Université Paris-Saclay
UFR de Mathématiques

Projet d'apprentissage supervisé

# Apprentissage supervisé : data challenge

Marie GENERALI, Pierre CAVALIER

Lien du repository git

Année universitaire 2023-2024

# Table des matières

# 1 Prédiction de la qualité d'un vin

L'objectif de ce projet est d'évaluer la qualité de vins en leur attribuant une note comprise entre 1 et 10. L'étude consiste tout d'abord à une analyse exploratoire des données ayant pour but d'effectuer un nettoyage des données, de se débarrasser d'éventuelles variables inutiles ou de données aberrantes. Ensuite plusieurs modèles ont été testés : forêt aléatoire, gradient boosting et un SVR (support vector regressor).

## 1.1 Analyse exploratoire des données.

Le jeu de données est composé de 13 variables explicatives quantitatives : la clé d'identification du vin **wine_ID, fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol**, le type de vin (rouge ou blanc) **wine_type**, ainsi que le **label** -la note qui lui a été attribué- **target**.

On peut raisonnablement supposer que la valeur de la variable **wine_ID** n'entre pas réellement en jeu lors de l'attribution de la note.

Le jeu de données a été découpé en deux parties : les données d'entraînement étiquetées **train** composé de 4547 individus et les données de test, non étiquetées **test** composées de 1950 individus.

### 1.1.1 Analyse de la distribution des données par variables.

Après avoir vérifié qu'aucune donnée n'était manquante dans le jeu de données, nous faisons une analyse des distributions des différentes variables ainsi que des étiquettes.

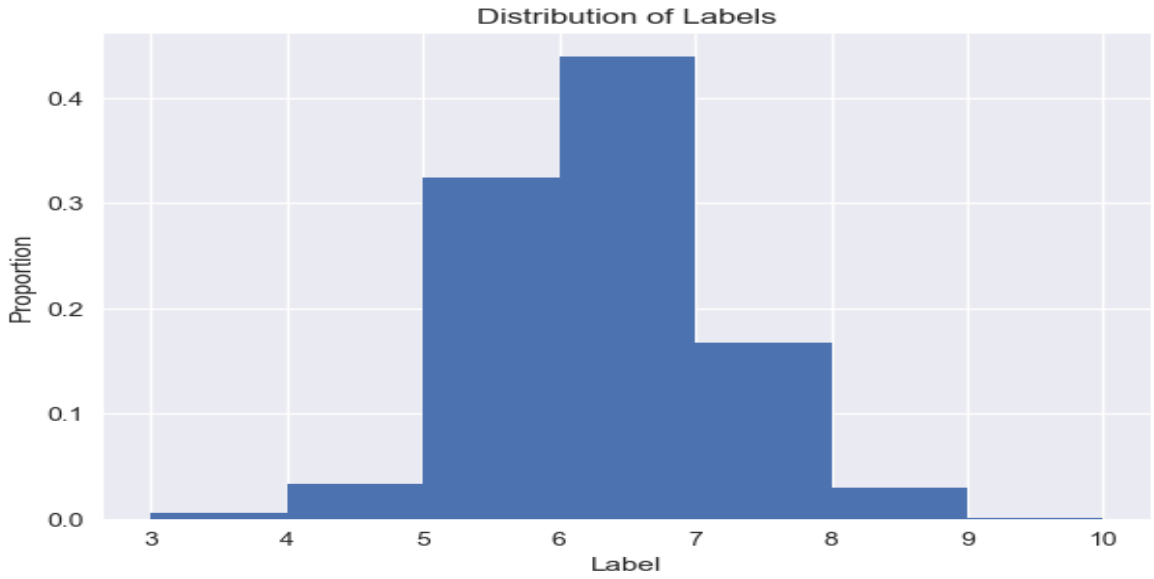



FIGURE 1 – Distribution des étiquettes.

Nous pouvons constater que les labels ne sont pas distribués manière égale.

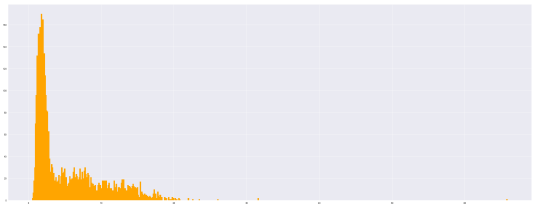
Afin de regarder les distributions des variables on affiche les boîtes a moustaches par variable ainsi que les histogrammes de leur distribution. Lors de cette analyse, on constate que certaines variables présentent des données rares comme illustré 2.

| statistic | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide |
|---|---|---|---|---|---|---|
| count | 4547.0 | 4547.0 | 4547.0 | 4547.0 | 4547.0 | 4547.0 |
| mean | 7.235363976248077 | 0.3401099626127117 | 0.31815482735869804 | 5.3675720255113255 | 0.056111941939740485 | 30.3644160985265 |
| std | 1.319966093018242 | 0.16664291920333768 | 0.14294189132029622 | 4.748314031379842 | 0.03523600907567879 | 17.8687512345704 |
| min | 3.8 | 0.08 | 0.0 | 0.6 | 0.012 | 1.0 |
| 25% | 6.4 | 0.23 | 0.25 | 1.8 | 0.038 | 17.0 |
| 50% | 7.0 | 0.29 | 0.31 | 2.9 | 0.047 | 29.0 |
| 75% | 7.7 | 0.4 | 0.39 | 8.0 | 0.065 | 41.0 |
| max | 15.6 | 1.58 | 1.0 | 65.8 | 0.611 | 289.0 |

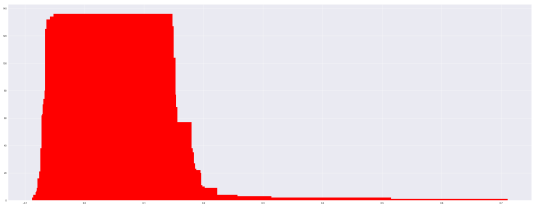
| total sulfur dioxide | density | pH | sulphates | alcohol | wine_type | target |
|---|---|---|---|---|---|---|
| 4547.0 | 4547.0 | 4547.0 | 4547.0 | 4547.0 | 4547.0 | 4547.0 |
| 115.50230921486694 | 0.9947001880360679 | 3.2194765779634924 | 0.5330063778315373 | 10.496161571732276 | 0.24983505608093248 | 5.824059819661315 |
| 56.989343262019275 | 0.0030154683929988243 | 0.16351126451772913 | 0.15072796140433778 | 1.1795639787771328 | 0.4329650421649823 | 0.8763460334684845 |
| 6.0 | 0.98713 | 2.74 | 0.22 | 8.0 | 0.0 | 3.0 |
| 77.0 | 0.99234 | 3.11 | 0.43 | 9.5 | 0.0 | 5.0 |
| 118.0 | 0.9948 | 3.21 | 0.51 | 10.3 | 0.0 | 6.0 |
| 156.0 | 0.997 | 3.32 | 0.6 | 11.3 | 0.0 | 6.0 |
| 440.0 | 1.03898 | 4.01 | 2.0 | 14.2 | 1.0 | 9.0 |

FIGURE 2 – Analyse quantitative des boîtes a moustache des variables.

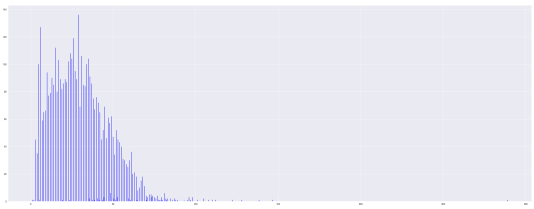
En effet, on constate que les variables, **residual sugar, chlorides, free sulfur dioxide** et **total sulfur dioxide** ont des maximums très élevés par rapport à la valeur de leur 3ème quartile. On observe alors de plus près les distributions de ces variables.



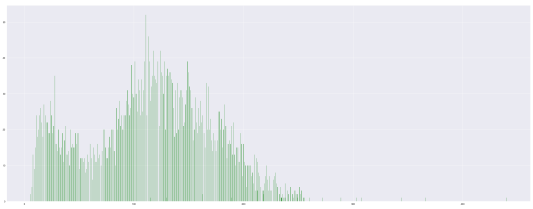
(a) Distribution Residual Sugar



(b) Distribution Chlorides



(c) Distribution Free Sulfur Dioxide



(d) Distribution Total Sulfur Dioxide

Après avoir regardé le nombre d'outlier dans les données d'entraînement et dans les données de test aucun comportement particulier n'a pu être mis en avant. L'étude a donc été poursuite sans prendre de mesures vis à vis de ces données aberrantes.

### 1.1.2 Analyse des corrélations.

Ensuite, on étudie la corrélation entre les variables définie par la relation 1

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

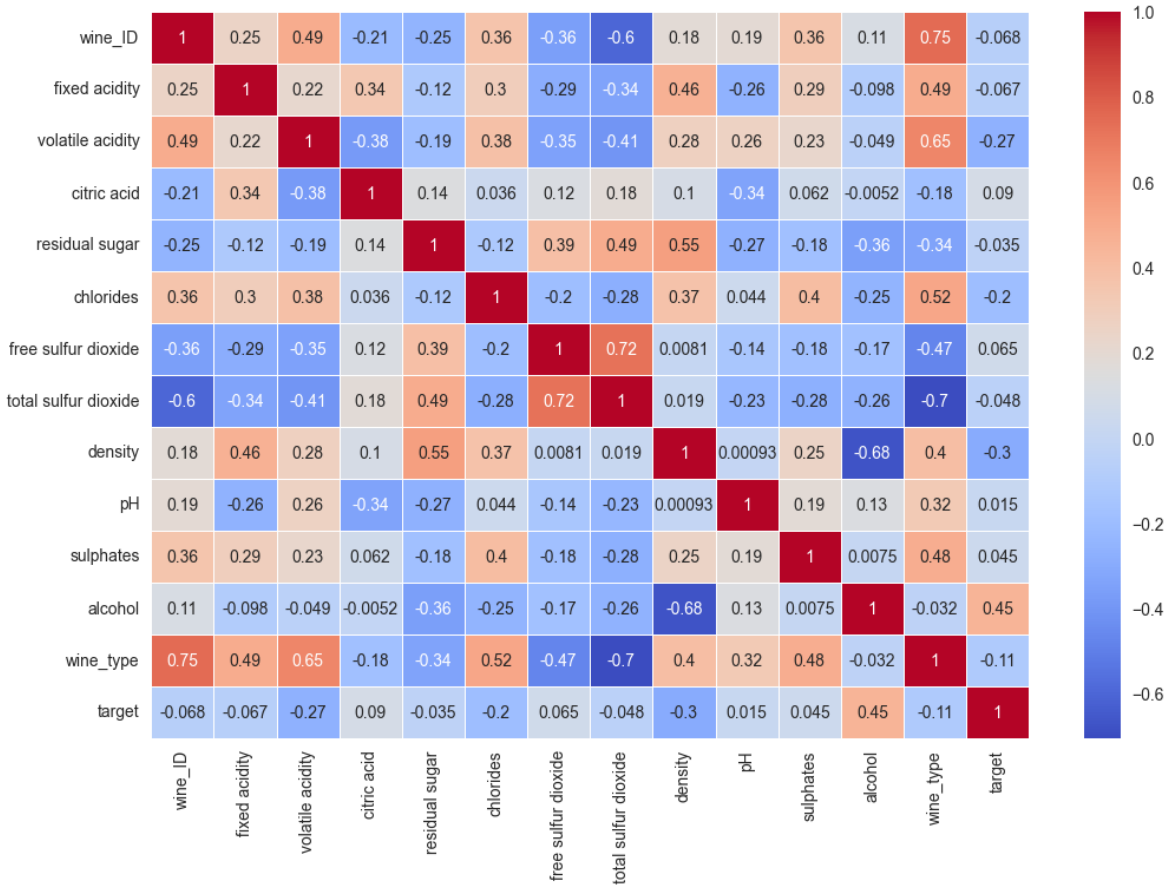


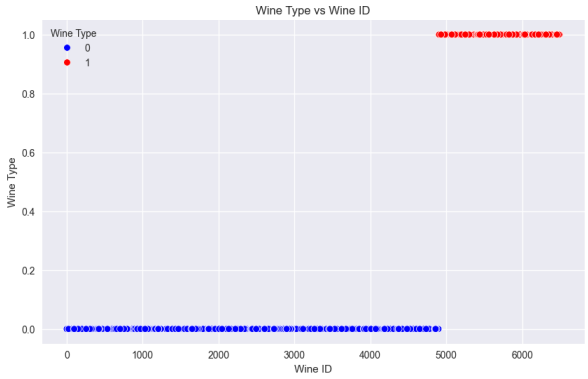

FIGURE 4 – Matrice de corrélation.

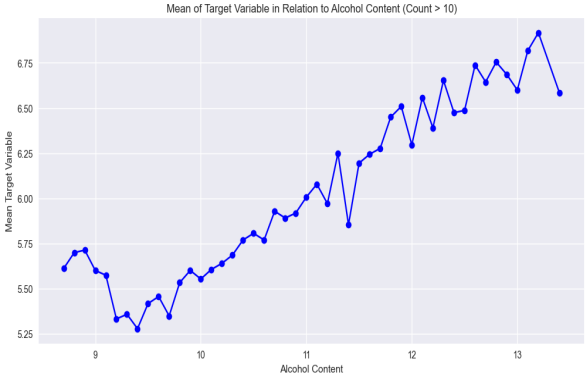
La matrice de corrélation met en évidence une forte corrélation entre la caractéristique **wine ID** et le **wine type** ce qui est assez surprenant puisqu'on pourrait penser que l'ID du vin est une variable "objective".
Les deux valeurs mesurant le dioxyde sont également (sans surprise) fortement corrélées. La densité et le taux de sucre sont positivement corrélés, ce qui n'est pas étonnant puisque le sucre est un élément très dense. Il semble également que le taux de souffre dans le vin ait un impact négatif sur sa note.
En étudiant la distribution des type de vin en fonction de leur ID (cf. fig.5a) on comprend que les types de vin sont rangés dans l'ordre : la première partie des ID sont les vins rouges et la deuxième partie sont les vins blancs. Afin d'éviter toute sur-corrélation on supprime donc la variable **wine ID** des variables explicatives.

En analysant les **corrplot** entre les données (cf. fig.5b) on constate que le niveau d'alcool a un impact positif sur la note [1]. Toutefois l'analyse des graphiques de corrélation sont assez limitées, étant donné que toutes les notes ne sont pas également représentées, d'autres conclusions ne peuvent pas vraiment être tirées : en effet, plus il y a de données d'une classe, plus il

1. Les valeurs affichées ne concernent que les niveaux d'alcool pour lesquels plus de 10 échantillons ont été observés.

est probable que ses valeurs seront étendues.




(a) Type de vin en fonction de leurs ID (vin blanc en rouge et vin rouge en bleu)

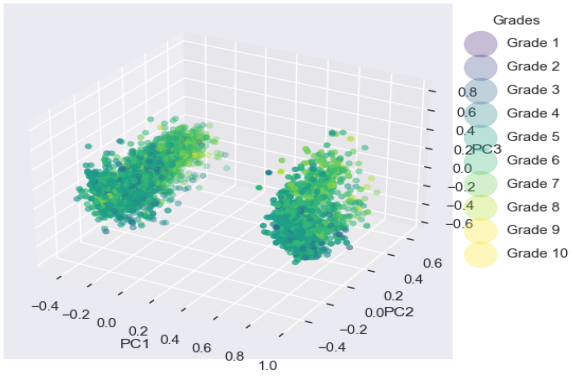
(b) Moyenne des notes en fonction du taux d'alcool (nb d'individus$>$ 10).

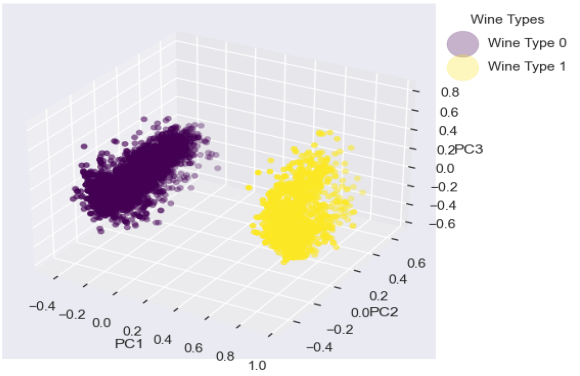
### 1.1.3 Analyse en Composantes Principales.

Pour avoir une meilleure idée des contributions des variables ainsi que des dynamiques ainsi que de l'impact des données aberrantes déterminées précédemment, nous réalisons une ACP 6a.

Les données sont correctement représentées.

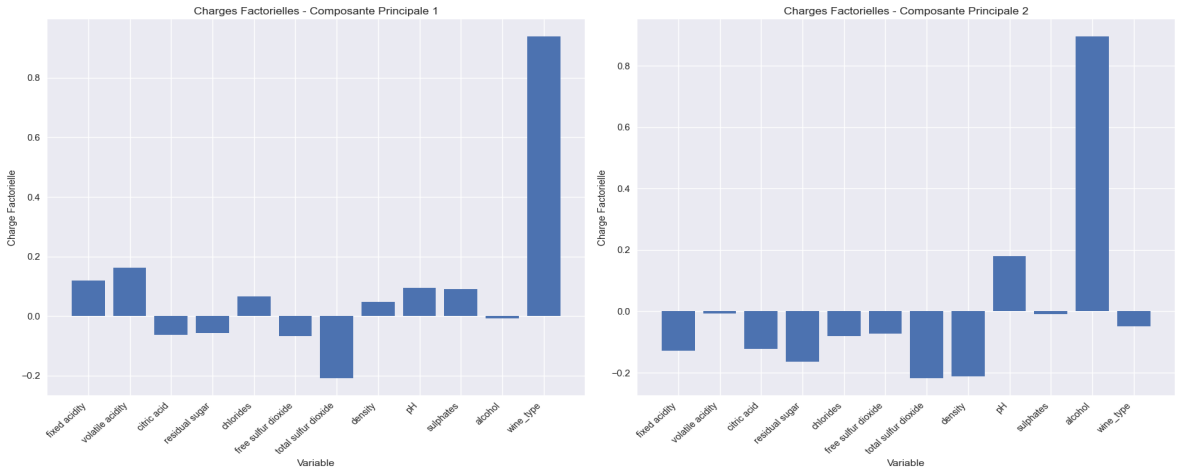
On constate que les données semblent se répartir en deux clusters mais que ces derniers ne sont pas liés à la valeur de **target**. Après observation des charges factorielles 7a il semble que les données se séparent en fonction du type de vin, cette observation se confirme en affichant l'ACP en colorant les types de vins.




(a) Trois premières CP en colorant target.

(b) Trois premières CP en colorant wine type.




(a) Charge factorielle pour la première CP. (b) Charge factorielle pour la deuxième CP.

Ces observations suggèrent qu'il pourrait être judicieux de traiter les deux types de vin séparément. L'analyse des charges factorielles permet également de constater que les variables présentant des valeurs aberrantes n'ont pas une contribution déterminante pour l'assignation des notes : il n'est donc pas nécessaire de retirer ces valeurs aberrantes du jeu de données d'entraînement.

### 1.1.4 Analyse en considérant les type de vin séparément.

Afin de vérifier qu'un travail différentié entre les deux types de vin est judicieux, nous regardons les distributions en fonction du type (cf. fig. 8).

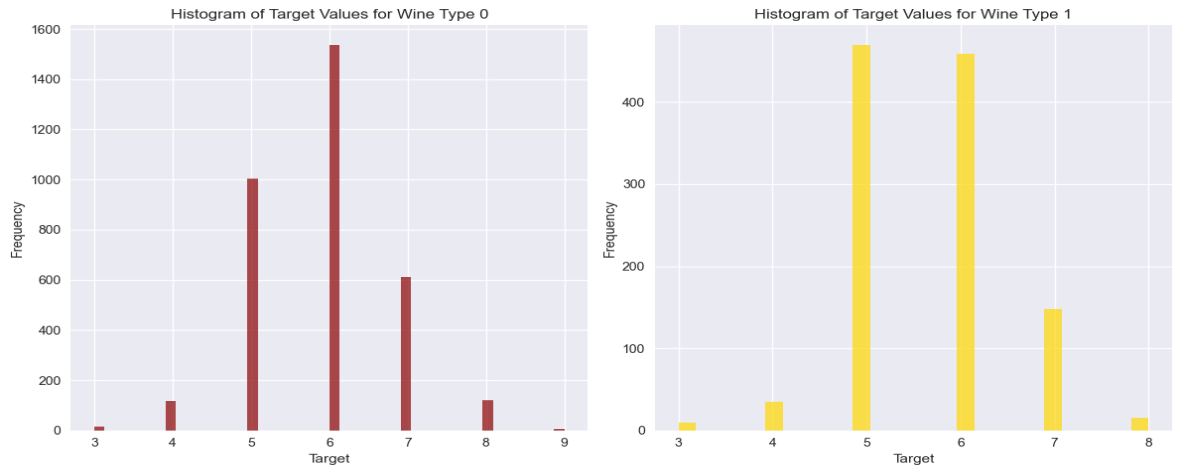



FIGURE 8 – Distribution des notes pour chaque type de vin (rouge à gauche, blanc à droite).

Les distributions ne sont pas identiques, le vin rouge présente des valeurs nettement plus élevées que le vin blanc. l'étendue des notes est également plus élevée mais cela peut être expliqué par le fait qu'il y a beaucoup plus d'échantillon de vin rouge que de vin blanc.

## 1.2 Apprentissage supervisé

Dans cette section, nous explorons différentes techniques d'apprentissage afin de construire le modèle le plus à même d'attribuer une note appropriée au vin en fonction de ses caractéristiques.
Nous séparerons le jeu de données en 80% pour les données d'entraînement, et 20% de données test.
Le mode d'évaluation est le coefficient de détermination, noté $R^2$. Le coefficient de détermination est une mesure statistique de la proportion de la variance de la variable dépendante qui est expliquée par le modèle de régression. Il est défini comme :

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

où :

— $n$ est le nombre d'observations,

— $y_i$ est la valeur observée de la variable dépendante pour l'observation $i$,

— $\hat{y}_i$ est la valeur prédite de la variable dépendante pour l'observation $i$ selon le modèle,

— $\bar{y}$ est la moyenne des valeurs observées de la variable dépendante.

Un $R^2$ proche de 1 indique que le modèle explique une grande partie de la variabilité de la variable dépendante, tandis qu'un $R^2$ proche de 0 indique que le modèle n'explique pas bien la variabilité de la variable dépendante.

### 1.2.1 Forêt aléatoire

Le premier modèle testé est un modèle de forêt aléatoire. La forêt aléatoire est un modèle d'apprentissage automatique qui combine plusieurs modèles d'arbres de décision pour améliorer la performance prédictive et réduire le surajustement. Un régresseur par forêt aléatoire utilise ce principe pour la régression.

L'idée principale derrière une forêt aléatoire est de construire un grand nombre d'arbres de décision indépendants les uns des autres, puis de moyenne (pour la régression) les prédictions de chaque arbre pour obtenir une prédiction finale plus robuste.

Le processus de construction d'un modèle de régresseur par forêt aléatoire peut être décrit en plusieurs étapes :

(i) **Sélection aléatoire des échantillons :** À partir de l'ensemble de données d'entraînement, des échantillons sont sélectionnés aléatoirement avec remplacement pour construire chaque arbre (ce qui est connu sous le nom de bagging).

(ii) **Construction d'arbres de décision :** Pour chaque échantillon sélectionné, un arbre de décision est construit en choisissant de manière récursive les divisions qui minimisent l'erreur quadratique moyenne ou une autre mesure de l'erreur.

(iii) **Entraînement indépendant des arbres :** Chaque arbre est formé indépendamment des autres, ce qui permet à la forêt d'explorer différentes parties de l'espace des caractéristiques.

(iv) **Prédiction :** Lorsqu'il s'agit de faire des prédictions, chaque arbre de la forêt donne sa propre prédiction, et la prédiction finale du modèle de régresseur par forêt aléatoire est donnée par la moyenne des prédictions individuelles, c'est-à-dire :

$$\hat{y}_{\text{ensemble}} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i \tag{3}$$

où $N$ est le nombre d'arbres dans la forêt, et $\hat{y}_i$ est la prédiction de l'arbre $i$.

Le modèle de régresseur par forêt aléatoire bénéficie de la diversité des arbres qui le composent, ce qui le rend plus robuste face au surajustement et améliore sa capacité à généraliser sur de nouveaux exemples.

En pratique, nous avons utilisé le module sklearn.ensemble.RandomForestClassifier

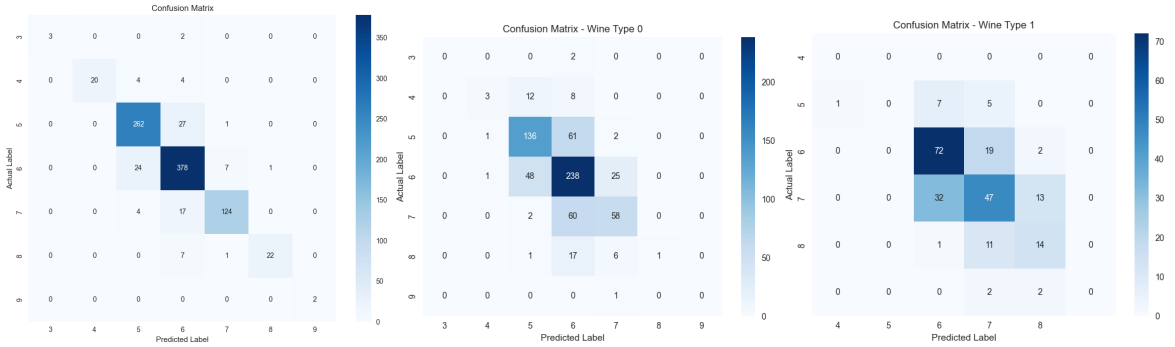
Nous avons testé l'entraînement d'un modèle sur le jeu de données complet ainsi qu'en séparant le jeu de données suivant le type de vin. Les resultats suivant sont obtenus :

— jeu de donnée complet : R2= 0.34

— vin rouge : R2= 0.39

— vin blanc : R2= 0.40

On obtient egalement les matrices de confusion suivantes.




(a) Dataset complet (b) Matrice de confusion type 0 (c) Matrice de confusion type 1

Les classes 5 et 6 sont les mieux prédites pour le type 1 et les classes 6 et 7 pour le type 0. L'optimisation des hyperparamètres de la forêt aléatoire n'a pas été très fructueuse, certainement à cause de l'over-fitting de cette méthode.

### 1.2.2 Regression Linéaire.

La régression linéaire est un modèle statistique qui cherche à établir une relation linéaire entre une variable dépendante $y$ et une ou plusieurs variables indépendantes $x$. Le modèle de régression linéaire simple s'exprime comme suit :

$$y = \beta_0 + \beta_1 x + \varepsilon$$

où :

— $y$ est la variable dépendante,

— $x$ est la variable indépendante,

— $\beta_0$ est l'ordonnée à l'origine (intercept),

— $\beta_1$ est la pente de la ligne de régression,

— $\varepsilon$ est le terme d'erreur qui représente les écarts entre les observations réelles et les valeurs prédites.

L'objectif de la régression linéaire est de trouver les valeurs des coefficients $\beta_0$ et $\beta_1$ qui minimisent la somme des carrés des résidus, c'est-à-dire la somme des carrés des différences entre les valeurs observées $y_i$ et les valeurs prédites $\hat{y}_i$ :

$$\text{minimiser } \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Une fois les coefficients estimés, le modèle peut être utilisé pour faire des prédictions sur de nouvelles données en utilisant l'équation de régression.

Il est également possible d'étendre le modèle à la régression linéaire multiple, où plusieurs variables indépendantes sont prises en compte. Dans ce cas, l'équation devient :

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p + \varepsilon$$

où $x_1, x_2, \ldots, x_p$ sont les variables indépendantes et $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ sont les coefficients associés.

Nous avons testé l'entraînement d'un modèle sur le jeu de données complet ainsi qu'en séparant le jeu de données suivant le type de vin. Les resultats suivant sont obtenus :

— jeu de donnée complet : R2= 0.26

— vin rouge : R2= 0.34

— vin blanc : R2= 0.40

### 1.2.3 Gradient boosting

Le Gradient Boosting est une technique d'apprentissage ensembliste efficace pour modéliser des relations complexes dans les données. Nous décrivons ci-dessous le processus d'entraînement d'un modèle de régression par Gradient Boosting.

#### Initialisation de l'Ensemble

Le processus débute par l'initialisation de l'ensemble de modèles, noté $F_0(x)$, qui représente la prédiction initiale. Cette prédiction peut être simplement la moyenne de la variable cible pour la régression.

#### Entraînement Itératif des Arbres

Pour chaque itération ($m = 1$ à $M$, où $M$ est le nombre total d'arbres dans l'ensemble), le modèle procède comme suit :

### Calcul des Résidus

Les résidus, $r_{im}$, sont calculés pour chaque observation $i$, représentant la différence entre la valeur réelle de la variable cible $y_i$ et la prédiction actuelle $F_{m-1}(x_i)$.

$$r_{im} = y_i - F_{m-1}(x_i) \tag{4}$$

### Entraînement d'un Modèle Faible (Arbre)

Un modèle faible, généralement un arbre de décision peu profond, est entraîné sur les résidus. Le modèle résultant est noté $h_m(x)$.

### Mise à Jour de l'Ensemble

L'ensemble global est mis à jour en ajoutant une contribution pondérée du nouvel arbre, contrôlée par le taux d'apprentissage $\eta$.

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x) \tag{5}$$

## Prédiction Finale

La prédiction finale de l'ensemble de Gradient Boosting est donnée par $F_M(x)$, où $M$ est le nombre total d'arbres.

## Paramètres du Modèle

Les hyperparamètres importants à ajuster lors de l'utilisation du Gradient Boosting incluent le taux d'apprentissage $\eta$ et le nombre total d'arbres $n$ et leur profondeur. Ces paramètres jouent un rôle crucial dans la convergence du modèle et doivent être soigneusement sélectionnés.

Deux principales librairies existent pour le gradient boosting : XGBoost et sklearn.ensemble.GradientBoostingClassifier

Nous avons testé les deux, étant donné que le XGBoost permettait d'optimiser les hyperparamètre avec une complexité computationnelle moindre et des performances meilleurs nous présentons ici cette méthode.
Nous avons optimisé les hyperparametres en utilisant la cross validation.
Nous avons testé l'entraînement d'un modèle sur le jeu de données complet ainsi qu'en séparant le jeu de données suivant le type de vin. Les resultats suivant sont obtenus :

- jeu de donnée complet : R2= 0.42
- vin rouge : R2= 0.52
- vin blanc : R2= 0.42

### 1.2.4 Reduire le surapprentissage.

Face aux performance observées sur Kaggle, nous avons essayé de réduie le sur-apprentissage. Toutefois les courbes de validation (dont un exemple pour le nombre d'estimateurs est affiché en 10 étaient plutot fixes, remettant en question l'utilité du fine-tuning.

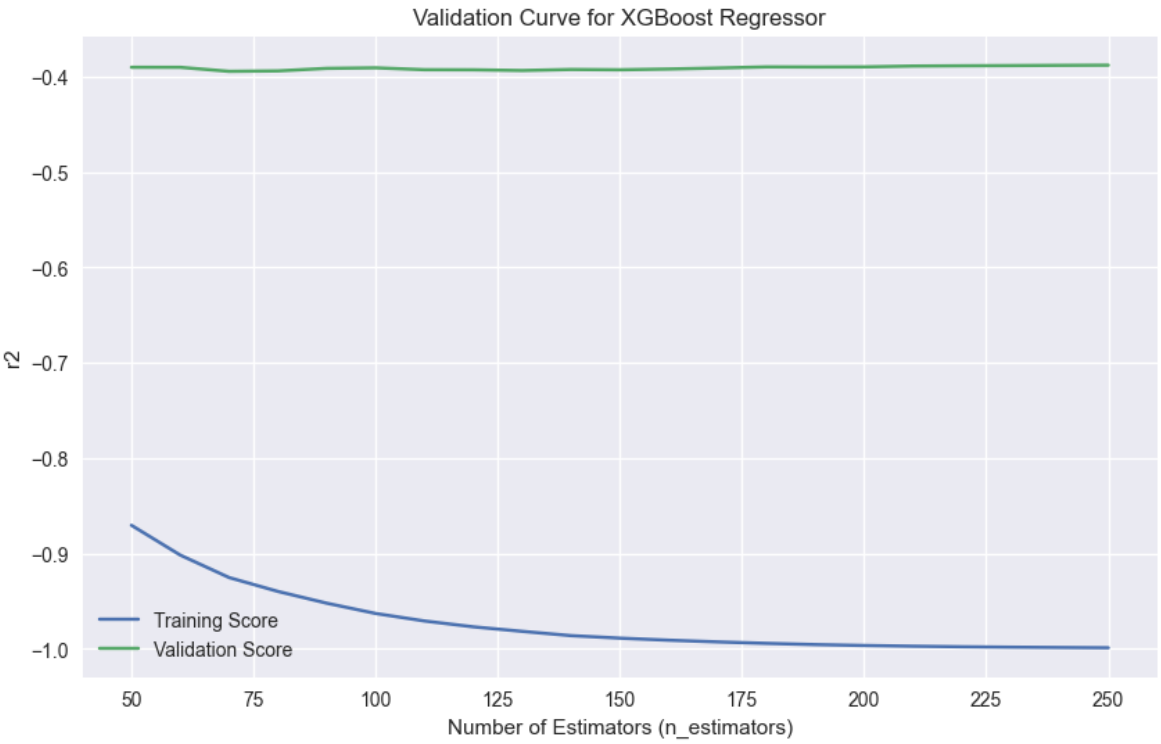



FIGURE 10 – Courbe de validation pour le nombre d'estimateurs.

Nous avons ensuite mis en place une démarche d'early-stopping. L'early stopping est une technique d'entraînement fréquemment utilisée dans les modèles d'apprentissage pour prévenir le surajustement et optimiser l'efficacité du processus d'apprentissage. Cette approche implique de surveiller une métrique de performance, telle que l'erreur sur l'ensemble de validation ou la perte du modèle, pendant l'entraînement. L'arrêt de l'entraînement intervient dès que la métrique cesse de s'améliorer ou commence à se dégrader, indiquant un possible surajustement.
Les avantages de l'early stopping incluent la prévention du surajustement, améliorant ainsi la généralisation du modèle sur de nouvelles données, et la réduction du temps d'entraînement en évitant des itérations inutiles. Cette technique peut être intégrée à divers algorithmes d'apprentissage itératifs, tels que les réseaux de neurones et les modèles de gradient boosting, en se basant sur des critères tels que le nombre d'itérations ou le nombre d'arbres entraînés. Une utilisation judicieuse de l'early stopping contribue à renforcer la robustesse des modèles tout en garantissant une meilleure généralisation.

## 1.3 Conclusion

En conclusion, cette étude visait à évaluer la qualité des vins en attribuant des notes comprises entre 1 et 10. L'analyse exploratoire des données a permis d'identifier des variables importantes et de comprendre la distribution des caractéristiques en fonction du type de vin. L'utilisation de techniques telles que l'analyse en composantes principales a également offert des perspectives intéressantes sur la structure des données.

Dans la phase d'apprentissage supervisé, trois modèles ont été explorés : la forêt aléatoire, la régression linéaire, et le gradient boosting. Les résultats montrent des performances variables en fonction du type de vin, avec des coefficients de détermination ($R^2$) variant entre 0.26 et 0.52. L'analyse des matrices de confusion a permis d'identifier les classes mieux prédites par chaque modèle.

L'optimisation des hyperparamètres, en particulier pour le modèle de gradient boosting, a été réalisée pour améliorer les performances. De plus, des efforts ont été déployés pour réduire le surajustement, notamment avec l'implémentation de l'early stopping.

En fin de compte, bien que les résultats soient encourageants, il reste des défis à relever pour améliorer davantage la précision des modèles. L'exploration de nouvelles techniques d'ensemble, l'ajustement plus fin des hyperparamètres, et l'augmentation de la taille du jeu de données pourraient être des pistes intéressantes pour de futures améliorations.

# 2 Classification d'objets célestes

Dans cette partie nous allons nous concentrer sur la classification d'objets célestes, à partir de 9 caractéristiques, le but est de prédire si un objet est une galaxie (label 0), une étoile (label 1) ou un quasar (label 2).

## 2.1 Analyse exploratoire des données

Dans un premier temps, nous allons décrire qualitativement les variables explicatives. Outre le `label` que nous cherchons à prédire, chaque objet comporte 9 variables explicatives (features), le premier étant simplement un ID (`obj_ID`) n'apportant a priori pas d'information. La position est décrite dans l'espace à partir de deux angles, `alpha` et `delta`. La spectre de luminosité décrivant l'astre est réduit à 5 fréquences, l'ultraviolet (`u`), le vert (`g`), le rouge (`r`), l'infrarouge proche (`i`), l'infrarouge (`z`). La dernière feature est le `redshift` permettant de décrire la vitesse de l'astre vis-à-vis de l'observateur.

Les galaxies étant constitué d'étoiles, on s'attend à une luminosité plus importante pour ces dernières vis à vis des étoiles. Les quasar étant des trous noirs, la luminosité transmise dépend des astres voisins ce qui ne permet pas de formuler d'hypothèse supplémentaire. Les étoiles observables ont plus de chance d'être dans le plan de la voie lactée et les galaxies à l'extérieur, il faudra vérifier si cela se confirme à travers l'analyse des données.

Pour travailler sur ce sujet, on possède deux jeux de données. Le premier s'appelle train et contient 52295 éléments et est labelisé, le second s'appelle test et possède 25758 éléments mais ne possèdent pas de labels et servira à l'évaluation de la prédiction sur kaggle. Les jeux de données sont tous complets et ne présentent pas de valeurs absentes. Nous allons faire une analyse exploratoire du jeu de données train et partir du postulat que nos deux jeux de données sont homogènes entre eux.

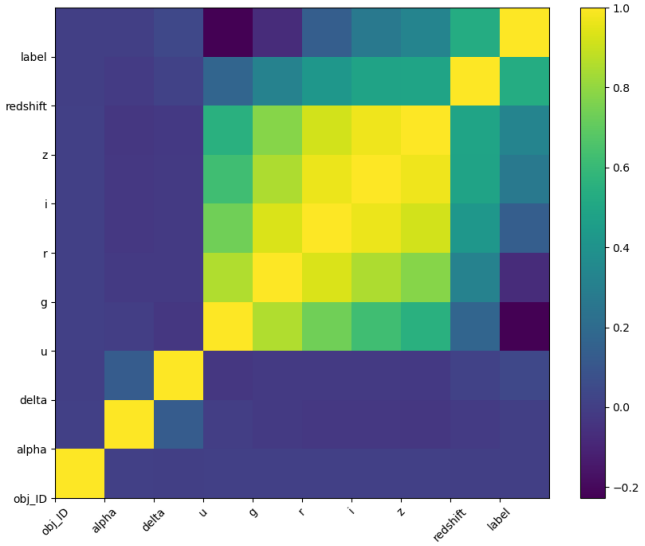



Figure 11

Dans un premier temps nous allons regarder la matrice de corrélation pour chaque couple de features $(x, y)$ donné par l'équation 1.

Avec $\bar{x}$ et $\bar{y}$ les moyennes empiriques de $x$ et $y$. On utilise le module `pandas`, qui possède une implémentation pour obtenir une matrice de corrélation qu'on affiche graphiquement avec le module `matplotlib.pyplot` comme sur la figure 11.

Cette visualisation permet un support graphique pour découvrir les régions d'intérêt avec lesquelles on peut obtenir les valeurs numériques probantes. Ainsi on remarque une légère corrélation entre les deux angles `alpha` et `delta` de l'ordre de 0.12. Les coefficients les plus élevés sont néanmoins entre les cinq bandes lumineuses, allant de 0.62 à 0.96 ce qui semble indiquer qu'en règle général les objets n'émettent pas dans une seule bande de fréquence et que leur intensité globale va être un élément discriminant pour les classer. Le `redshift` est corrélé avec les bandes de lumière et d'autant plus avec celle de basse fréquence ce qui est cohérent, vu que ces dernières vont être d'autant plus affecté par le phénomène de redshift. Le `label` a un coefficient de corrélation de 0.52 avec le `redshift` ce qui peut s'expliquer par le fait que les quasars et les galaxies sont des objets plus lointain que les étoiles et donc s'éloigne plus vite de nous ce qui augmente leur `redshift`. On remarque en outre un coefficient de corrélation proche de 0 entre les deux angles ce qui contredit le point énoncé plus haut.

A l'aide du même module, on affiche les boîtes à moustache pour chaque features en fonction du label et on obtient la figure 12.

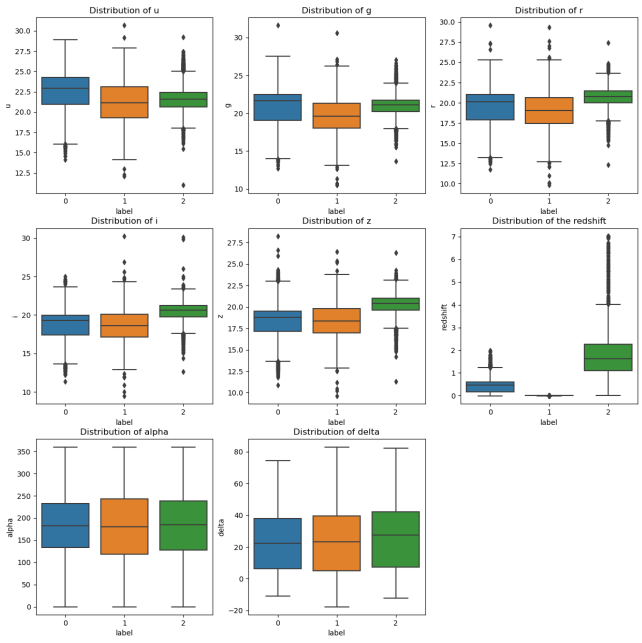
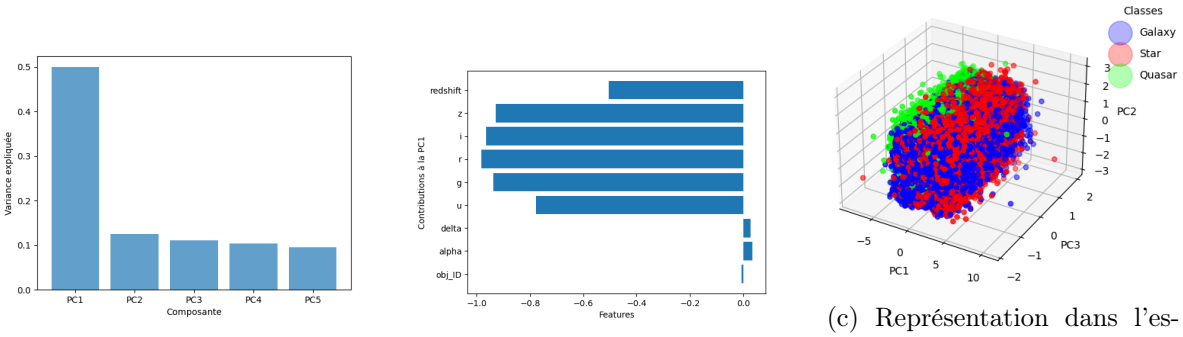



FIGURE 12

On retrouve les mêmes résultats que précédemment, les angles ne discriminent absolument pas chaque catégorie. Les différentes intensités lumineuses présentent des tendances mais pas spécialement prononcé tandis que le `redshift` des étoiles est proche de 0, tandis que les valeurs

extrêmes sont atteintes pour les quasars.

Pour finir cette analyse, on se propose de faire une analyse en composante principalement pour voir si des tendances se dessinent et confirmer les importances de chaque variable.




(a) Variance expliquée par chaque composante

(b) Contribution à la première composante principale

(c) Représentation dans l'espace des 3 premières composantes

FIGURE 13 – Analyse en composantes principale

Après avoir normalisé les données, on utilise le module PCA du module `sklearn`, et on obtient les figures 13. Sur la première composante principale, qui explique environ 50% de la variance, on remarque que les variables contribuant le plus sont celles de l'intensité lumineuse, le redshift contribuant légèrement moins qu'escompté tandis que les angles et l'ID ne contribuent presque pas.

En représentant chaque objet dans l'espace des trois premières composantes principales (fig 13c), on remarque qu'il n'y a pas de cluster distinct bien que les quasars semblent s'isoler d'un côté. Nous allons donc garder chacune des variables du à leur faible nombre.

## 2.2 Apprentissage supervisé

Dans cette section, nous allons chercher à mettre en place différentes techniques d'apprentissage pour essayer de prédire au mieux la classification de chaque objet. Dans cette partie, sauf contre indication, nous allons séparer le jeu de données train en deux partie, une pour l'entraînement (70%) et une pour l'évaluation (30%).

Dans un premier temps, nous allons essayé tout un panel de méthode pour ensuite essayer d'ajuster les hyper-paramètres de celle(s) présentant les meilleurs résultats. Pour évaluer les modèles, nous allons nous baser sur la précision, définie comme le rapport des bonnes prédictions et le nombre d'éléments, ainsi que la matrice de confusion, nous permettant de regarder plus précisément les types d'erreurs du modèle.

### 2.2.1 Analyse linéaire discriminante

Le premier modèle que nous avons essayé est l'analyse linéaire discriminante, qui se base sur la règle bayésienne :

$$P(Y = y_k \mid X) = \frac{P(Y = y_k) \times P(X \mid Y = y_k)}{\sum_{i=1}^{3} P(Y = y_i) \times P(X \mid Y = y_i)} \tag{6}$$

Pour cela, on utilise le module LinearDiscriminantAnalysis de `sklearn` qui permet de résoudre ce problème en utilisant une décomposition en valeurs singulière. On obtient une précision de 84 % ainsi que la matrice de confusion suivante :

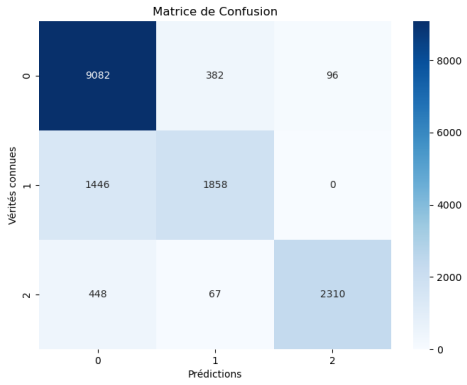



FIGURE 14

On remarque que quand le modèle prédit des quasars pour un objet, 96 % des prédictions sont bonnes, ce qui peut être intéressant dans le cas d'un modèle d'aggrégation d'expert.

### 2.2.2 Machine à vecteurs de support

Comme vu en cours, le concept du SVM s'appuie sur la minimisation du problème suivant :

$$\min_{w,b,\xi} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \tag{7}$$

Sous les contraintes $y_i(w \cdot x_i + b) \geq 1 - \xi_i, et \xi_i \geq 0 \quad i = 1, 2, \ldots, N.$

En plus de ça, on peut appliquer un noyau à notre espace de départ pour essayer de mieux s'adapter au données. On se retrouve donc avec 2 hyperparamètres, $C$ et le choix du noyau. Pour trouver le bon jeu de paramètre, on se propose de faire une cross-validation avec 3 dossiers, le choix de ce faible nombre est le temps de calcul de nos machines.

On se propose d'essayer les noyaux :

— Linéaire : $K(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{x}_2$

— Radial basis function (RBF) : $K(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \cdot \|\mathbf{x}_1 - \mathbf{x}_2\|^2\right)$

— Sigmoid : $K(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma \cdot \mathbf{x}_1^\top \mathbf{x}_2 + r)$

Les (nouveaux) hyperparamètres des fonction RBF et Sigmoid sont donnés par $\gamma = 1/(n_features * X.var())$ et $r = 0$ car nous n'avions encore une fois pas la puissance computationnelle requise pour entraîner tout les modèle. L'autre hyperparamètre $C$ qui régit la souplesse de la marge prendra pour valeurs des puissances de 10, allant de 0.1 à 100. Cela nous donne 12 ($3 \times 4$) modèles à entrainer. Pour cela on utilise les modules `cross-validation` de sklearn ainsi que le module `SVM` de sklearn. Nous avions essayé des noyaux polynomiaux ainsi que faire varier $\gamma$ mais le nombre de modèle explosait malheureusement.

Pour chaque modèle, on l'entraîne sur 2 dossiers et on l'évalue sur le 3ème et on calcule l'accuracy et ce, pour chaque dossier en tant que test. A partir de l'accuracy moyenne, on détermine le meilleur jeu de paramètre et on obtient le noyau linéaire et $C = 1$.

Pour l'évaluation finale sur le jeu de données test on obtient une accuracy de 79% ce qui est moins bon que l'analyse linéaire discriminante 2.2.1 ce qui peut peut s'expliquer par le fait que nous n'avons pas pu tester beaucoup de paramètres. Même sur la figure 15, la matrice de corrélation n'est meilleur en aucun point nous ne conserverons pas ce modèle

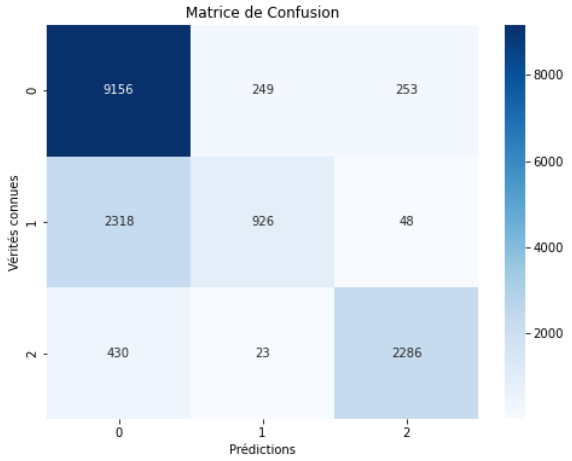



FIGURE 15

### 2.2.3 Gradient Boosting

Le Gradient Boosting est une technique d'ensemble d'apprentissage automatique largement utilisée pour améliorer la précision prédictive des modèles. Il s'agit d'une méthode itérative qui combine plusieurs modèles simples, dans notre cas des arbres de décisions. Le but du gradient boosting est d'entrainer ces modèles les uns à la suite des autres en entraînant chacun sur l'erreur des précédant.

$$f(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

Ici, le but est d'ajuster les $\alpha_m$ et $h_m$, pour obtenir un modèle prédictif. On se retrouve donc avec un jeu d'hyperparamètres, $M$ le nombre total d'estimateurs, le learning rate $\eta$ de

la fonction qui a chaque étape va influer sur le gradient de l'erreur, la fonction de perte pour évaluer le modèle, ainsi que la profondeur maximal de chaque estimateur.

Comme dans la partie précédante, on va procéder par cross validation avec :

— Pour la fonction d'erreur : la logloss (qui se base sur la déviance) ou la loss exponentiel (qui se base sur l'Adaboost)
— Le learning rate : 0.001, 0.01, 0.1
— Le nombre d'estimateur : 100, 200
— La profondeur maximale : 3, 5

On utilise le module `GradientBoostingClassifier` du module scikit-learn, et on obtient comme meilleurs paramètres 0.1 pour le learning rate, la logloss pour la fonction de perte, une profondeur maximale de 5 et 200 estimateurs.

Cette fois-ci, on obtient une accuracy de 0.97 ce qui est une nette amélioration vis-à-vis des sections d'avant et la matrice de confusion suivante :

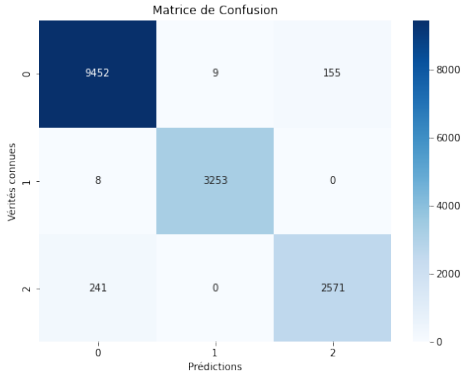



FIGURE 16

Comme on l'avait remarqué dans l'analyse exploratoire, les étoiles se démarquent des quasars et des galaxies, et le prédicteur obtenu ne se trompe que 0,3 % quand il prédit la classe étoile et a identifier 99,7 % des étoiles, cependant les confusions se font sur les deux autres classes.

### 2.2.4 Forêt aléatoire

La forêt aléatoire est similaire au boosting dans le sens où on l'entraîne plusieurs arbres aléatoires sur le jeu de données mais cette fois par sur les erreurs des arbres précédent. On essaye cette méthode en s'attendant à des résultats assez similaires mais comme les résultats sont déjà très bon (97% d'accuracy) il n'est pas inutile de gagner un demi pourcent.

Comme précédemment, on se retrouve avec plusieurs hyperparamètres, le premier étant le nombre d'arbre, puis la fonction de perte et pour finir la profondeur maximale des arbres

composant la forêt. Pour les fonction de perte, le but sera de minimiser l'un des trois critères suivant :

Le critère de Gini mesure la pureté d'un ensemble en évaluant la probabilité qu'un élément choisi au hasard appartienne à la mauvaise classe. Il est défini comme suit pour un ensemble $S$ contenant $K$ classes :

$$Gini(S) = 1 - \sum_{i=1}^{K} p_i^2$$

où $p_i$ est la proportion d'éléments de la classe $i$ dans l'ensemble $S$.

L'entropie est une mesure d'incertitude ou de désordre dans un ensemble. Plus l'entropie est élevée, plus l'ensemble est mélangé. Pour un ensemble $S$ avec $K$ classes, l'entropie $H(S)$ est définie comme :

$$H(S) = -\sum_{i=1}^{K} p_i \log_2(p_i)$$

où $p_i$ est la proportion d'éléments de la classe $i$ dans l'ensemble $S$.

La perte logarithmique (log loss) est couramment utilisée pour mesurer la performance d'un classificateur probabiliste. Pour un ensemble de données avec $N$ exemples, $y_i$ étant la véritable classe et $p_i$ la probabilité prédite d'appartenance à la classe $y_i$, la log loss est définie comme :

$$LogLoss = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} y_{ij} \log(p_{ij})$$

où $y_{ij}$ est une variable indicatrice qui vaut 1 si l'exemple $i$ appartient à la classe $j$ et 0 sinon.

Pour la nombre d'estimateur, on le fera varier entre 100, 250 et 500 et la profondeur maximale quant à elle sera soit 5, 10 ou 20.

En utilisant le module `RandomForestClassifier` de scikit-learn on obtient que le meilleur jeu d'hyperparamètre est celui utilisant la log_loss, une profondeur maximale de 20 ainsi que 500 estimateurs. On obtient une accuracy de 97,6% ce qui est similaire à celle obtenue plus haut avec une matrice de confusion qui est somme toute assez similaire à celle du gradient boosting

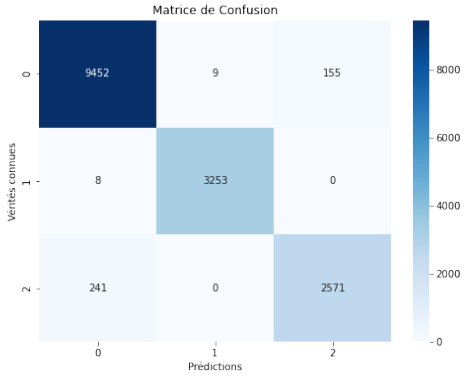



FIGURE 17

## 2.3 Conclusion et soumission sur kaggle

Pour modéliser nos données, nous avons donc essayé 4 modèles différents, l'analyse linéaire disciminante, le SVM, le gradient boosting ainsi que les fôrets aléatoires. Pour les trois derniers modèles, nous avons fait une cross validation pour essayer de déterminer le meilleur jeu d'hyperparamètres, bien que nous ayons été limité par la puissance de nos ordinateurs, particulièrement pour le SVM. Avoir un ordinateur plus puissant nous aurait permis de tester plus de noyau avec un maillage plus fin.

Finalement, nous avons obtenus de très bons résultats sur le gradient boosting et et les forêt aléatoires avec une accuracy de plus de 97%, ce sont donc les modèles que nous avons décider de garder. En publiant le résultat sur kaggle, nous obtenons une accuracy du même ordre de grandeur ce qui est rassurant car nous n'avons pas fait d'overfitting. Le modèle possèdant les meilleurs résultats sur kaggle correspond à une forêt aléatoire sur 100 arbres que nous avions essayé en début de période, il se rapproche des autres mais a une meilleure accuracy de l'autre du centième ce qui n'est pas significatif.