Test Technique

Rédigé par : DRIOUCHE Adnane

I. Introduction :

Cet exercice a pour objectif d'évaluer mes compétences en data science. Pour l'attaquer, j'ai développé 2 classes dans le fichier « Quantmetry.py » :

- ***La classe PreProcessing :*** contient tous les méthodes nécessaires pour prétraiter les données (e.g. traitement des données manquantes, oneHotEncoding …)
- ***La classe Stats :*** contient tout ce qui sera nécessaire pour l'étude de corrélation, la dépendance et la visualisation des distributions.

Pour les parties prétraitement, analyse et test je les ai fait dans jupyter notebook «main.ipynb».

Pour vous faciliter la correction, Je vous présente l'architecture du code (figure 1). Les cases vertes signifient que c'est un dossier et les jaunes sont des fichiers.

J'ai hébergé aussi mon notebook sous format html sur Github Page.

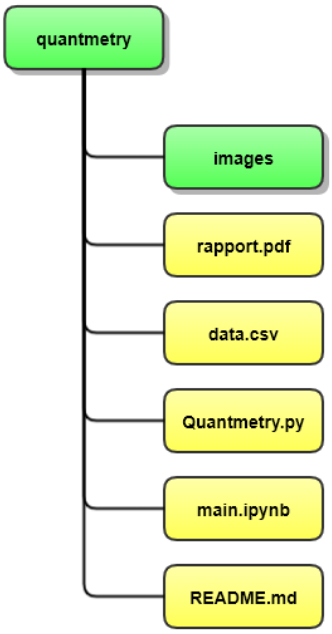

Figure 1 Chemain des fichiers

II. Statistiques descriptives :

i. Description des données :

Je récapitule les informations relatives à nos données dans la figure suivante :

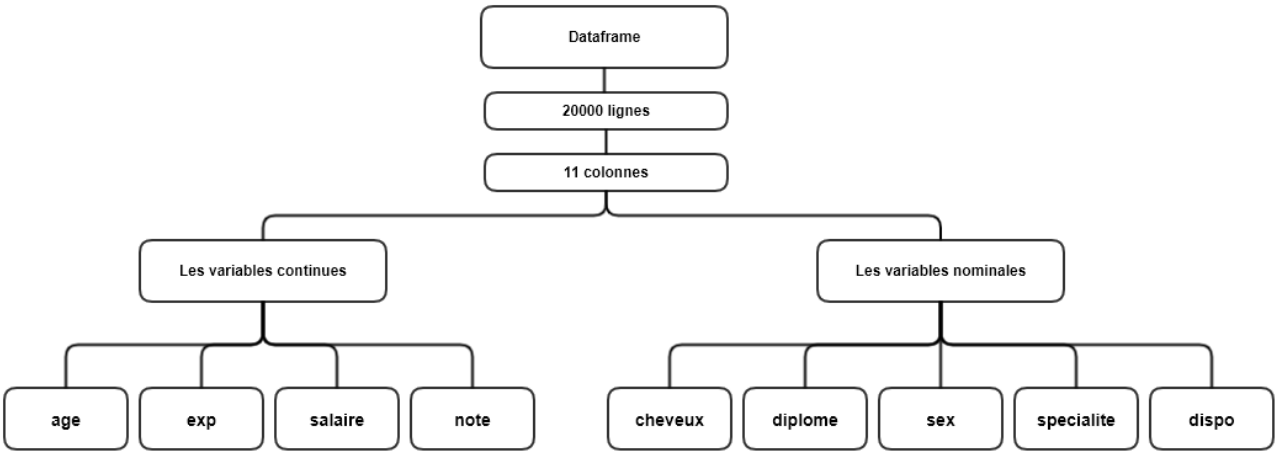

Figure 2 roadmap de nos données

Notre dataframe contient :

- ***20000 lignes :*** y compris des lignes nulles (je les ai pris en considération pendant le prétraitement).
- ***11 colonnes :*** 4 variables continues et 5 variables nominales. Donc, nos données sont bien hétérogènes.

ii. Les variables pertinentes :

Pour le choix des variables pertinentes, j'ai adopté l'approche filter et j'ai utilisé précisément ExtraTreesClassifier. J'ai obtenu les résultats suivants :

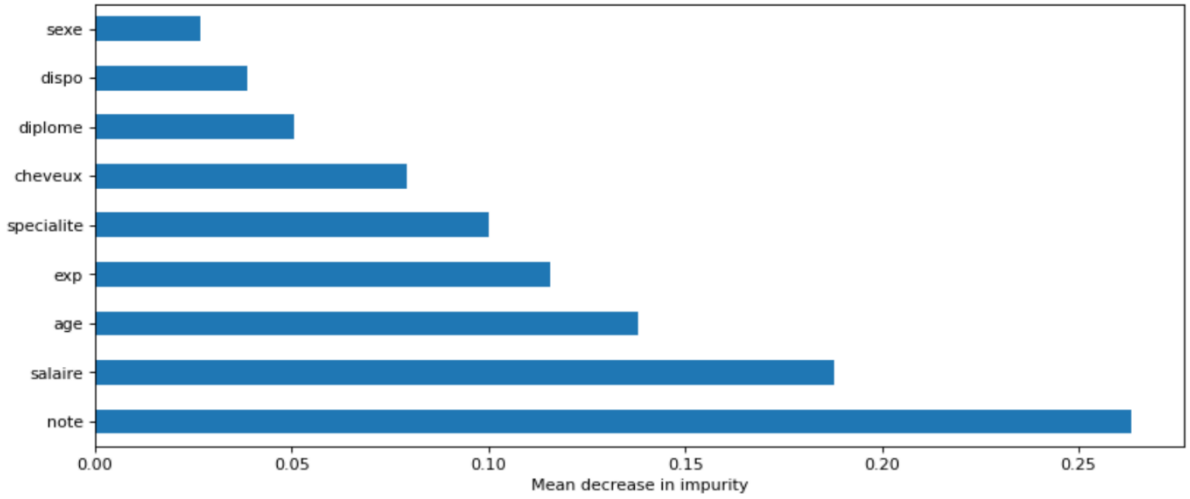

Figure 3 Classement des variables selon leur importances

Par la suite, j'ai pris les variables qui ont un « Mean decrease in impurity » supérieur à 0.1

iii. Distribution des variables pertinentes :

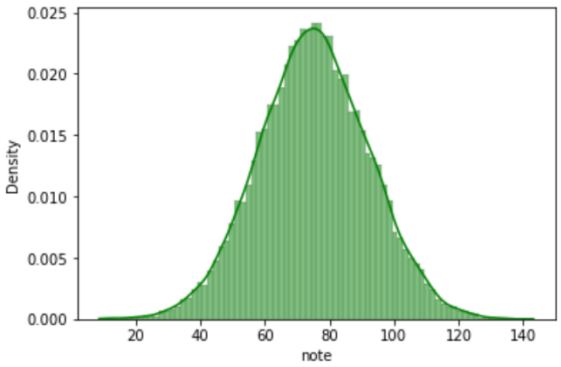

Figure 5 Distribution des notes

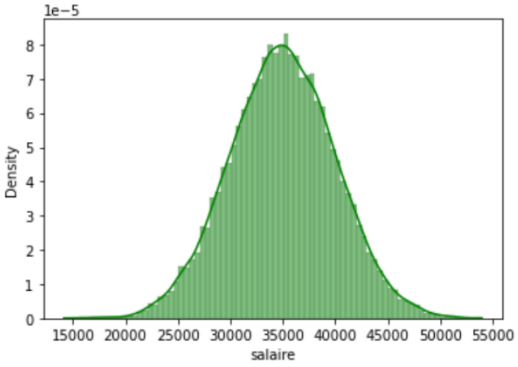

Figure 4 Distribution des salaires

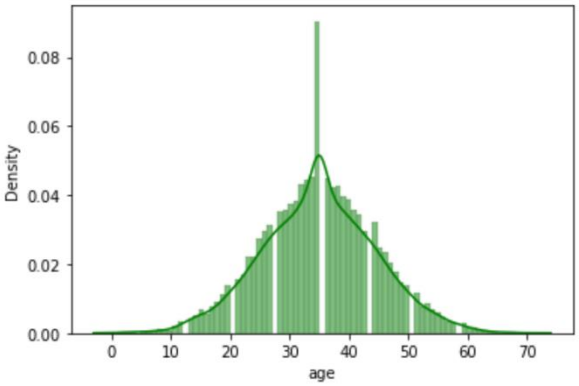

Figure 6 Distribution des ages

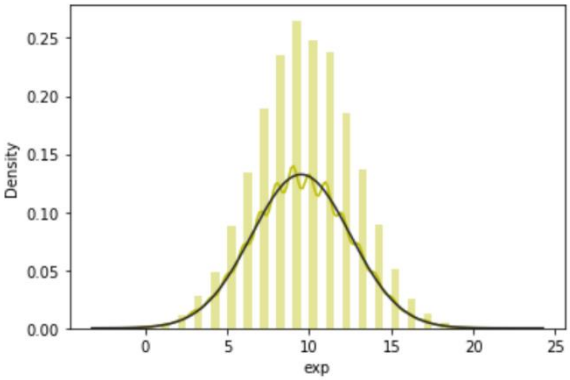

Figure 7 Distribution des expériences

On remarque que nos variables suivent bien la loi normale !

iv. La corrélation entre les variables :

Par la suite, j'ai étudié la corrélation entre toutes les variables.

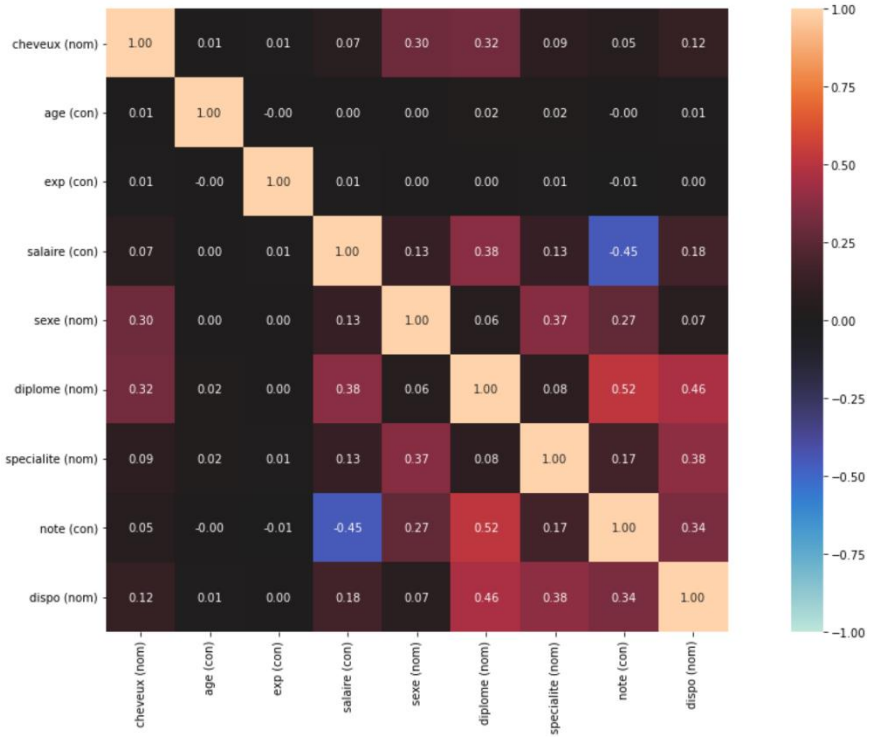

Figure 8 Matrice de corrélation entre les variables

Pour vérifier s'il y a une dépendance statistiquement significative entre les variables données. J'ai supposé au premier temps notre hypothèse nulle avec une valeur de p <0.05.

Soit H0 l'hypothèse nulle utilisée dans le test du chi-deux.

a. Spécialité Vs Sexe :

- ***H0 :*** sexe et spécialité ne sont pas corrélés.
- ***Résultat :*** J'ai obtenu un p presque nulle, ce qui équivaut à dire il y a plus de preuves en faveur de l'hypothèse alternative. Ce qui signifie que les variables sont corrélées entre elles.

b. Cheveux VS Salaire :

- ***H0 :*** cheveux et salaire ne sont pas corrélés entre eux.
- ***Résultat :*** J'ai obtenu un p = 0.13 > 0.05. Par conséquent, H0 sera acceptée, ce qui signifie que les variables ne sont pas corrélées.

c. Expérience VS Note

- ***H0 :*** expérience et note ne sont pas corrélées entre elles.
- ***Résultat :*** J'ai eu un p = 0.13 > 0.05. Par conséquent, H0 sera acceptée, ce qui signifie que les variables ne sont pas corrélées.

III. Machine Learning :

i. Choix de modèle :

Nous sommes en face d'un problème de prédiction du succès ou d'échec d'un candidat X. Par la suite, on a le choix entre l'un des algorithmes d'apprentissage supervisé (plus précisément classification) (figure 9) .

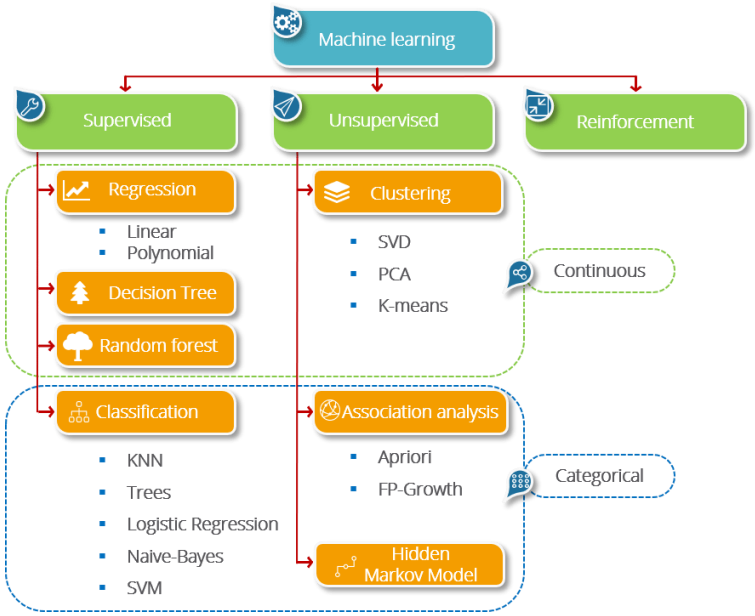

Figure 9 Roadmap d'un projet ML

Pour ce genre de cas, je privilège ***RandomForest***, ça me donne toujours des bons résultats, mais ça n'empêche pas d'essayer SVM et KNN. Pour GNB, je ne le conseille pas dans cette cas, car il suppose que les variables sont indépendants sauf que ce n'est pas le cas.

ii. Les paramètres du modèle :

En général, plus il y a d'arbres dans la forêt, plus la forêt est robuste. De la même manière, pour le classifieur ***RandomForest***, plus le nombre d'arbres dans la forêt est élevé, plus les résultats de précision sont élevés.

Le paramètre qui contrôle le nombre des arbres est :

- ***N_estimators :*** Nombre d'arbres dans RF.

Pour éviter overfitting, on doit mieux choisir :

- ***Max_depth :*** est défini comme le plus long chemin entre le nœud racine et le nœud feuille. Une grande valeur pour cet hyperparamètre va sûrement causer overfitting.

iii. Choix du critère de performance (metric) :

Comme je vous ai montré dans « ***main.ipynb*** », nos données sont déséquilibrées ce qui nous empêche de faire confiance à Accuracy comme critère de performance. En fait, dans notre cas on a juste 12% des candidats sont embauchés c'est-à-dire si le modèle va prédire que tous les candidats sont embauchés, on aura accuracy = 88%. C'est pour cela, que je me suis basé sur d'autres critères : *Precision, Recall et F1 score*.

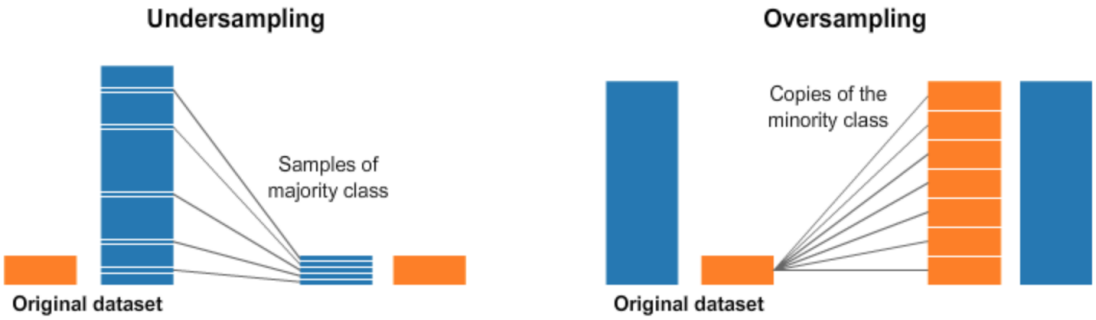

Figure 10 La méthode Undersampling VS Oversampling

Le choix du critère de performance est nécessaire mais pas suffisant pour l'apprentissage de notre modèle. C'est pour cela, j'ai utilisé overSampling (figure 10) pour avoir des données plus ou moins équilibrées. J'ai obtenu les résultats suivants :

	Sans OverSampling	Avec Oversampling
Accuracy	88.33%	97.1%
Precision	65.12%	97.00%
Recall	52.5 %	97.00%
F1-score	50.07%	97.00%

iv. Amélioration du modèle :

Nous pouvons améliorer notre modèle en :

- ***Affinant nos hyperparamètres :***

On peut utiliser GridSearchCV.

- ***Entraînant notre modèle utilisant cross-validation*** :

  ça va nous permettra aussi d'éviter l'overfitting en divisant l'ensemble d'entraînement global en petits morceaux, puis en utilisant chaque morceau pour entraîner le modèle.

- ***Bénéficiant des variables pour créer des nouveaux caractéristiques :***
  Par exemple, on peut utiliser la date pour créer une nouvelle variable ➔ Saison
  Les valeurs uniques de Saison seront : été, hiver, automne et printemps. Comme l'embauche dépend plus ou moins des saisons ça va être bénéfique pour l'apprentissage de notre modèle