudocode kNN :
Pour chaque x de l'ensemble de test :

- Calculez la distance entre x et chaque observation dans le train.
- Triez les distances par ordre croissant et obtenez les classes des k- voisins les plus proches.
- En utilisant la règle de la majorité, affectez x à la classe prédite.

Nous avons entraîner notre modèle à en utilisant la bibliothèque "carret" qui nous fournit une méthode **train()** pour la formation de nos données pour différents algorithmes. Avant la méthode **train()**, nous avons utilisé la méthode de validation croisée répétée à l'aide de la méthode **trainControl()**. La méthode trainControl() nous renvoie une liste que nous transmettrons sur la méthode train()

Listing 3 – Méthode trainControl()

```
trainCtrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

- method : contient les détails de la méthode de rééchantillonnage, nous avons utilité repeatedcv pour la validation croisée
- number : contient le nombre d'itérations de rééchantillonnage
- repeats : contient les ensembles complets de plis à calculer pour notre validation croisée répétée.

Modèle KNN :

Pour l'apprentissage du classificateur KNN, la méthode train() doit être passée avec le paramètre «method = "knn" ».

Listing 4 – Méthode train() - model KNN

```
model_knn <- train(type ~., data = df2_train, method = "knn",
                   trControl = trainCtrl,
                   preProcess = c("center", "scale"),
                   tuneLength = tunel )
```

Nous passons ensuite notre variable cible qui est "type ".

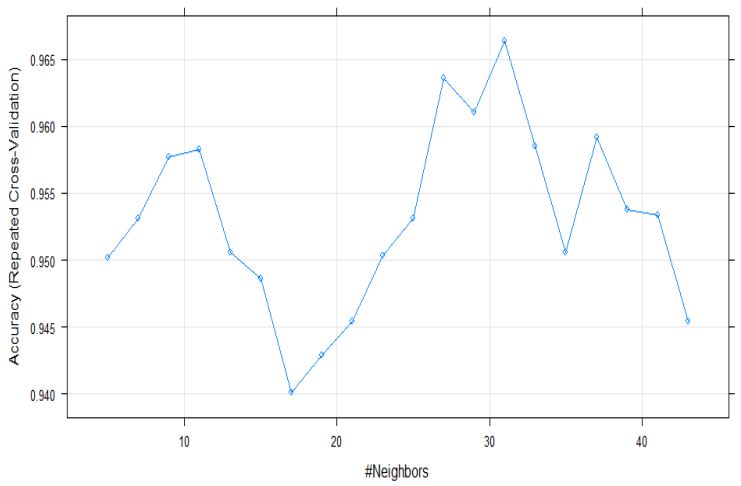
- trControl : prend les résultats de la méthode trainControl()
- preProcess : sert au prétraitement de nos données d'entrainement.

Le pré-traitement est une tâche obligatoire. Nous passons 2 valeurs dans notre paramètre «preProcess» : «center» et «scale». Ces deux éléments aident à centrer et à mettre à l'échelle les données. Après le pré-traitement, ces données convertissent nos données d'entraînement avec une valeur moyenne d'environ «0» et un écart type de «1».
tuneLength : contient une valeur entière. C'est pour régler notre algorithme.

Listing 5 – Le resultat de l'affichage de notre modele KNN

```
k-Nearest Neighbors

126 samples
  4 predictor
  3 classes: '1', '2', '3'

Pre-processing: centered (4), scaled (4)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 114, 114, 114, 113, 113, 113, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9501832  0.9246020
   7  0.9531441  0.9294288
   9  0.9576923  0.9360187
  11  0.9583028  0.9370961
  13  0.9506105  0.9255226
  15  0.9486264  0.9224533
  17  0.9400794  0.9090323
  19  0.9428571  0.9132428
  21  0.9454212  0.9171467
  23  0.9503663  0.9244135
  25  0.9531441  0.9284438
  27  0.9636142  0.9444306
  29  0.9610501  0.9405267
  31  0.9663919  0.9487784
  33  0.9584860  0.9366228
  35  0.9505800  0.9247946
  37  0.9591270  0.9378500
  39  0.9537851  0.9295010
  41  0.9533578  0.9284586
  43  0.9454518  0.9161983

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 31.
```

À partir des résultats, cette méthode train() sélectionne automatiquement la meilleure valeur de k. Ici, notre modèle de formation choisit k = 31 comme valeur finale.
Nous pouvons visualiser sur le graphique ci-dessous la variation de la précision par rapport à la valeur K



Nous pouvons également inspecter l'importance de chaque variable qui a été utilisée dans notre modèle KNN en utilisant varImp ().

Listing 6 – Resultat varImp(model$_k nn$)

```
ROC curve variable importance

  variables are sorted by maximum importance across the classes
                       X1     X2     X3
flavanoids           100.00   0.00 100.00
id_of_diluted_wine   100.00  29.60 100.00
proline               65.03  65.03  55.24
color_intensity        7.39  48.80  48.80
```

Prédiction de l'ensemble de test :

Maintenant, notre modèle est formé avec une valeur K=31. Nous avons utiliser la méthode Predict() fournit par la bibliothèque carret pour faire de la prédiction.

Listing 7 – Méthode pour la prédiction

```
prediction_knn <- predict(model_knn, newdata = test_df2)
```

Nous passons 2 arguments. Le premier paramètre est notre modèle knn formé et le second paramètre «newdata» contient notre base de données de test.

En utilisant la matrice de confusion, nous pouvons imprimer des statistiques de nos résultats. Il montre que la précision de notre modèle pour l'ensemble de test est de 94.23%.

Listing 8 – Le resultat de l'affichage de la matrice de confusion

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3
         1 16  1  0
         2  1 19  0
         3  0  1 14

Overall Statistics

               Accuracy : 0.9423
                 95% CI : (0.8405, 0.9879)
    No Information Rate : 0.4038
    P-Value [Acc > NIR] : 2.471e-16

                  Kappa : 0.9126

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3
Sensitivity            0.9412   0.9048   1.0000
Specificity            0.9714   0.9677   0.9737
Pos Pred Value         0.9412   0.9500   0.9333
Neg Pred Value         0.9714   0.9375   1.0000
Prevalence             0.3269   0.4038   0.2692
Detection Rate         0.3077   0.3654   0.2692
Detection Prevalence   0.3269   0.3846   0.2885
Balanced Accuracy      0.9563   0.9363   0.9868
```

4 Conclusion

L'objectif de ce projet était de réfléchir à des problèmes de fouille de données, choisir un jeu de données et utiliser des algorithmes pour analyser les données et tirer les conclusions par rapport aux problèmes posés.
Le problème auquel nous voulions répondre était un problème de classification : à partir d'un jeu de données caractérisant un ensemble d'items fini (classes), trouver des attributs clés qui permettent d'attribuer un nouvel item non identifié à son groupe.
Parmi les 13 attributs, notre analyse primaire a permis d'isoler 4 attributs qui semblaient être assez indicatifs pour la décision. Nous avons ensuite mis ces 4 attributs au test en utilisant deux techniques de classifications différentes : Decision Tress, K-NearestNeighbors.
Avec RStudio, nous avons pu implémenter ces algorithmes en les entraînant avec le subset d'apprentissage, et avons pu observer un succès implacable des deux méthodes (supérieur à 90%) pour identifier les items présent dans le subset de test. Il est donc clair que les attributs utilisés pour les deux algorithmes ont été judicieusement choisis.
Nous avons donc répondu au problème de classification posé, à savoir proposer un modèle permettant d'identifier les vins 1 à 3.