ROYLAB
CODE & WEB

# RAPPORT DE PROJET

**SUJET :**

# Mise en place d'une Application Web de Prédiction de l'éligibilité à un Prêt Bancaire

**Réalisé par :**

M. Malick Royce LAYINDE

Période : Juillet 2O24

# RESUME

Dans un monde où la gestion des risques financiers devient de plus en plus complexe, il est essentiel pour les institutions bancaires de pouvoir évaluer avec précision l'éligibilité de leurs clients à des prêts. Face à cette problématique, ce projet propose une solution pour assister les gestionnaires de comptes dans leurs prises de décision.

L'objectif principal de ce projet est de concevoir une plateforme web de prédiction de l'éligibilité des clients à un prêt bancaire. En utilisant des algorithmes de machine learning, la plateforme permet aux banquiers de remplir un formulaire contenant les informations du client, puis d'obtenir une prédiction sur son éligibilité au prêt. En plus de cette fonctionnalité principale, la plateforme offre la possibilité de consulter les informations personnelles du client, ses transactions, son historique de prêts, ainsi que de comparer les performances financières des différents clients à travers des visualisations graphiques.

Ce projet a ainsi permis d'allier des compétences techniques acquises avec les besoins concrets du secteur bancaire, tout en mettant en œuvre des technologies modernes pour la gestion des données et la prise de décision.

# LISTE DES FIGURES

# LISTE DES SIGLES, ACRONYMES ET ABREVIATIONS

**IA :** Intelligence Artificielle

**CSS :** Cascading Style Sheets

**HTML :** HyperText Markup Language

**JS :** JavaScript

**SQL :** Structured Query Language

**RL :** Regression Logistique

**KNN :** K-Nearest Neighbors (Algorithme de Machine Learning)

**MySQL :** My Structured Query Language (Système de gestion de base de données)

**SGBDR :** Système de Gestion de Base de Données Relationnelles

**ML :** Machine Learning (Apprentissage automatique)

**DB :** Base de données (Database)

**ID :** Identifiant

**HC :** Historique Client

# SOMMAIRE

# INTRODUCTION

De nos jours, l'information et la donnée jouent un rôle crucial dans la prise de décision, l'essor des technologies du Big Data et de l'intelligence artificielle a transformé plusieurs secteurs, y compris celui des services financiers. Les institutions bancaires font face à des volumes de données toujours croissants, notamment dans la gestion des clients et l'évaluation des risques financiers. Ainsi, la capacité de tirer parti de ces données pour prédire des résultats tels que l'éligibilité des clients à des prêts devient une nécessité stratégique.

Dans ce contexte, ce projet s'inscrit comme une approche de réponse à la problématique de la prédiction de l'éligibilité des prêts bancaires. Les gestionnaires de comptes et les banquiers doivent non seulement évaluer la capacité de remboursement des clients, mais aussi analyser de manière efficace l'historique financier de chaque client pour prendre des décisions éclairées. Le projet propose une solution qui permet de prédire l'éligibilité d'un client à un prêt, tout en offrant des outils de visualisation et d'analyse des données clients.

L'objectif principal de ce projet est de développer une plateforme web qui aide les gestionnaires de comptes à évaluer de manière rapide et précise l'éligibilité des clients à des prêts, à travers un modèle prédictif basé sur le ML. En outre, cette plateforme fournit des fonctionnalités complémentaires telles que la visualisation des transactions, des statistiques sur les prêts et la comparaison entre clients, afin d'améliorer l'expérience utilisateur et d'optimiser le processus de gestion des comptes.

Ce rapport est structuré en trois grandes parties :

- **Cadre théorique et méthodologique**: Cette première partie examine le cadre théorique et méthodologique, en présentant les concepts clés, les approches existantes en matière de prédiction de prêts et les technologies pertinentes utilisées dans ce domaine.

- **Cadre Pratique** : Cette section détaille le processus de développement du modèle de prédiction, la conception et le développement de l'interface utilisateur, ainsi que l'intégration des données.
- **Cadre Analytique** : La dernière partie se concentre sur l'analyse des résultats obtenus, en discutant des performances du modèle, des limites rencontrées et des pistes d'amélioration possibles. Cette section évalue également les besoins futurs pour l'évolution de la plateforme et propose des recommandations pour des travaux ultérieurs.

# 1ere PARTIE : CADRE THEORIQUE ET METHODOLOGIQUE

## 1-1. INTRODUCTION DU CADRE THEORIQUE

Dans le secteur bancaire, l'évaluation de l'éligibilité des prêts est un processus crucial qui permet aux institutions financières de gérer les risques et de prendre des décisions éclairées concernant l'octroi de crédits. Avec l'augmentation des volumes de données et l'évolution des technologies, les banques ont recours à des modèles prédictifs pour améliorer l'efficacité et la précision de cette évaluation. Ce cadre théorique vise à explorer les concepts clés, les technologies utilisées, et les meilleures pratiques en matière de prédiction de l'éligibilité des prêts.

## 1-2. REVUE DE LA LITTERATURE

### Introduction au Machine Learning

Tout commence par l'Intelligence Artificielle. Ce domaine est défini comme : *''l'effort d'automatisation des tâches intellectuelles normalement effectuées par des humains''* [1]. Le ML (apprentissage automatique) quant à lui, est un sous-domaine de l'IA qui permet à un système informatique d'apprendre et d'améliorer ses performances en effectuant des tâches sans être explicitement programmé pour chaque tâche. Cette capacité d'apprentissage repose sur l'utilisation d'algorithmes et de modèles statistiques capables de détecter des motifs et de faire des prédictions ou des décisions basées sur des données.

Le ML se divise principalement en deux grandes catégories : l'apprentissage supervisé et l'apprentissage non supervisé.

- **Apprentissage Supervisé** : Cette approche implique l'utilisation d'un ensemble de données d'entraînement étiqueté, où chaque entrée est associée à une sortie connue. L'objectif est de construire un modèle capable de faire des prédictions ou des classifications sur des données nouvelles, non étiquetées. Les algorithmes d'apprentissage supervisé incluent la régression logistique, les k-plus proches voisins (KNN), et les arbres de décision.

[1] François CHOLLET, (2020) L'apprentissage profond avec python, Les essentiels de l'IA,Saint-Cyr sur Loire Manning publications,p5

- **Apprentissage Non Supervisé** : Contrairement à l'apprentissage supervisé, l'apprentissage non supervisé n'utilise pas de données étiquetées. L'objectif est de découvrir des structures sous-jacentes ou des motifs dans les données. Les techniques courantes incluent le clustering (regroupement).

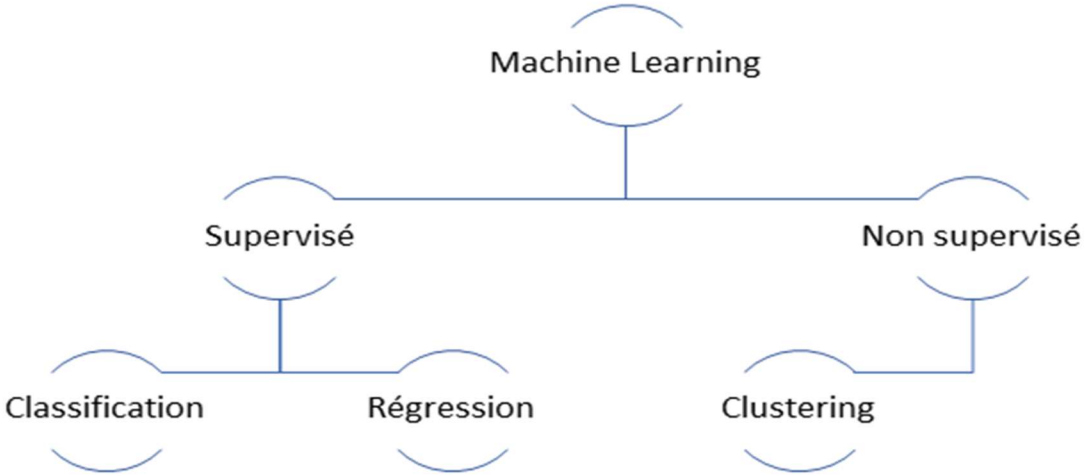



*Figure 1 : Le Machine Learning*

## Régression Logistique et K-Nearest Neighbors (KNN)

- **Régression Logistique** : La régression logistique est un modèle de classification utilisé pour prédire la probabilité qu'une observation appartienne à une classe spécifique. Contrairement à la régression linéaire, qui est utilisée pour les prédictions continues, la régression logistique est adaptée aux problèmes de classification binaire, où les sorties possibles sont souvent codées en 0 ou 1. Elle utilise une fonction logistique (ou sigmoïde) pour modéliser la probabilité que l'événement d'intérêt se produise.

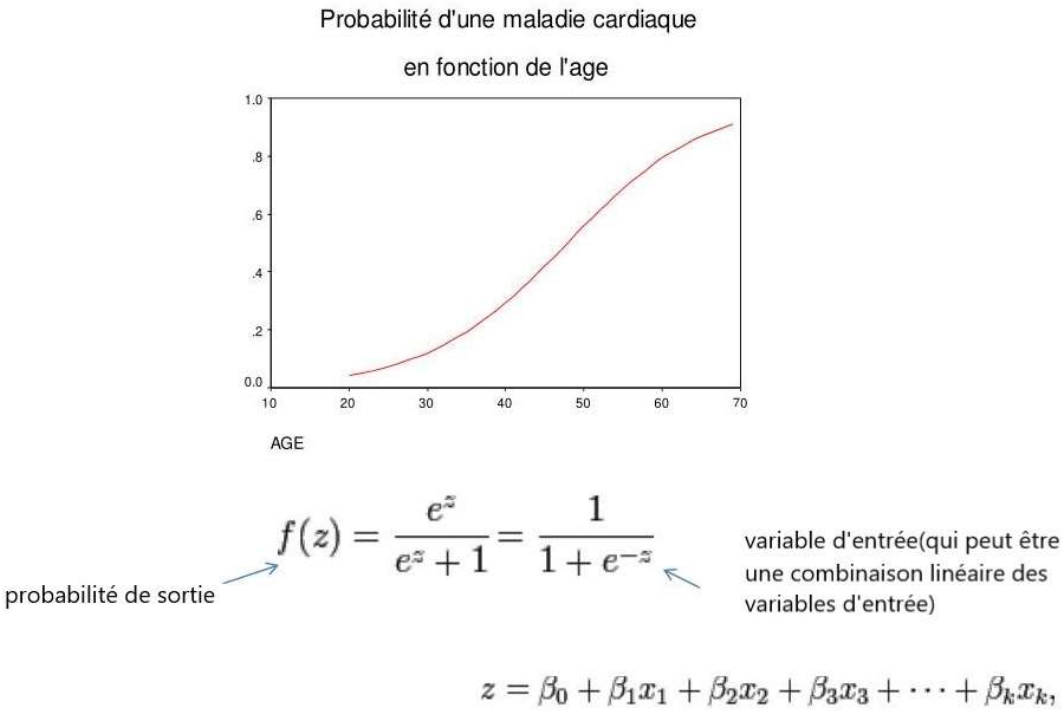



*Figure 2 : Le Modele Logistique*

Le modèle de régression logistique fonctionne donc en prédisant des probabilités pour classer des observations dans l'une des deux catégories possibles. Il commence par une combinaison linéaire des variables explicatives, semblable à la régression linéaire. Cependant, au lieu de produire directement une valeur continue, le modèle applique la fonction sigmoïde (ou logistique) à cette combinaison. La fonction sigmoïde transforme cette valeur en une probabilité comprise entre 0 et 1. Si la probabilité est supérieure à un seuil, généralement 0,5, l'observation est classée dans une catégorie, sinon dans l'autre. Grâce à cette transformation, la régression logistique permet de modéliser des problèmes de classification binaire avec efficacité.

- **K-Nearest Neighbors (KNN)** : KNN est un algorithme de classification basé sur la proximité. Il classe un échantillon en fonction des classes des k voisins les plus proches dans l'espace des caractéristiques. KNN est simple à comprendre et à mettre en œuvre, et il ne nécessite pas de modèle d'apprentissage préalable. Cependant, il peut être coûteux en calcul pour des ensembles de données volumineux, car il doit comparer chaque nouvelle donnée à toutes les autres.

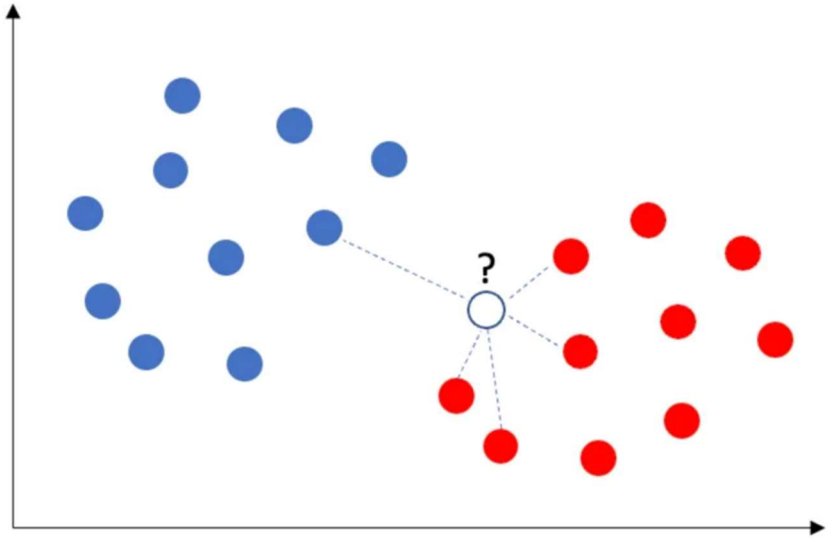



*Figure 3 : Le Modèle KNN*

Ainsi donc, avec le modèle KNN, lorsqu'une nouvelle observation doit être classée, l'algorithme recherche les k observations les plus proches dans l'ensemble d'entraînement en mesurant la distance (généralement la distance euclidienne[2]). Ensuite, il attribue à la nouvelle observation la catégorie qui est la plus fréquente parmi ces voisins

Ces modèles jouent un rôle crucial dans la prédiction des résultats en fonction des données d'entrée et sont au cœur des nombreux systèmes de décision automatisés dans divers domaines, y compris le crédit et les prêts.

## Technologies et Outils Utilisés

Pour la mise en œuvre des modèles prédictifs, ainsi que du reste, divers technologies et outils ont étés utilisées :

**Python** : Langage de programmation largement utilisé pour le ML en raison de sa richesse en bibliothèques telles que scikit-learn, pandas, et NumPy.

**NumPy** : Bibliothèque Python pour le calcul scientifique, offrant des structures de données et des fonctions pour le traitement des tableaux multidimensionnels.

[2] Mesure de la distance entre deux points dans un espace à plusieurs dimensions.

**Seaborn** : Bibliothèque Python basée sur Matplotlib pour la visualisation des données, offrant des graphiques informatifs et esthétiques.

**Flask** : Framework web en Python qui permet de créer des applications web légères et flexibles. Il est souvent utilisé pour développer des interfaces utilisateur et des API pour les modèles de prédiction.

**MySQL** : Système de gestion de bases de données relationnelles qui stocke et gère des données.

**mysql-connector** : Bibliothèque Python pour connecter des applications Python à une base de données MySQL, facilitant l'exécution des requêtes SQL.

**HTML** : Langage de balisage utilisé pour structurer le contenu des pages web.

**CSS** : Langage de feuille de style utilisé pour définir l'apparence et la mise en page des pages web, permettant de créer des interfaces utilisateur attrayantes.

**Chart**.js : Bibliothèque JavaScript pour la création de graphiques interactifs et dynamiques sur le web, permettant une visualisation efficace des données dans l'interface utilisateur.

**Docker** : Plateforme de virtualisation qui permet de déployer des applications dans des conteneurs isolés, facilitant ainsi l'intégration continue et la portabilité de l'application web et des modèles prédictifs.

## 1-3. METHODOLOGIE DES APPROCHES EXISTANTES

### Approches en Machine Learning

Les approches en ML appliquées à la prédiction des prêts incluent l'utilisation de modèles supervisés pour classer les clients en fonction de leur probabilité d'éligibilité. Les modèles supervisés comme la régression logistique sont couramment utilisés pour ce type de tâche, en se basant sur des données historiques pour entraîner le modèle et faire des prédictions sur de nouveaux clients.

### Applications Pratiques dans le Domaine Bancaire

Les modèles de prédiction des prêts sont utilisés pour diverses applications pratiques, notamment :

- **Évaluation du Risque de Crédit** : En analysant les données des clients, les banques peuvent prédire la probabilité qu'un client rembourse un prêt, ce qui permet de réduire les risques de défaut de paiement.

- **Optimisation du Portefeuille de Prêts** : Les modèles permettent aux banques d'optimiser leur portefeuille de prêts en équilibrant le risque et la rentabilité.

## 1-4. SYNTHESE DES MEILLEURES PRATIQUES

### Comparaison des Méthodes

Les méthodes de prédiction des prêts varient en fonction de leur complexité et de leur précision. Par exemple, les arbres de décision offrent une interprétabilité plus élevée, tandis que les modèles plus complexes comme les réseaux de neurones peuvent fournir des prédictions plus précises mais moins interprétables. Le choix de la méthode dépend des objectifs spécifiques du projet et des ressources disponibles.

### Choix Méthodologiques pour le Projet

Pour ce projet, le choix des modèles a été basé sur leur capacité à gérer les caractéristiques spécifiques des données des clients et leur efficacité dans la prédiction de l'éligibilité des prêts. Les modèles de régression logistique et KNN ont été sélectionnés pour leur simplicité et leur efficacité dans la gestion des données de crédit.

## 1-5. IDENTIFICATION DES LACUNES

### Limites des Études Précédentes

Les études précédentes montrent que les modèles de prédiction des prêts peuvent être limités par la qualité et la quantité des données disponibles. Les données manquantes ou inexactes peuvent affecter la précision des prédictions.

### Opportunités pour la Recherche Future

Il existe des opportunités pour améliorer les modèles de prédiction en utilisant des techniques avancées comme les réseaux de neurones profonds et en intégrant des données supplémentaires comme les comportements des clients en ligne.

# 2eme PARTIE : CADRE PRATIQUE

# Chapitre 1 : Développement du Modèle de Prédiction

## 1-1. Objectif du Modèle

L'objectif principal du modèle de prédiction est de déterminer l'éligibilité d'un client à un prêt en fonction de ses caractéristiques personnelles et financières. En conséquence les caractéristiques du modèle doivent être en accord avec l'objectif, surtout dans le choix de nos variables d'entrée

## 1-2. Sélection et Préparation des Données

### *Source des Données*

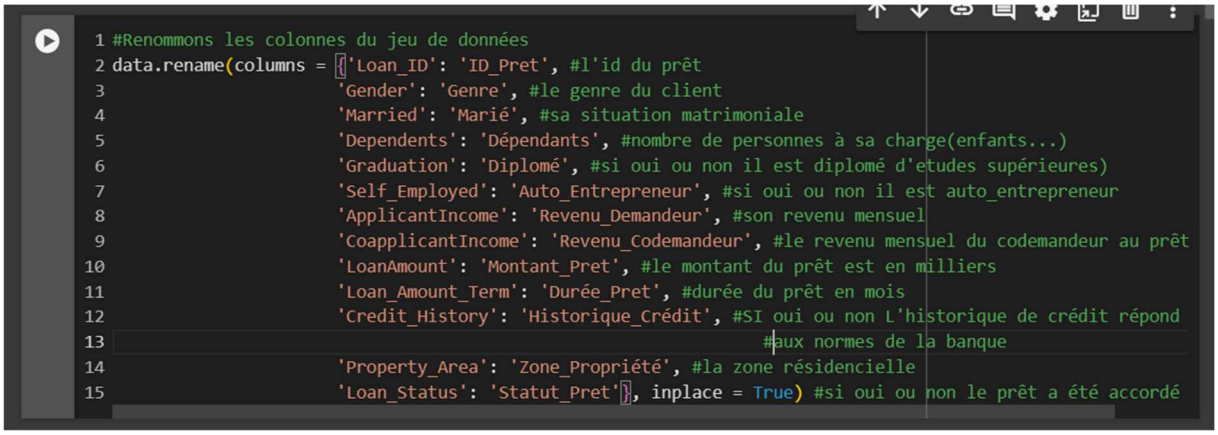
Le jeu de données utilisé pour le développement du modèle a été téléchargé depuis **Analytics Vidhya[3].** Ce choix a été fait après une recherche approfondie sur plusieurs jeux de données disponibles sur **Kaggle**[4]. L'objectif était de trouver un jeu de données suffisamment pertinent pour entraîner un modèle capable de prédire l'éligibilité aux prêts avec précision. Le jeu de données de Analytics Vidhya a été retenu en raison de sa qualité et de sa pertinence pour le problème à traiter. Une fois téléchargée, nous l'avons ensuite importé sur **Google Colaboratory**[5] en commençant bien évidemment par l'importation des modules et bibliothèques que nous utiliserons.


*Figure 4 : Importation des bibliothèques*

[3] Communauté de science des données interactive en Inde.
[4] Plateforme web qui accueille l'une des plus grandes communautés de Data Science au monde.
[5] IDE qui permet à tout utilisateur d'écrire du code source dans son éditeur et de l'exécuter depuis le navigateur

## *Réparation des données*

Afin de mieux faire correspondre le jeu de données avec nos besoins, nous avons décider de procéder à quelques petites modifications notamment ;



```
#Renommons les colonnes du jeu de données
data.rename(columns = {'Loan_ID': 'ID_Pret', #l'id du prêt
                       'Gender': 'Genre', #le genre du client
                       'Married': 'Marié', #sa situation matrimoniale
                       'Dependents': 'Dépendants', #nombre de personnes à sa charge(enfants...)
                       'Graduation': 'Diplomé', #si oui ou non il est diplomé d'etudes supérieures)
                       'Self_Employed': 'Auto_Entrepreneur', #si oui ou non il est auto_entrepreneur
                       'ApplicantIncome': 'Revenu_Demandeur', #son revenu mensuel
                       'CoapplicantIncome': 'Revenu_Codemandeur', #le revenu mensuel du codemandeur au prêt
                       'LoanAmount': 'Montant_Pret', #le montant du prêt est en milliers
                       'Loan_Amount_Term': 'Durée_Pret', #durée du prêt en mois
                       'Credit_History': 'Historique_Crédit', #SI oui ou non L'historique de crédit répond
                                                              #aux normes de la banque
                       'Property_Area': 'Zone_Propriété', #la zone résidencielle
                       'Loan_Status': 'Statut_Pret'}, inplace = True) #si oui ou non le prêt a été accordé
```

```
#Reformatons les données de certaines variables catégorielles
data.replace({"Male": "M", "Female": "F", "Yes": "O", "No": "N",
              "Graduate": "O", "Not Graduate": "N", "Y":"O"}, inplace = True)
```

```
[8] #Convertissons la variable Historique_Crédit en catégorielle
    data["Historique_Crédit"] = data["Historique_Crédit"].astype("object")
```

Les colonnes du jeu de données ont été renommées pour plus de clarté. Les variables catégorielles ont été reformatées pour une meilleure uniformité.

## *Analyse Exploratoire des Données*

Une fois nos modifications terminées, nous pouvons voir à quoi ressemble maintenant notre jeu de données.

Nous affichons également quelques informations comme la taille, la description statistique de nos données ainsi que les types de chaque colonne. Tout ça respectivement avec « data.shape » « *data.describe(include ="all")* » , « data.info() ». Cela nous renseigne sur le fait que nous avons à notre disposition 614 insertions (lignes) pour 13 variables (colonnes), sur les différentes

mesures statistiques appliquées à ces variables (mesures de dispersions, de tendances et de fréquences) et enfin sur le type de chaque variable (catégorielle, entière ou réelle).

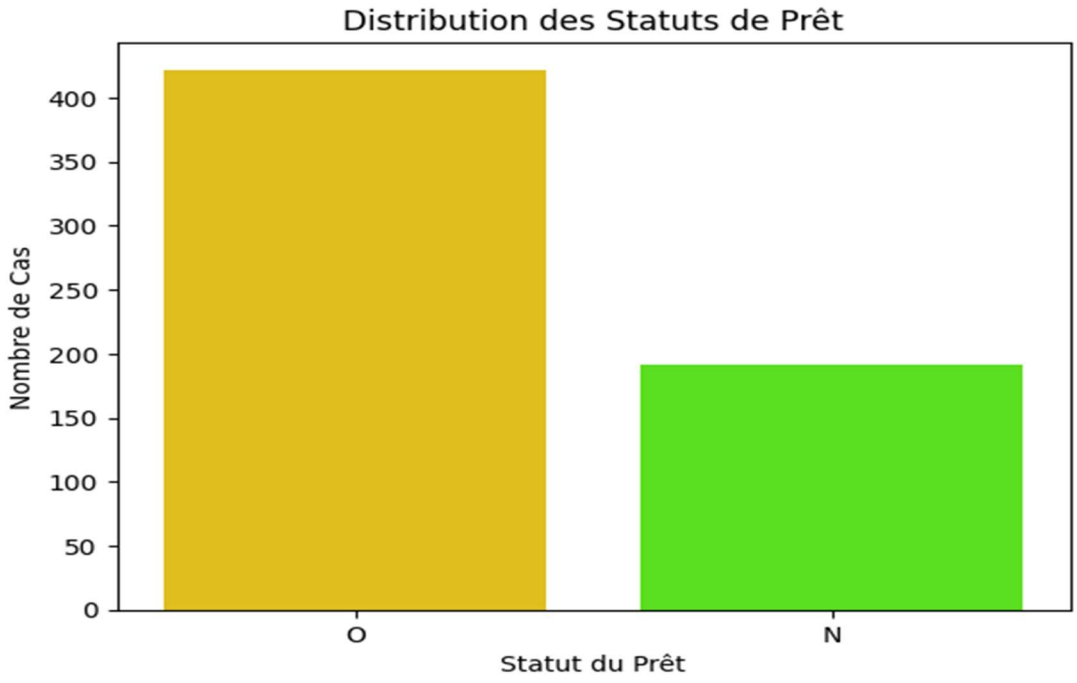
1 to 20 of 20 entries Filter

| index | ID_Pret | Genre | Marié | Dépendants | Education | Auto_Entrepreneur | Revenu_Demandeur | Revenu_Codemandeur | Montant_Pret | Durée_Pret | Historique_Crédit | Zone_Propriété | Statut_Pret |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | LP001002 | M | N | 0 | O | N | 5849 | 0.0 | NaN | 360.0 | 1.0 | Urban | O |
| 1 | LP001003 | M | O | 1 | O | N | 4583 | 1508.0 | 128.0 | 360.0 | 1.0 | Rural | N |
| 2 | LP001005 | M | O | 0 | O | O | 3000 | 0.0 | 66.0 | 360.0 | 1.0 | Urban | O |
| 3 | LP001006 | M | O | 0 | N | N | 2583 | 2358.0 | 120.0 | 360.0 | 1.0 | Urban | O |
| 4 | LP001008 | M | N | 0 | O | N | 6000 | 0.0 | 141.0 | 360.0 | 1.0 | Urban | O |
| 5 | LP001011 | M | O | 2 | O | O | 5417 | 4196.0 | 267.0 | 360.0 | 1.0 | Urban | O |
| 6 | LP001013 | M | O | 0 | N | N | 2333 | 1516.0 | 95.0 | 360.0 | 1.0 | Urban | O |
| 7 | LP001014 | M | O | 3+ | O | N | 3036 | 2504.0 | 158.0 | 360.0 | 0.0 | Semiurban | N |
| 8 | LP001018 | M | O | 2 | O | N | 4006 | 1526.0 | 168.0 | 360.0 | 1.0 | Urban | O |
| 9 | LP001020 | M | O | 1 | O | N | 12841 | 10968.0 | 349.0 | 360.0 | 1.0 | Semiurban | N |
| 10 | LP001024 | M | O | 2 | O | N | 3200 | 700.0 | 70.0 | 360.0 | 1.0 | Urban | O |
| 11 | LP001027 | M | O | 2 | O | NaN | 2500 | 1840.0 | 109.0 | 360.0 | 1.0 | Urban | O |
| 12 | LP001028 | M | O | 2 | O | N | 3073 | 8106.0 | 200.0 | 360.0 | 1.0 | Urban | O |
| 13 | LP001029 | M | N | 0 | O | N | 1853 | 2840.0 | 114.0 | 360.0 | 1.0 | Rural | N |
| 14 | LP001030 | M | O | 2 | O | N | 1299 | 1086.0 | 17.0 | 120.0 | 1.0 | Urban | O |
| 15 | LP001032 | M | N | 0 | O | N | 4950 | 0.0 | 125.0 | 360.0 | 1.0 | Urban | O |
| 16 | LP001034 | M | N | 1 | N | N | 3596 | 0.0 | 100.0 | 240.0 | NaN | Urban | O |
| 17 | LP001036 | F | N | 0 | O | N | 3510 | 0.0 | 76.0 | 360.0 | 0.0 | Urban | N |
| 18 | LP001038 | M | O | 0 | N | N | 4887 | 0.0 | 133.0 | 360.0 | 1.0 | Rural | N |
| 19 | LP001041 | M | O | 0 | O | NaN | 2600 | 3500.0 | 115.0 | NaN | 1.0 | Urban | O |

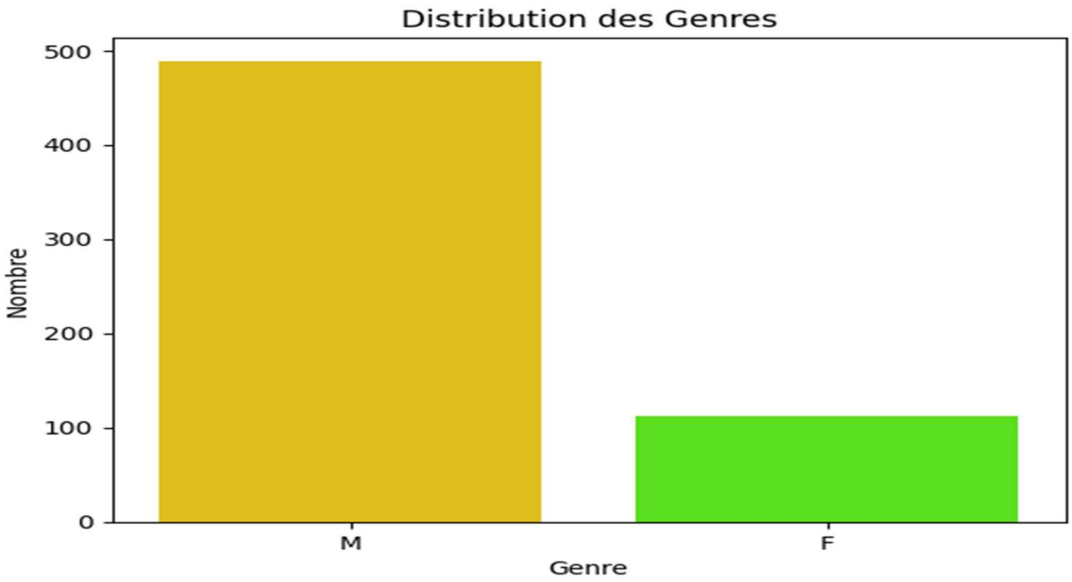
Show 25 per page

Maintenant, visualisons les variables et interprétons afin de voir les variables clés qui donnent l'éligibilité à un prêt.

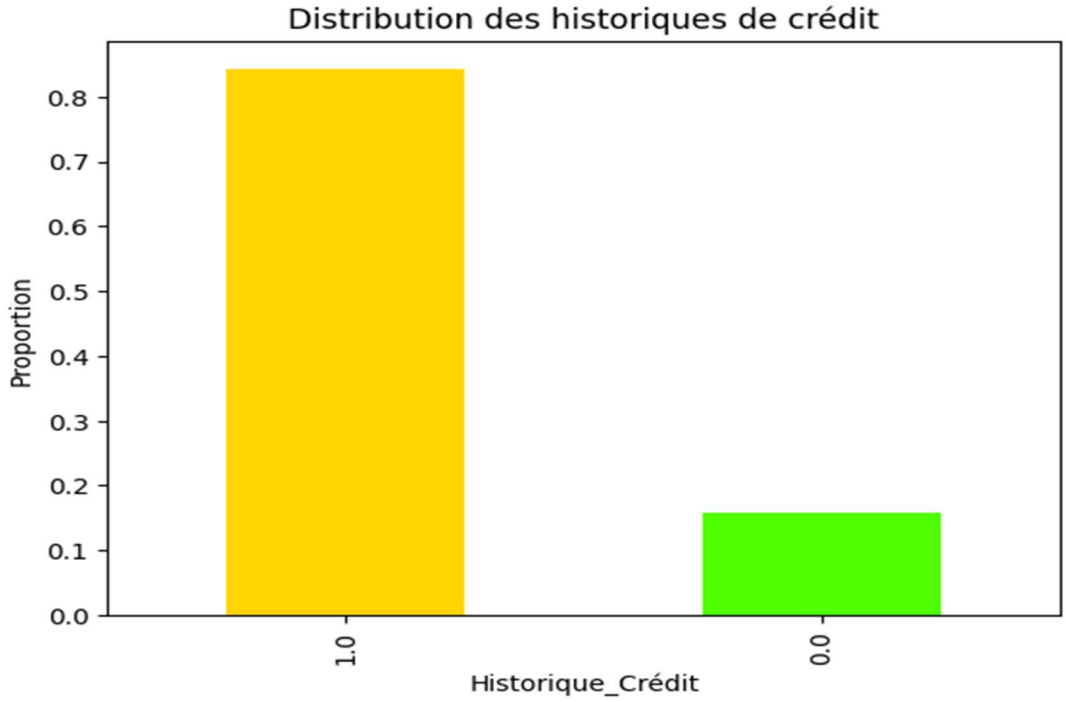
## *Analyse univariée des Données*



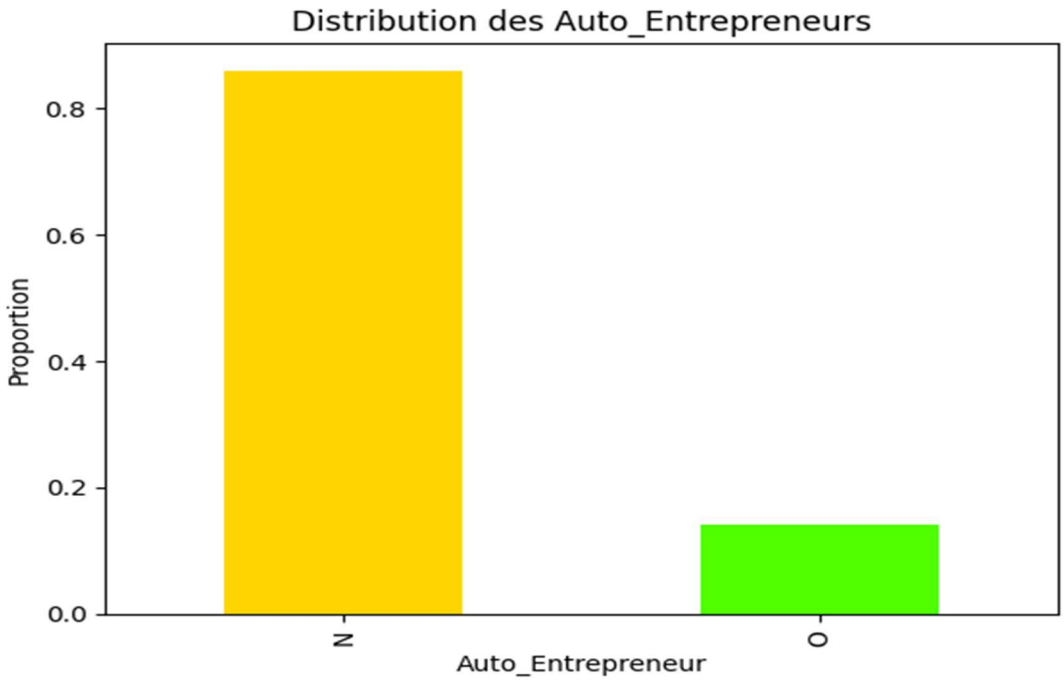

Ici, il ressort qu'environ 69% des prêts (422 personnes sur 614) ont été approuvés.



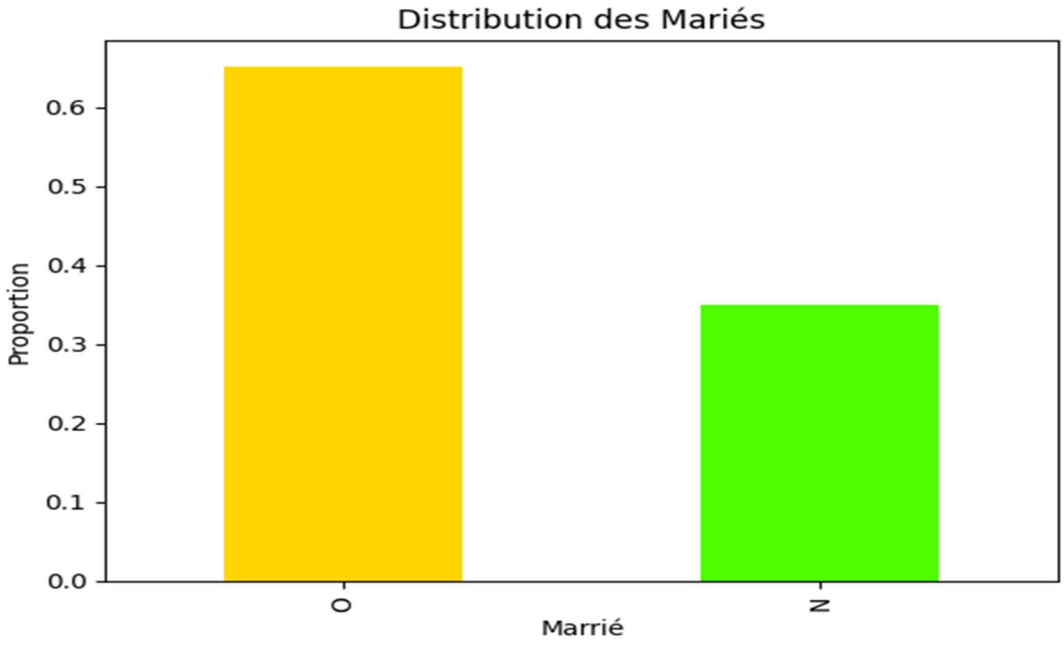

80 % des candidats dans l'ensemble de données sont donc des hommes.



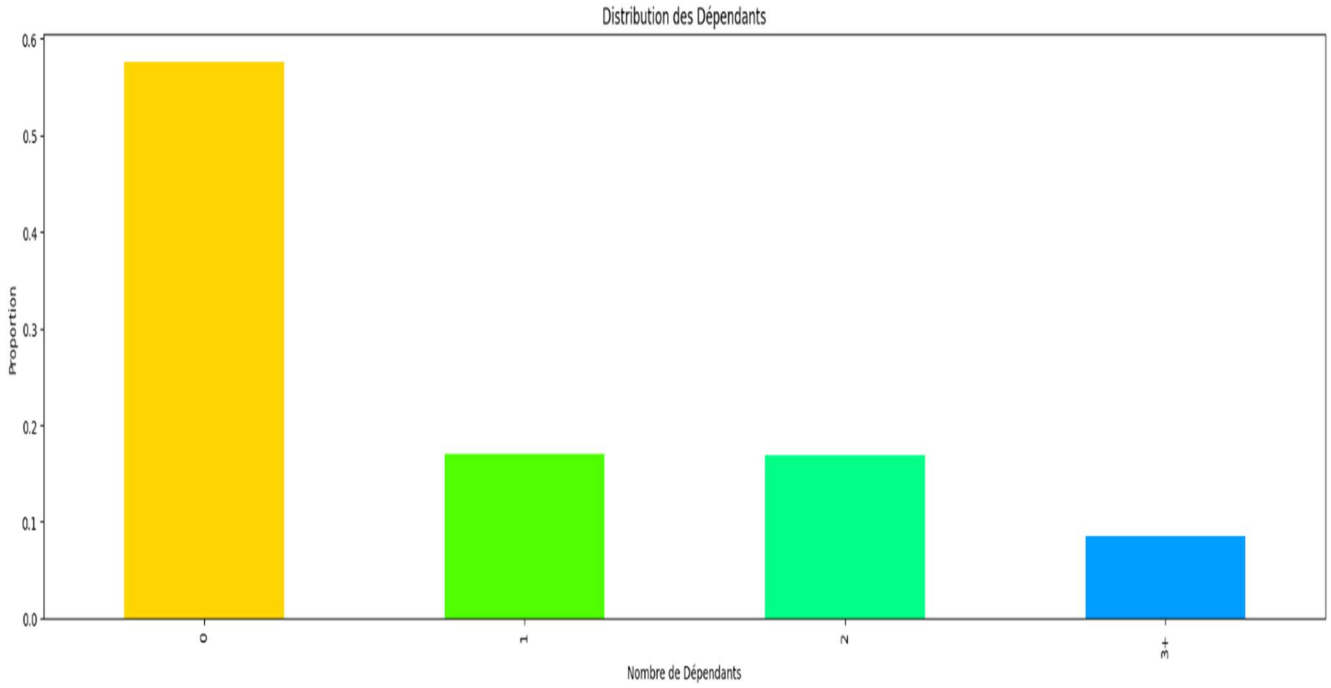

Environ 85 % des candidats ont un HC positif et respecte donc les normes de la banque concernant les crédits.



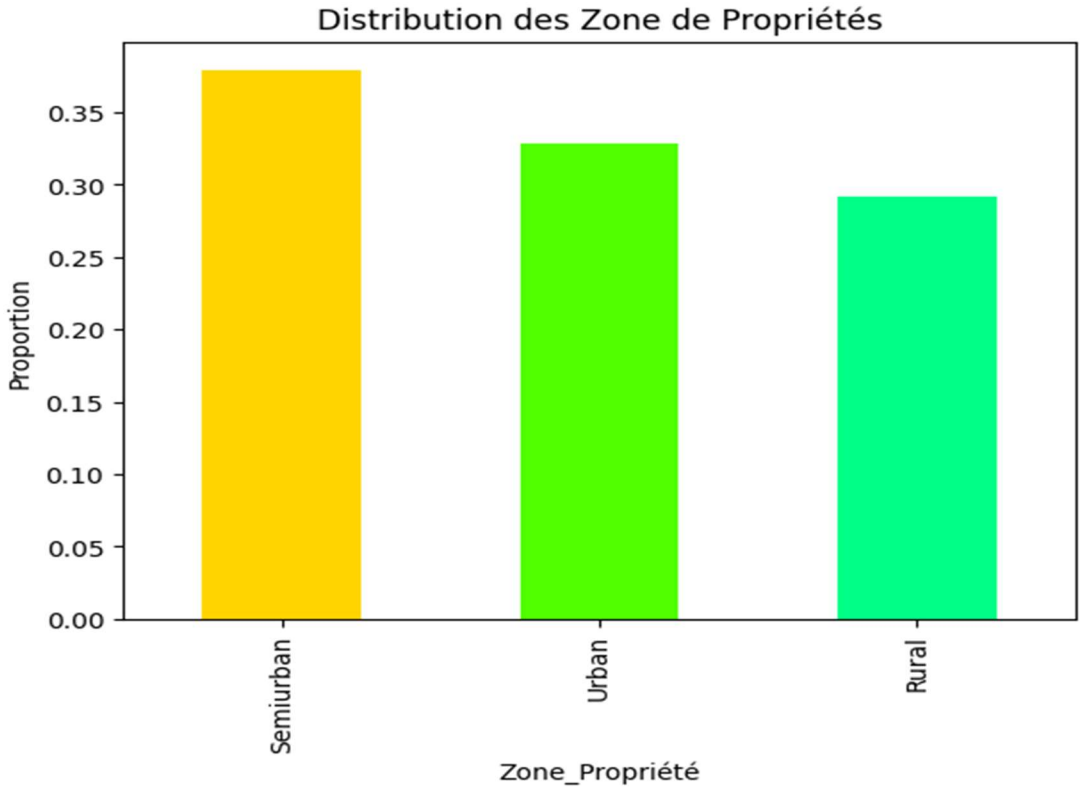

Seulement 15 % des candidats figurant dans l'ensemble de données sont des travailleurs indépendants.



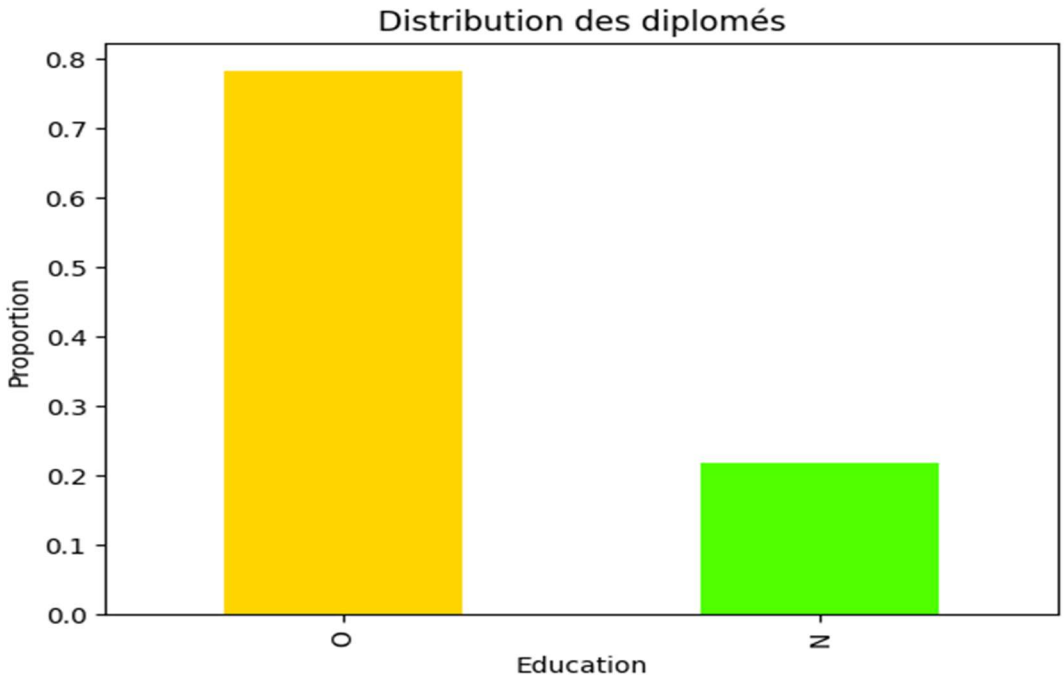

Environ 65 % des candidats figurant dans l'ensemble de données sont mariés.



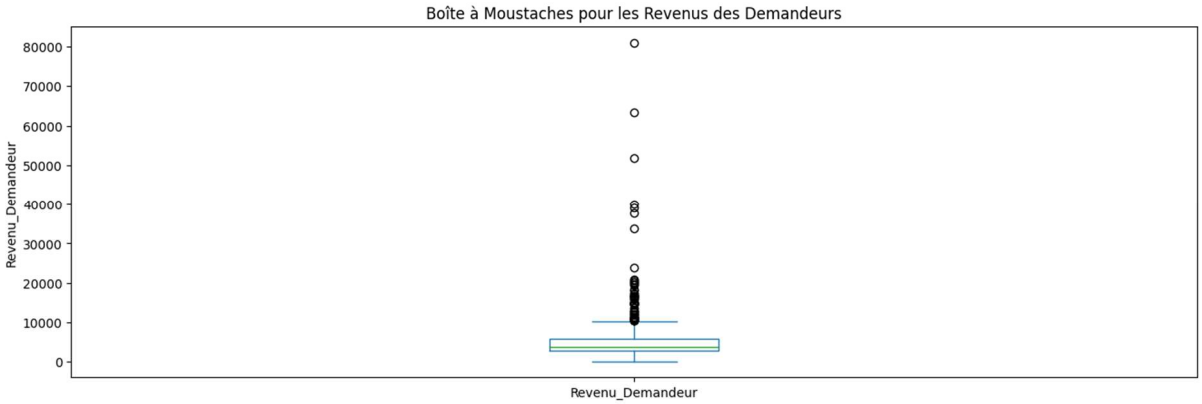

La plupart des candidats n'ont pas de personnes à charge.



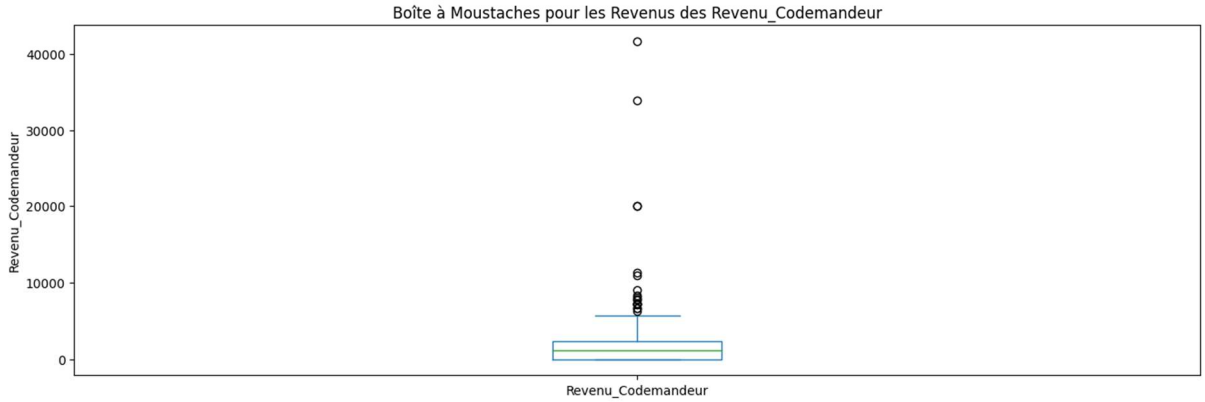

La plupart des candidats viennent de la zone semi-urbaine.



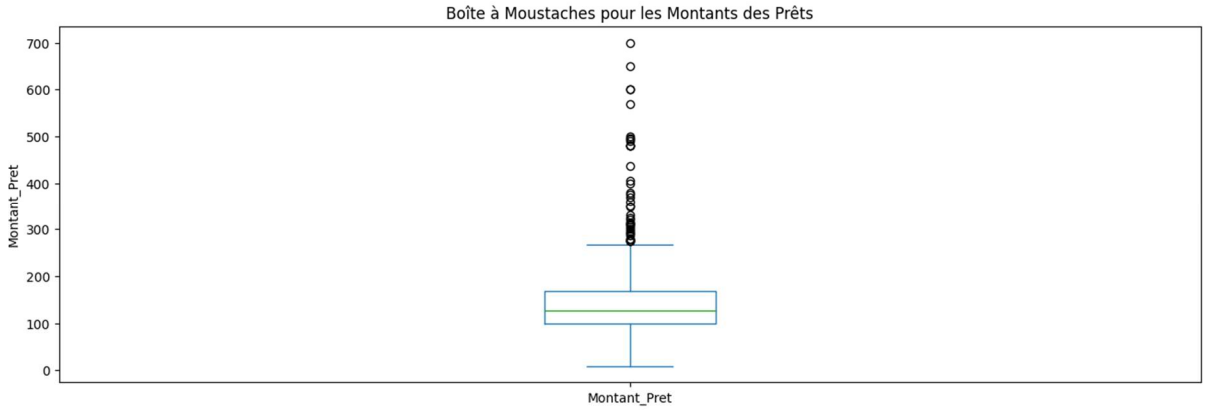

Environ 80 % des candidats sont des diplômés.




Le diagramme en boîte confirme la présence de nombreuses valeurs aberrantes. Cela peut être attribué à la disparité des revenus dans la société.



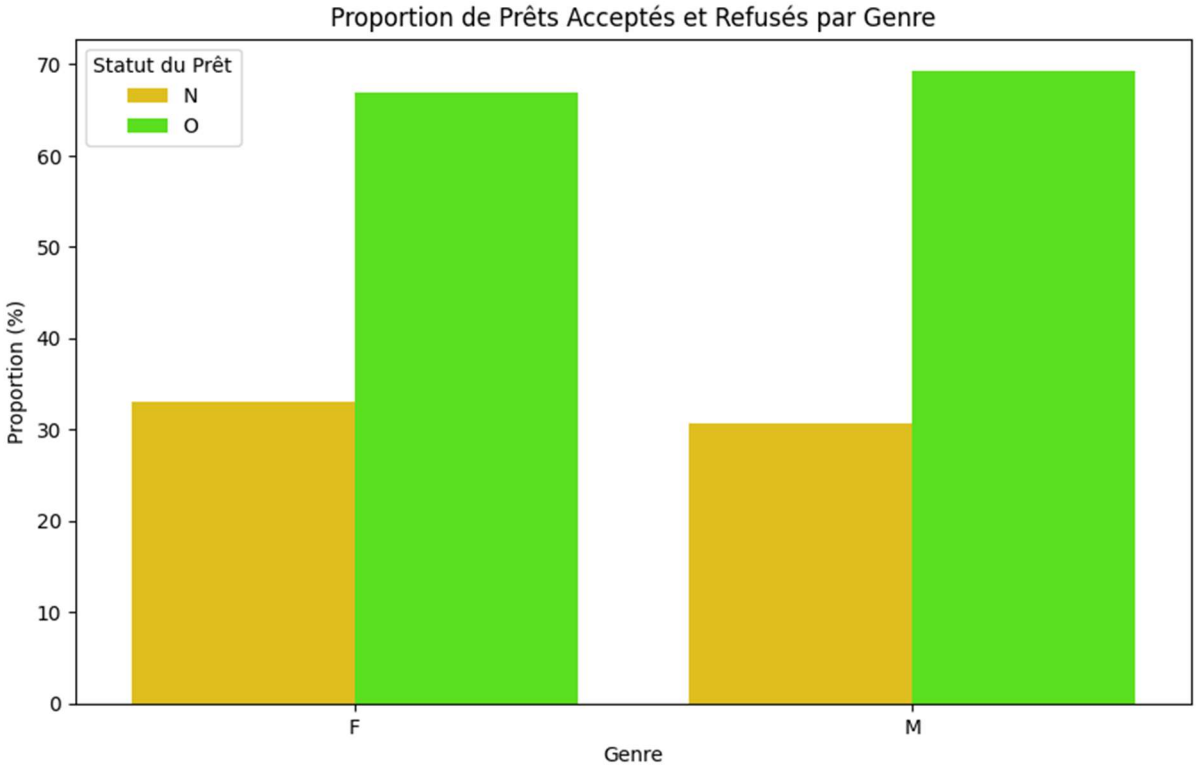

Nous observons également de nombreuses valeurs aberrantes dans les revenus du codemandeur et ils ne sont pas normalement distribués.



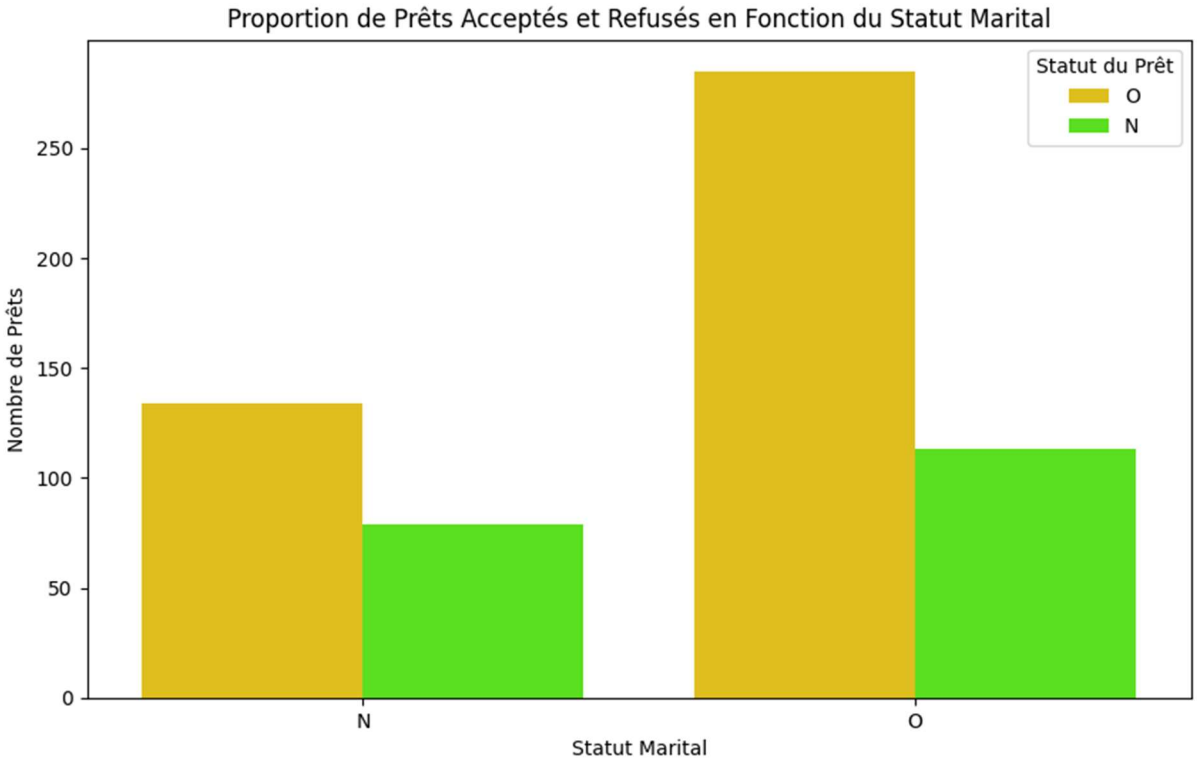

Nous observons de nombreuses valeurs aberrantes dans cette variable.

## *Analyse bivariée des Données*

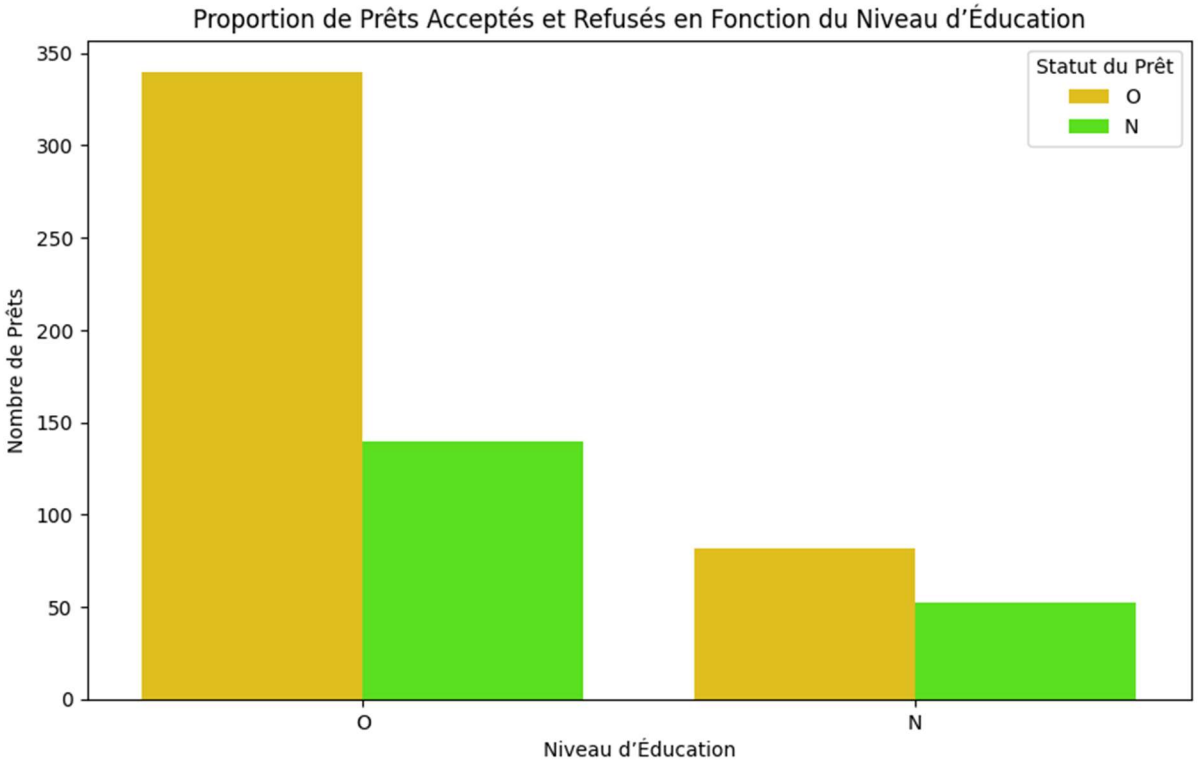
Ici, nous allons confrontés chacune des variables d'entrée avec notre variable cible (Statut du prêt). Cela pourrait nous donner des informations sur ce qui caractérise l'éligibilité à un prêt.

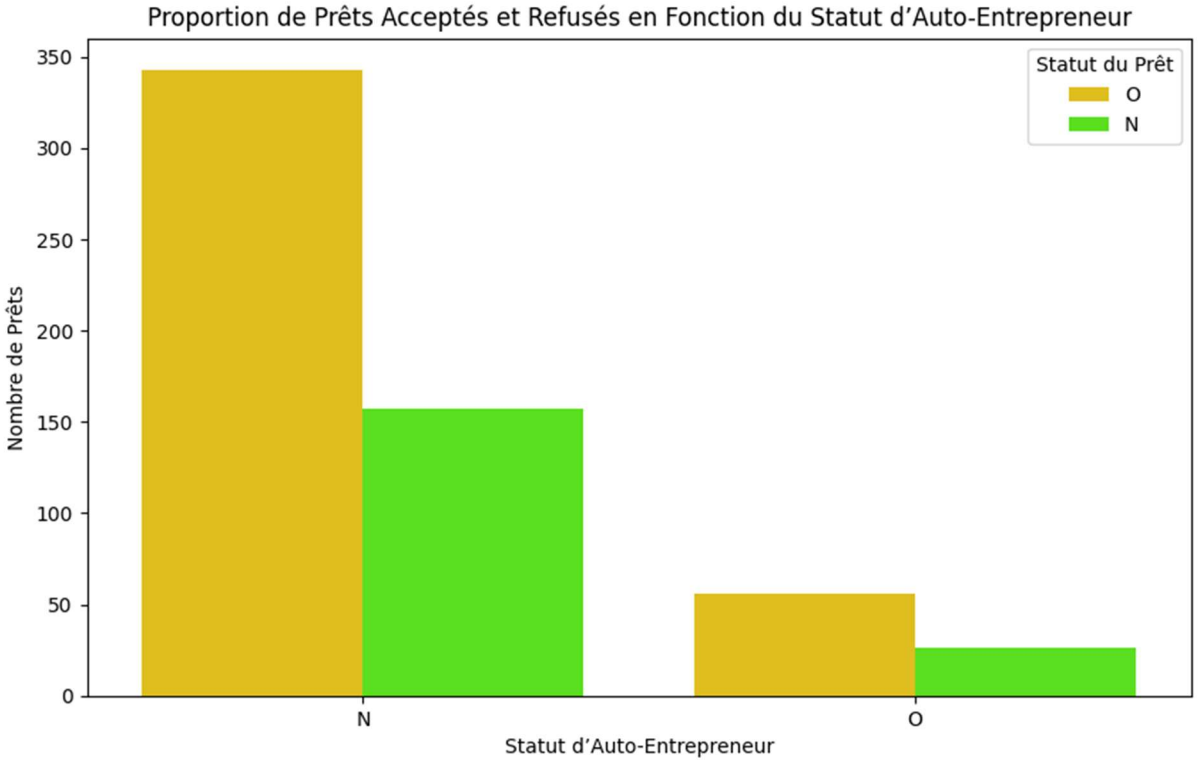


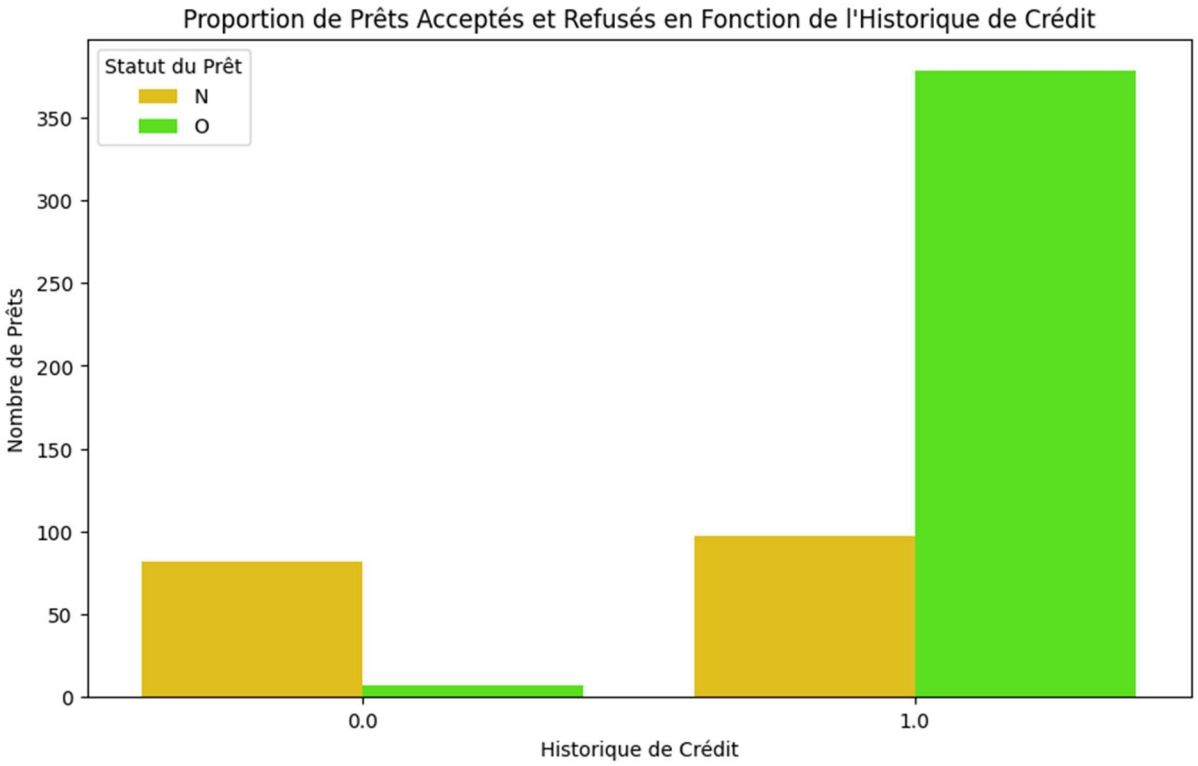

On peut en déduire que la proportion de demandeurs hommes et femmes est plus ou moins la même pour les prêts approuvés et non approuvés.



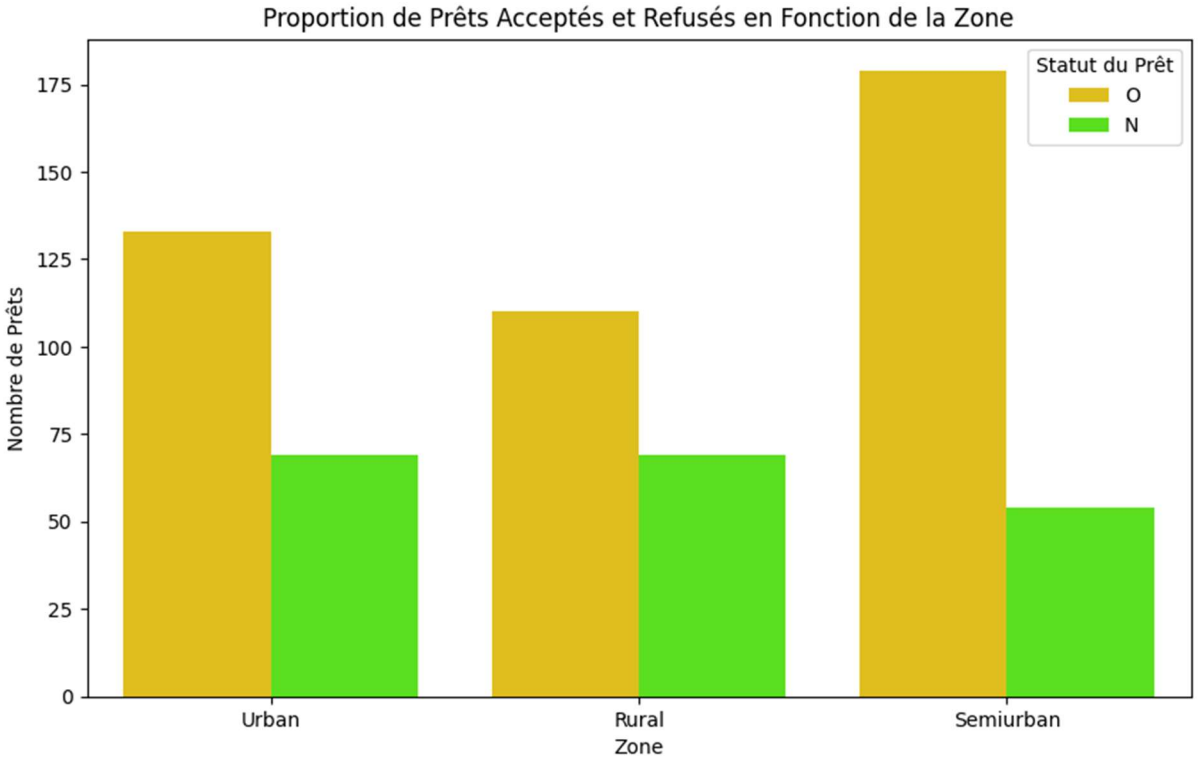

La proportion de demandeurs mariés est plus élevée pour les prêts approuvés.


Proportion de Prêts Acceptés et Refusés en Fonction du Niveau d'Éducation
Statut du Prêt
O
N
Nombre de Prêts
0
50
100
150
200
250
300
350
O
N
Niveau d'Éducation

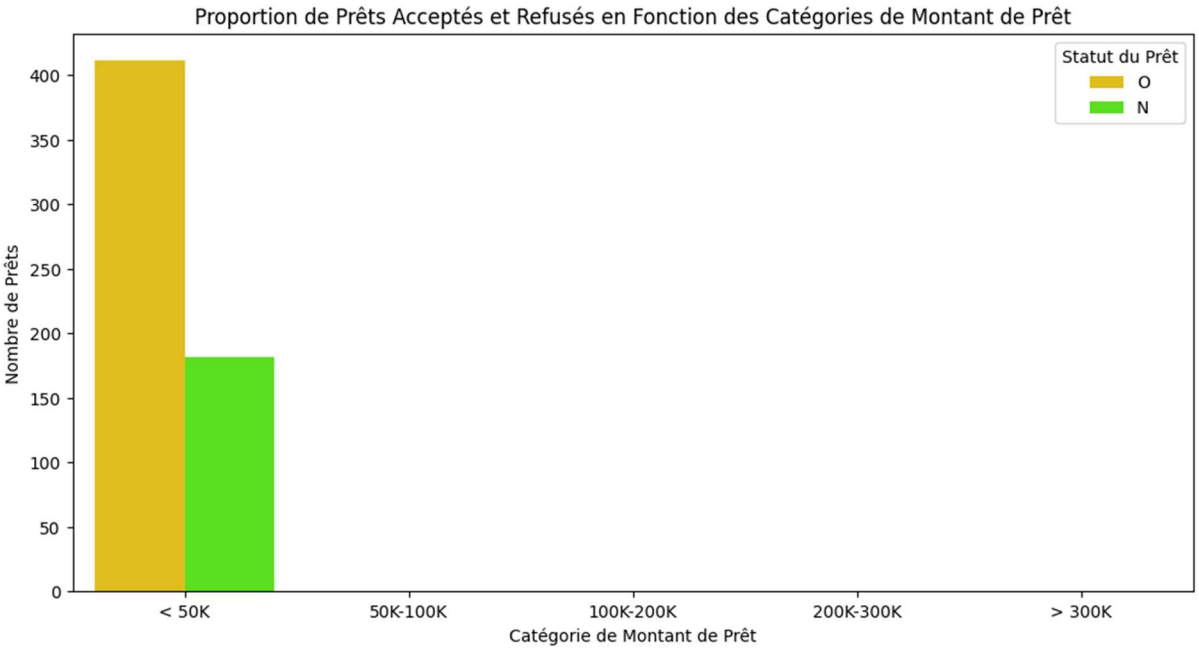

Proportion de Prêts Acceptés et Refusés en Fonction du Statut d'Auto-Entrepreneur
Statut du Prêt
O
N
Nombre de Prêts
0
50
100
150
200
250
300
350
N
O
Statut d'Auto-Entrepreneur




Il semble que les personnes ayant des antécédents de crédit comme 1 ont plus de chances de voir leurs prêts approuvés.



Il semble que les personnes vivantes dans la zone semi-urbaine ont plus de chances de voir leurs prêts approuvés.




Comme on pouvait s'en douter, les chances d'approbation du prêt sont élevées lorsque le montant du prêt est bas.

## *Exploration et Prétraitement des Données*

**Traitement des valeurs manquantes** :

En observant le tableau précédent, nous nous apercevons de la présence de valeurs manquantes caractérisé par l'apparition de « NaN[6] ». Cela s'est rapidement confirmé avec « data.isnull().sum() » qui nous a fourni le nombre exact de valeurs manquantes par colonne. Les valeurs manquantes peuvent poser plusieurs problèmes dont un impact sur la qualité des modèles[7], des biais dans les résultats [8] et une diminution de la puissance statistique[9].

[6] Not a Number
[7] Les algorithmes peuvent avoir du mal à faire des prédictions précises si les données d'entrée ne sont pas complètes.
[8] Si les données manquantes ne sont pas aléatoires, les modèles peuvent être biaisés vers des conclusions incorrectes.
[9] La présence de valeurs manquantes réduit la taille effective de l'échantillon, ce qui peut diminuer la puissance statistique des analyses

Pour les gérer, plusieurs options sont disponibles :

- Suppression des Lignes :

  Avantages : Simple à mettre en œuvre et permet de conserver uniquement les lignes complètes, ce qui peut simplifier les analyses ultérieures.

  Inconvénients : La suppression de lignes peut entraîner une perte importante d'informations, surtout si les valeurs manquantes sont nombreuses. Cela peut également réduire la taille de l'échantillon, impactant la puissance des analyses et la représentativité des données.

- Remplacement par une Valeur Fixe : On peut remplacer les valeurs manquantes par une valeur constante, telle que 0 ou une autre valeur significative.

  Avantages : Simple à mettre en œuvre, utile lorsque les valeurs manquantes ont une signification spécifique.

  Inconvénients : Peut introduire des biais si la valeur de remplacement n'est pas représentative des données manquantes.

- Imputation par la Moyenne/Mode/Mediane : On remplace les valeurs manquantes par la moyenne, le mode ou la médiane des données disponibles pour la variable.

  Avantages : Méthode courante et facile à appliquer. Préserve la structure globale des données.

  Inconvénients : Peut réduire la variabilité des données et introduire des biais si les données sont asymétriques ou si la distribution est fortement biaisée.

- Imputation par Modèles : Utilisation de modèles statistiques (comme la régression) ou de techniques de ML (comme KNN) pour prédire les valeurs manquantes en fonction des autres variables.

  Avantages : Peut fournir une estimation plus précise des valeurs manquantes en utilisant les relations entre variables.

Inconvénients : Complexe à mettre en œuvre et peut introduire un biais si les modèles ne sont pas correctement ajustés.

Dans notre cas, nous avons décidé d'utiliser l'Imputation par la Moyenne/Mode/Mediane car il n'y a pas beaucoup de valeurs manquantes (1.87% seulement) dans notre jeu de données.

```
[ ]  1 na_total = data.isnull().sum().sum()
     2 pourcentage_na_total = (na_total * 100) / data.size
     3 na_total, pourcentage_na_total

     (149, 1.8667000751691305)
```

```
[ ]  1 #Traitons les données manquantes des variables catégorielles
     2 VarCat = ["Genre","Marié","Dépendants","Education","Auto_Entrepreneur","Historique_Crédit","Zone_Propriété"]
     3 for i in VarCat:
     4   data[i].fillna(data[i].mode()[0],inplace = True)
```

```
[ ]  1 #Traitons les données manquantes des variables numériques
     2 data["Montant_Pret"].fillna(data["Montant_Pret"].median(),inplace = True) #continue
     3 data["Durée_Pret"].fillna(data["Durée_Pret"].mode()[0],inplace = True) #discret
```

Et voilà, nous avons nettoyé nos données des valeurs manquantes.

**Traitement des valeurs aberrantes** :

Pour le traitement de ces valeurs observées au niveau de l'analyse univariée, nous avons utiliser la transformation logarithmique. En effet, c'est une technique utilisée en statistique et en data science pour transformer des données en prenant le logarithme de chaque valeur. Elle est particulièrement utile lorsque les données ont une distribution asymétrique ou une grande variation entre les valeurs comme c'est le cas avec notre variable Montant_Pret.

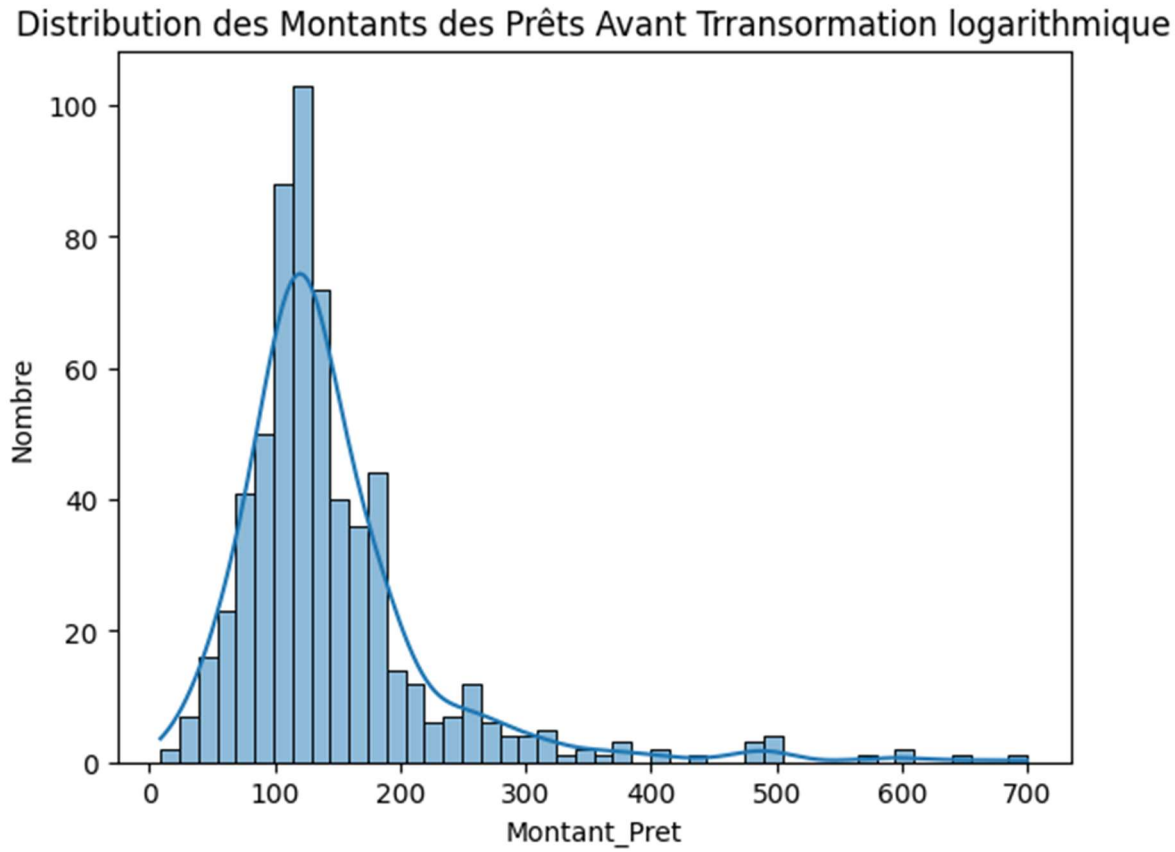
```python
#comparons le mode et la moyenne de la série
data['Montant_Pret'].mean(), data['Montant_Pret'].mode()[0]
#on constate que le mode et la moyenne sont totalement different, normal au vu de
                    #l'asymetrie (comme constaté sur le graphe)
```

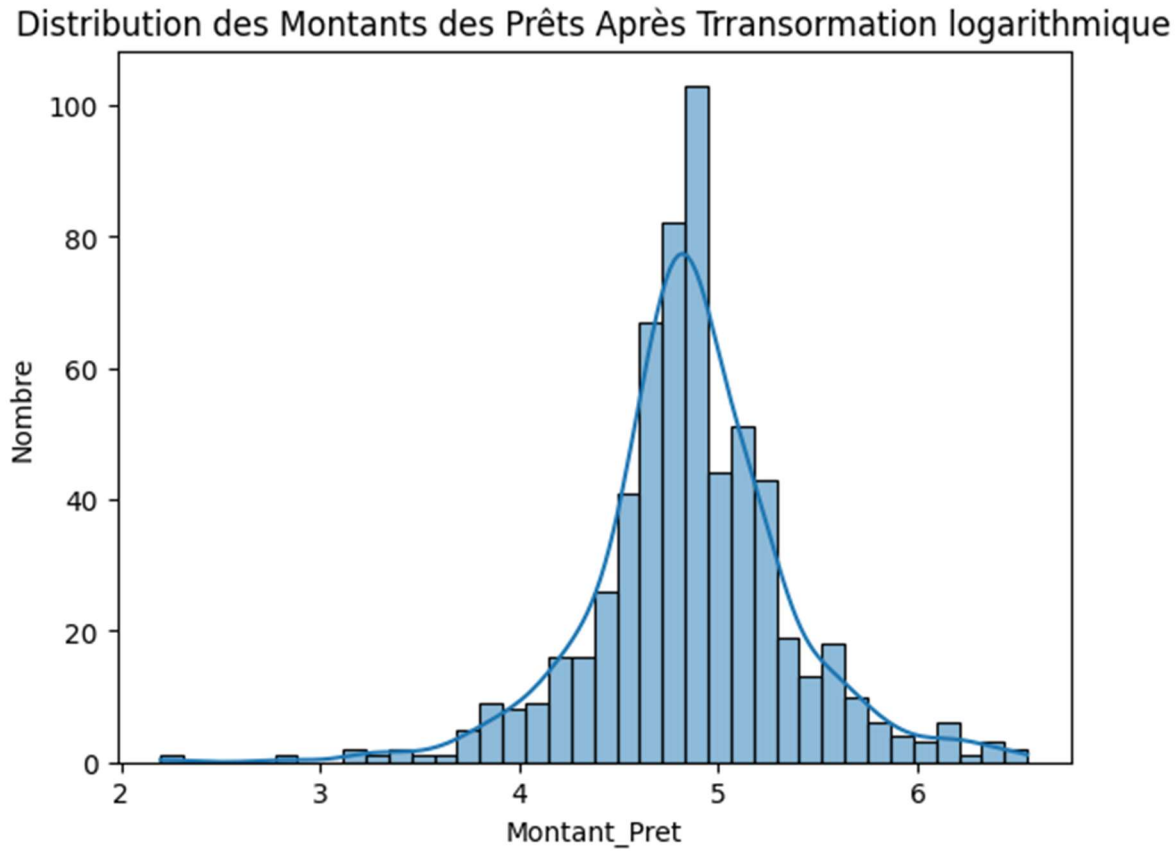
```
(145.75244299674267, 128.0)
```

```python
#comparons le mode et la moyenne de la série apres transformation logarithmique
np.log(data['Montant_Pret']).mean(), np.log(data['Montant_Pret']).mode()[0]
#on constate que le mode et la moyenne sont quasiment  identique, ce qui revele une symetrie.
                                        #Pour en etre sur tracons le graphe a nouveau
```

```
(4.857250194811088, 4.852030263919617)
```

```python
sns.histplot(np.log(data['Montant_Pret']),kde = True) #kernel density estimation
```






On constate bien que les données sont un peu plus symétriques au vu de la courbe kde [10] qui a une forme équilibrée de chaque côté du pic.

**Encodage des variables catégorielles**

Comme annoncé plus haut, nous avons décidé d'utiliser les modèles de RL et de KNN, ces deux modèles ne peuvent traiter que des variables numériques. Pour y remédier donc, nos variables catégorielles ont été converties en format numérique pour être utilisées dans les modèles de prédiction grâce à une technique d'encodage nommée « LabelEncoder ». En effet, le **LabelEncoder** scanne chaque modalité de la variable catégorielle et attribue à chacune une valeur entière unique. Il est simple à implémenter et permet de conserver les informations ordinales lorsque c'est nécessaire.

[10] kernel density estimation

```
[414]  1 from sklearn.preprocessing import LabelEncoder
       2 for i in VarCat:
       3   data[i] = LabelEncoder().fit_transform(data[i])
       4
```

**Fractionnement des données**

Les données ont été divisées en deux ensembles distincts : un jeu d'entraînement pour entraîner le modèle, et un jeu de test pour évaluer ses performances. Cette étape est cruciale pour éviter le surapprentissage et garantir que le modèle peut bien généraliser à des données non vues.

```
1 #variables d'entrées
2 X = data.drop(["ID_Pret","Statut_Pret"], axis = 1) #on retire ici la colonne ID_pret qui
3                                                    #n'est pas utile à notre modèle
4
5 #Variable cible
6 y = data["Statut_Pret"]
```

**Normalisation ou standardisation des variables**

Les variables numériques ont été normalisées pour garantir que toutes les caractéristiques aient la même échelle grâce au « **StandardScaler** ». Cette étape est particulièrement importante pour les modèles sensibles aux distances comme la régression logistique ou les k-plus proches voisins (KNN). En effet, le **StandardScaler** transforme chaque caractéristique individuelle pour qu'elle ait une distribution avec une moyenne égale à zéro et une variance égale à un. Notons que seules les variables d'entrée sont normalisées.

```
1 from sklearn.preprocessing import StandardScaler
2 X = StandardScaler().fit_transform(X)
```

**Splitage du jeu de données**

Le jeu de données a été divisé en un ratio de 80/20, avec 80 % des données utilisées pour l'entraînement et 20 % pour le test. Cela permet de s'assurer que le modèle est validé sur un échantillon représentatif de données non vues pendant l'entraînement.

```python
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, train_size= 0.8, random_state=42)
```

En pratique, nous utilisons la fonction « train_test_split » de la bibliothèque Scikit-learn avec les paramètres « train_size [11]» et « random_state [12]» pour diviser les données en un jeu d'entraînement et un jeu de test. Cette fonction prend comme entrée le jeu de données complet et le fractionne de manière aléatoire.

## 1-3. Mise en place des modèles

### KNN

A la recherche du K OPTIMAL

```python
#importation du modele et d'un metric pour mesurer la precision
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
scores = []
for k in range(1,21):
  mon_model_knn = KNeighborsClassifier(n_neighbors=k) #instanciation du modele avec chaque valeur de k
  mon_model_knn.fit(x_train, y_train) #entrainer le modele
  y_pred = mon_model_knn.predict(x_test) #prédiction
  score = accuracy_score(y_test, y_pred) #calcul de la précision du modele
  scores.append(score)

k_optimal = scores.index(max(scores)) + 1 #car le k optimal est celui qui genere le plus grand score de
                                          #precision(max(scores)) et aussi la position de ce max est son index plus 1
k_optimal
```

Une fois le k optimal trouvé nous avons instancié le modèle :

```python
#Réinstanciation du modele
mon_model_knn = KNeighborsClassifier(n_neighbors=k_optimal)
#Réentrainement du modele
mon_model_knn.fit(x_train,y_train)

```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

[11] Ce paramètre contrôle la proportion des données allouées au jeu d'entrainement.

[12] Ce paramètre est utilisé pour contrôler la reproductibilité du fractionnement. En fixant un nombre spécifique comme **random_state=42**, on s'assure que chaque exécution du code produira la même répartition des données, ce qui est utile pour obtenir des résultats cohérents lors de plusieurs tests ou expérimentations.

**Evaluation du modèle**

La précision du modèle KNN a été évaluée à la fois sur les données d'entraînement et de test :

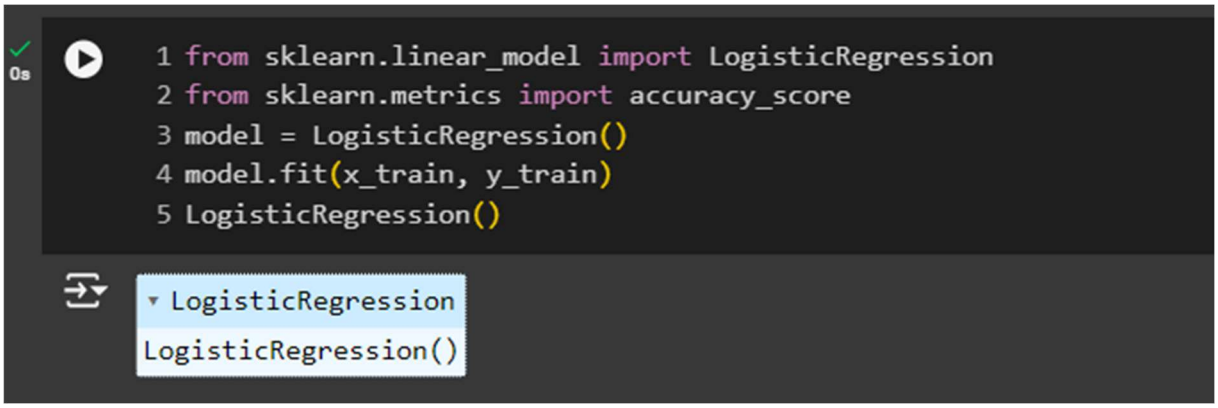
- Précision sur les données d'entraînement : « accuracy_score(y_train, mon_model_knn.predict(x_train)) »
- Précision sur les données de test : « accuracy_score(y_test, y_pred) »

Les résultats ont montré une précision de :

- 81.5 % sur les données d'entraînement
- 79.675 % sur les données de test

Ces résultats indiquent que le modèle KNN généralise raisonnablement bien, même si une légère diminution de la précision est observée sur les données de test. Cela peut être dû au fait que KNN, étant un algorithme non paramétrique, est sensible au bruit [13]dans les données.

RL




[13] Fait référence aux informations ou valeurs qui sont incorrectes, aléatoires ou non pertinentes pour la tâche à accomplir

**Evaluation du modèle**

Les performances du modèle ont été évaluées à l'aide de l'accuracy :

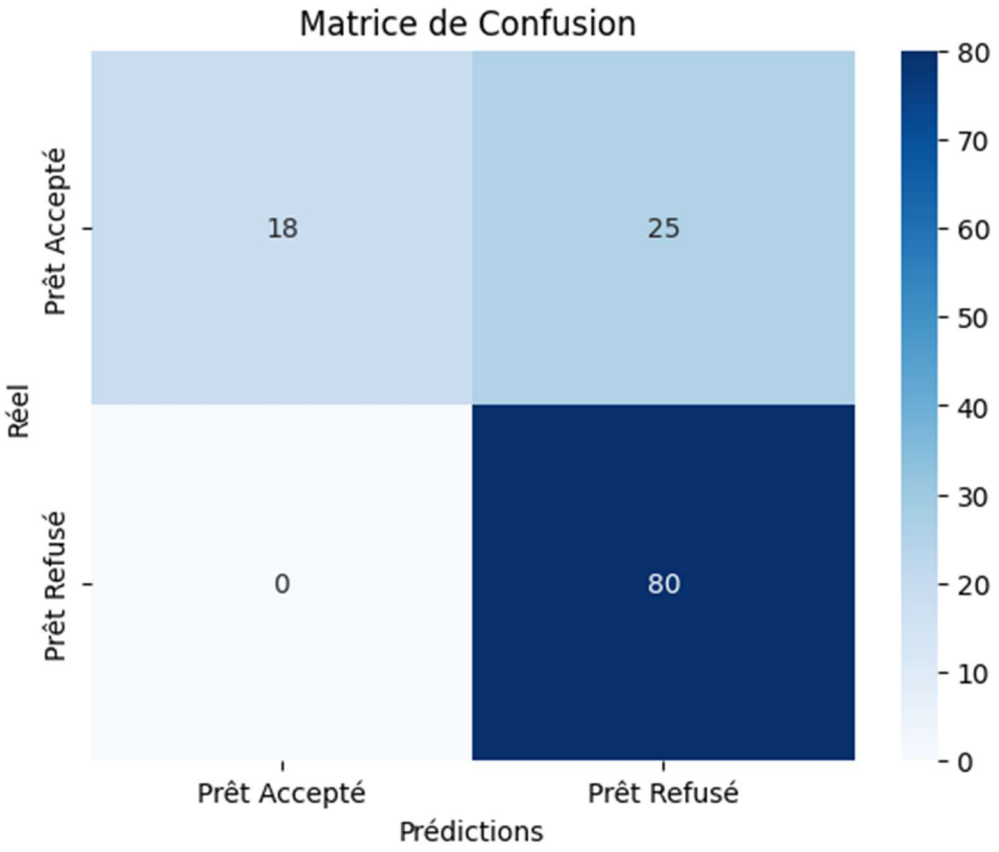
- Précision sur les données d'entraînement : accuracy_score(y_train, model.predict(x_train))
- Précision sur les données de test : accuracy_score(y_test, y_pred)

Les résultats ont montré une précision de :

- 81.874 % sur les données d'entraînement
- 78.862 % sur les données de test

La légère baisse de précision sur les données de test montre que le modèle n'est pas sur-appris (overfitted), ce qui est un bon signe de généralisation.

## Matrice de Confusion




**Interprétations** :

- Vrais Négatifs (18) : Le modèle a correctement prédit que 18 prêts allaient être refusés.
- Faux Positifs (25) : Le modèle a incorrectement prédit que 25 prêts allaient être acceptés alors qu'ils ont été refusés en réalité.
- Faux Négatifs (0) : Il n'y a pas eu d'erreurs où le modèle aurait prédit un refus alors que le prêt était effectivement accepté.
- Vrais Positifs (80) : Le modèle a correctement prédit que 80 prêts allaient être acceptés.

**Observations** :

- Très bon taux de vrais positifs (80) et aucun faux négatif (0) : Cela signifie que le modèle est très bon pour prédire les prêts acceptés.
- Un nombre important de faux positifs (25) : Le modèle a tendance à sur-prédire les prêts acceptés, même lorsque certains d'entre eux devraient être refusés.

## *Comparaison des modèles*

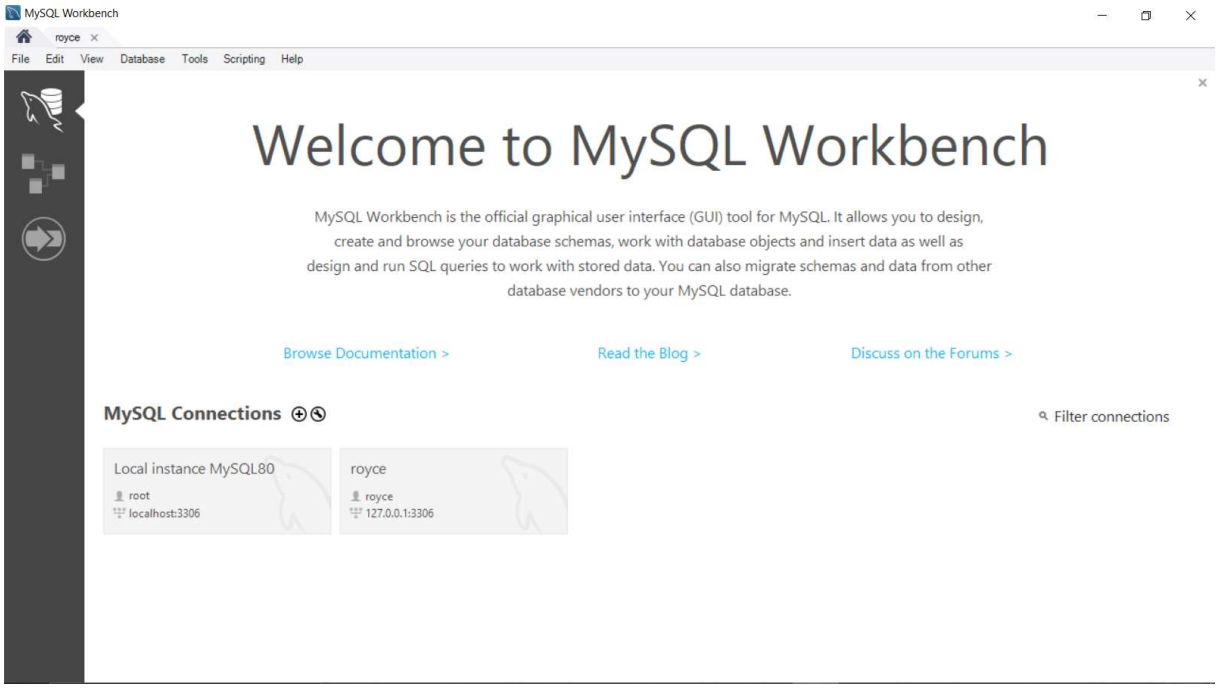
Les résultats ont montré que la régression logistique et KNN ont des performances comparables. La précision du KNN était légèrement supérieure à celle de la régression logistique, ce qui peut être attribué à la capacité du KNN à capturer des relations complexes dans les données.

Nous pouvons maintenant télécharger nos modèles.

```
import pickle
with open("prediction_eligibilite_pret_modelRL_byRoyce.pkl", "wb") as fichier_model:
  pickle.dump(model,fichier_model)
with open("prediction_eligibilite_pret_modelKNN_byRoyce.pkl", "wb") as fichier_model:
  pickle.dump(mon_model_knn,fichier_model)
```

## Chapitre 2 : Structure et gestion de la base de données (MySQL)

Dans cette partie, nous avons utilisé MySQL Workbench, un outil graphique permettant de concevoir, administrer, et gérer des bases de données. C'est à travers cet outil que nous avons d'abord créé une base de données dédiée au projet, puis défini les différentes tables nécessaires.




Une fois la base de données mise en place, nous avons créé les quatre tables suivantes :

- **Table clients** : Cette table contient les informations personnelles et financières de chaque client, comme le genre, la situation matrimoniale, les revenus mensuels, etc. Elle constitue la base des informations que nous allons relier aux autres tables.

```sql
create database if not exists roy_bank ;
use roy_bank;
create table clients (
    id_client int auto_increment primary key,
    nom varchar(100),
    prenom varchar(100),
    genre varchar(1), /* M ou F */
    marie varchar(1), /* O ou N */
    dependants int,
    auto_entrepreneur varchar(1), /* O ou N */
    education varchar(1), /* O ou N */
    zone_propriete varchar(10), /* Urban ou Ruran ou Semiurban   */
    revenu_mensuel decimal(15,2) );

```

- **Table transactions** : Chaque transaction financière est enregistrée dans cette table. Elle permet de suivre les flux d'argent entrant et sortant pour chaque client. Cette table est reliée à la table clients via une clé étrangère (id_client).

```sql
create table transactions (
id_transaction int auto_increment primary key,
id_client int,
nature varchar(10),
montant_transaction decimal(15,2) ,
date_transaction date,
foreign key (id_client) references clients(id_client)on delete cascade /* Pour éviter les lignes orheli
);
```

- **Table historique_pret** : Cette table enregistre les informations relatives aux prêts accordés ou refusés pour chaque client, telles que le montant, la durée du prêt, et le statut. Elle est également liée à la table clients par une clé étrangère.

```sql
create table historique_pret (
id_pret int auto_increment primary key,
id_client int,
montant_pret decimal(15,2) ,
duree_pret int,
date_emprunt date,
statut_pret varchar(1), /* O ou N */
foreign key (id_client) references clients(id_client) on delete cascade
);
```

- Table utilisateurs : Bien que non utilisée dans les tests fictifs, cette table est destinée à gérer les informations d'authentification des utilisateurs de la plateforme.

```sql
use roy_bank;
CREATE TABLE utilisateurs (
    id INTEGER PRIMARY KEY auto_increment,
    email varchar(100) NOT NULL UNIQUE,
    mot_de_passe varchar(100) NOT NULL
);
insert into utilisateurs( email, mot_de_passe) values
('admin@roybank.com', 'admin'), ('malick@roybank.com', 'admin');

```

Nous avons rempli les trois premières tables avec des données fictives, afin de tester la structure et le fonctionnement de la base. Ce processus a permis de s'assurer de la cohérence entre les différentes tables et de valider leur intégration avec l'application en cours de développement.

| id_client | nom | prenom | genre | marie | dependants | auto_entrepreneur | education | zone_propriete | revenu_mensuel |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Kone | Aissatou | F | O | 2 | O | N | Urban | 1200000.00 |
| 2 | Diouf | Mamadou | M | O | 3 | N | O | Semiurban | 280000.00 |
| 3 | Toure | Fatoumata | F | N | 1 | N | N | Rural | 70000.00 |
| 4 | Ndiaye | Ousmane | M | O | 4 | O | O | Urban | 950000.00 |
| 5 | Sow | Aminata | F | N | 0 | N | N | Semiurban | 160000.00 |
| 6 | Fofana | Ibrahime | M | N | 2 | N | O | Rural | 65000.00 |
| 7 | Bamba | Mariama | F | O | 1 | O | N | Urban | 900000.00 |
| 8 | Cisse | Abdoulaye | M | N | 3 | O | N | Semiurban | 570000.00 |
| 9 | Sarr | Moussa | M | O | 2 | O | N | Urban | 110000.00 |
| 10 | Ba | Ndeye | F | N | 0 | N | O | Rural | 55000.00 |
| 11 | Diallo | Issa | M | O | 5 | O | N | Urban | 1500000.00 |
| 12 | Tiana | Aissatou | F | N | 1 | N | O | Semiurban | 75000.00 |
| 13 | Gaye | Abibatou | F | O | 2 | O | N | Urban | 235000.00 |
| 14 | Thiam | Sidy | M | N | 1 | N | O | Rural | 70000.00 |
| 15 | Kane | Yacine | F | O | 3 | O | N | Semiurban | 90000.00 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

| id_transaction | id_client | nature | montant_transaction | date_transaction |
|---|---|---|---|---|
| 1 | 1 | passif | 60307.42 | 2024-06-21 |
| 3 | 1 | passif | 21963.53 | 2024-08-25 |
| 5 | 1 | passif | 416519.64 | 2024-10-19 |
| 6 | 1 | actif | 380854.43 | 2024-11-01 |
| 7 | 1 | passif | 222324.60 | 2024-03-21 |
| 8 | 2 | passif | 2899.25 | 2024-11-06 |
| 9 | 2 | actif | 229176.91 | 2024-09-04 |
| 10 | 2 | passif | 255873.44 | 2024-03-12 |
| 11 | 2 | actif | 282099.54 | 2024-07-28 |
| 12 | 2 | passif | 159381.02 | 2024-08-07 |
| 13 | 2 | actif | 133794.59 | 2024-08-10 |
| 14 | 2 | passif | 400280.92 | 2024-12-16 |
| 15 | 3 | passif | 410020.40 | 2024-06-19 |
| 16 | 3 | actif | 494047.81 | 2024-12-28 |
| 17 | 3 | passif | 453274.76 | 2024-08-10 |
| 18 | 3 | actif | 422690.53 | 2024-03-16 |
| 19 | 3 | passif | 76824.40 | 2024-01-27 |
| 20 | 3 | actif | 233308.92 | 2024-07-04 |
| 21 | 3 | passif | 99455.56 | 2024-06-25 |

Result Grid | Filter Rows: | Edit: | Export/Import:

| id_pret | id_client | montant_pret | duree_pret | date_emprunt | statut_pret |
|---|---|---|---|---|---|
| 1 | 1 | 73021.67 | 17 | 2024-04-14 | O |
| 2 | 2 | 284980.25 | 17 | 2024-01-05 | N |
| 3 | 3 | 333475.36 | 42 | 2024-04-26 | O |
| 4 | 4 | 298651.43 | 15 | 2024-09-17 | N |
| 5 | 5 | 425941.80 | 24 | 2024-12-04 | O |
| 6 | 6 | 407276.13 | 35 | 2024-03-23 | N |
| 7 | 7 | 194429.07 | 27 | 2024-08-13 | O |
| 8 | 8 | 199808.35 | 31 | 2024-12-25 | N |
| 9 | 9 | 285514.15 | 10 | 2024-03-19 | O |
| 10 | 10 | 246882.42 | 26 | 2024-03-10 | N |
| 11 | 11 | 54028.78 | 26 | 2024-05-07 | O |
| 12 | 12 | 34400.29 | 31 | 2024-12-27 | N |
| 13 | 13 | 422971.86 | 18 | 2024-04-15 | O |
| 14 | 14 | 292929.81 | 19 | 2024-03-23 | N |
| 15 | 15 | 214208.78 | 36 | 2024-05-21 | O |
| NULL | NULL | NULL | NULL | NULL | NULL |

# Chapitre 3 : Conception de l'interface utilisateur (UI) pour la plateforme

## 3.1. Introduction

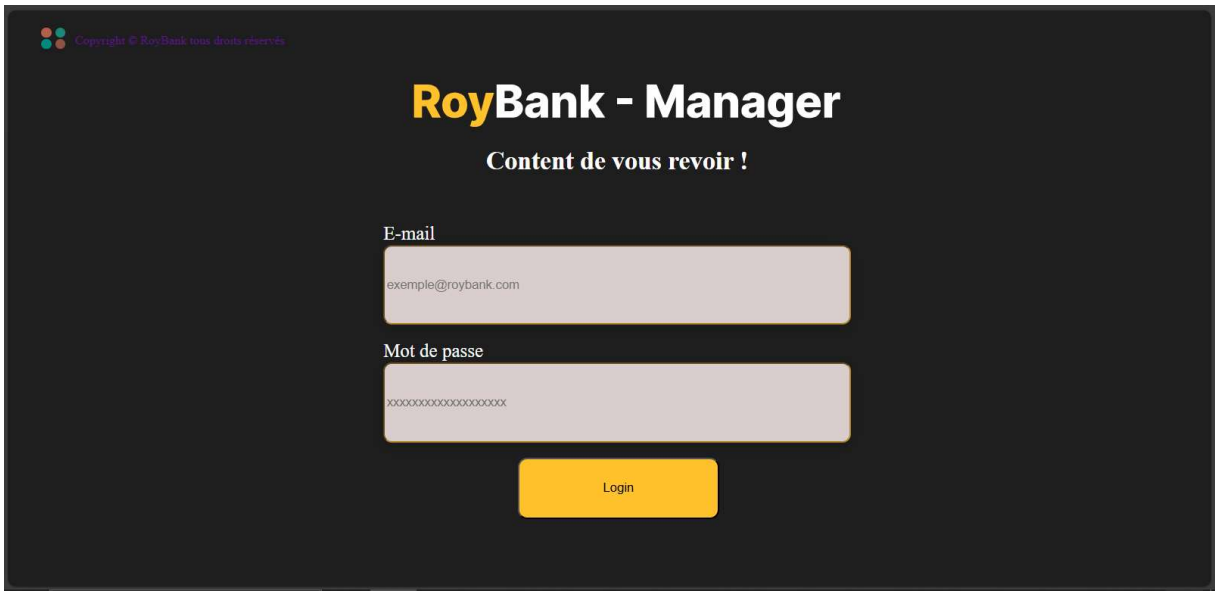
Ce chapitre traite de la conception et du développement de l'interface utilisateur de la plateforme de prédiction de prêt. L'interface a été développée en utilisant des technologies simples et efficaces, notamment HTML, CSS, JavaScript pour le frontend, et Flask pour le backend. Nous détaillerons les différentes pages de l'application, leurs fonctionnalités, et les interactions avec la base de données MySQL. Nous fournirons également un aperçu du processus d'authentification des utilisateurs et de la navigation entre les différentes pages.

## 3.2. Structure des Pages

L'application est composée de plusieurs pages interconnectées, chacune ayant un rôle bien défini. Voici une description détaillée de chaque page :

### *3.2.1. Page de Connexion*

Cette page permet aux utilisateurs (employés de la banque) de se connecter à la plateforme. Les identifiants sont vérifiés à l'aide de la fonction « verification_id() » qui interagit avec la base de données MySQL. Si l'authentification est réussie, l'utilisateur est redirigé vers la page d'accueil.

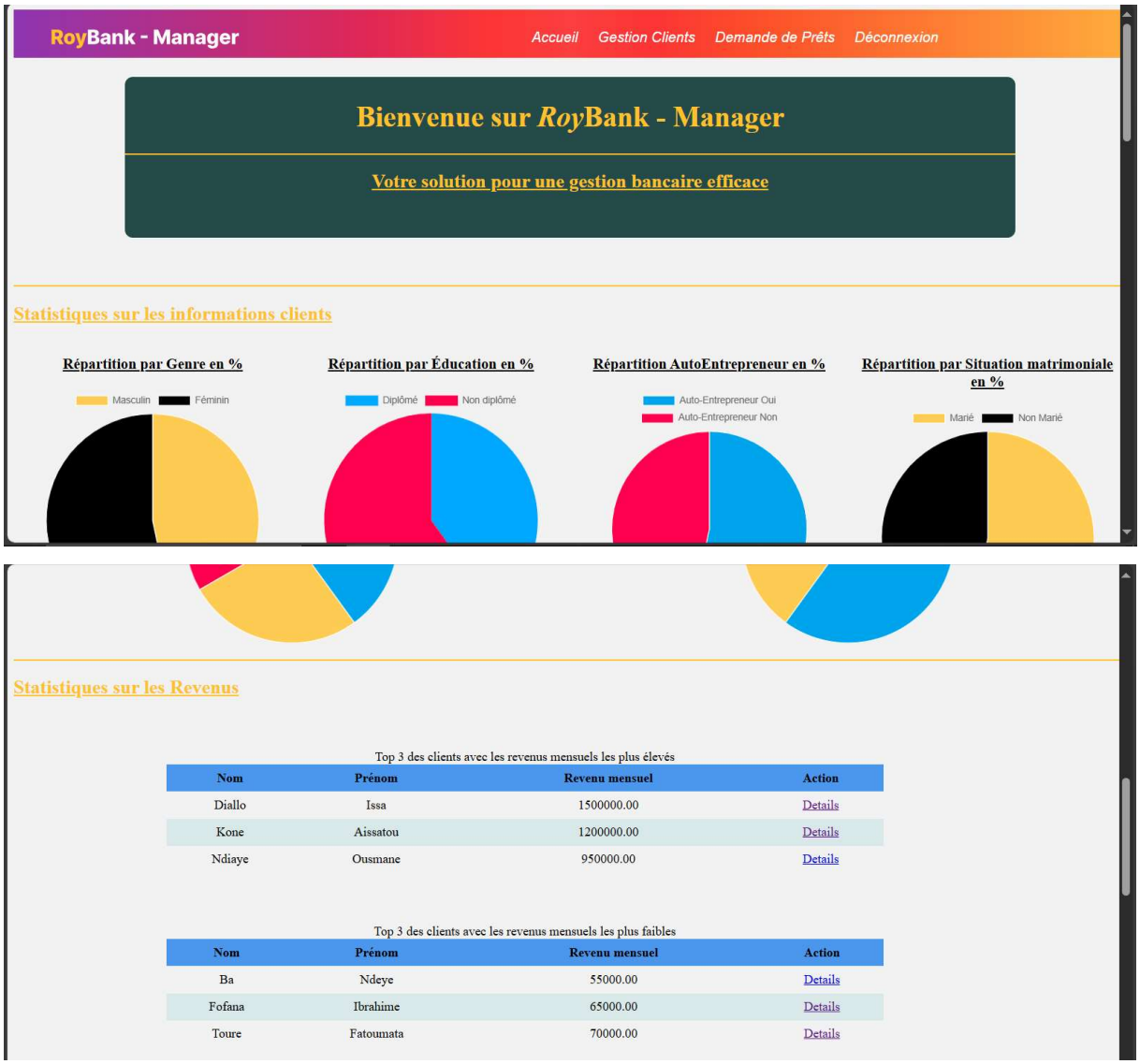



```python
# Vérification des identifiants de l'utilisateur dans la base de données
def verification_id(email, password):
    conn = connexion_db()
    cursor = conn.cursor()
    requete = "SELECT id, mot_de_passe FROM utilisateurs WHERE email = %s"
    cursor.execute(requete, (email,))
    user_data = cursor.fetchone()
    cursor.close()
    conn.close()
    if user_data:
        user_id, vrai_password = user_data
        if password == vrai_password:
            return User(user_id)
    return None

# Route pour afficher la page de connexion
@app.route('/')
def afficherPageLogin():
    return render_template('login.html')
```

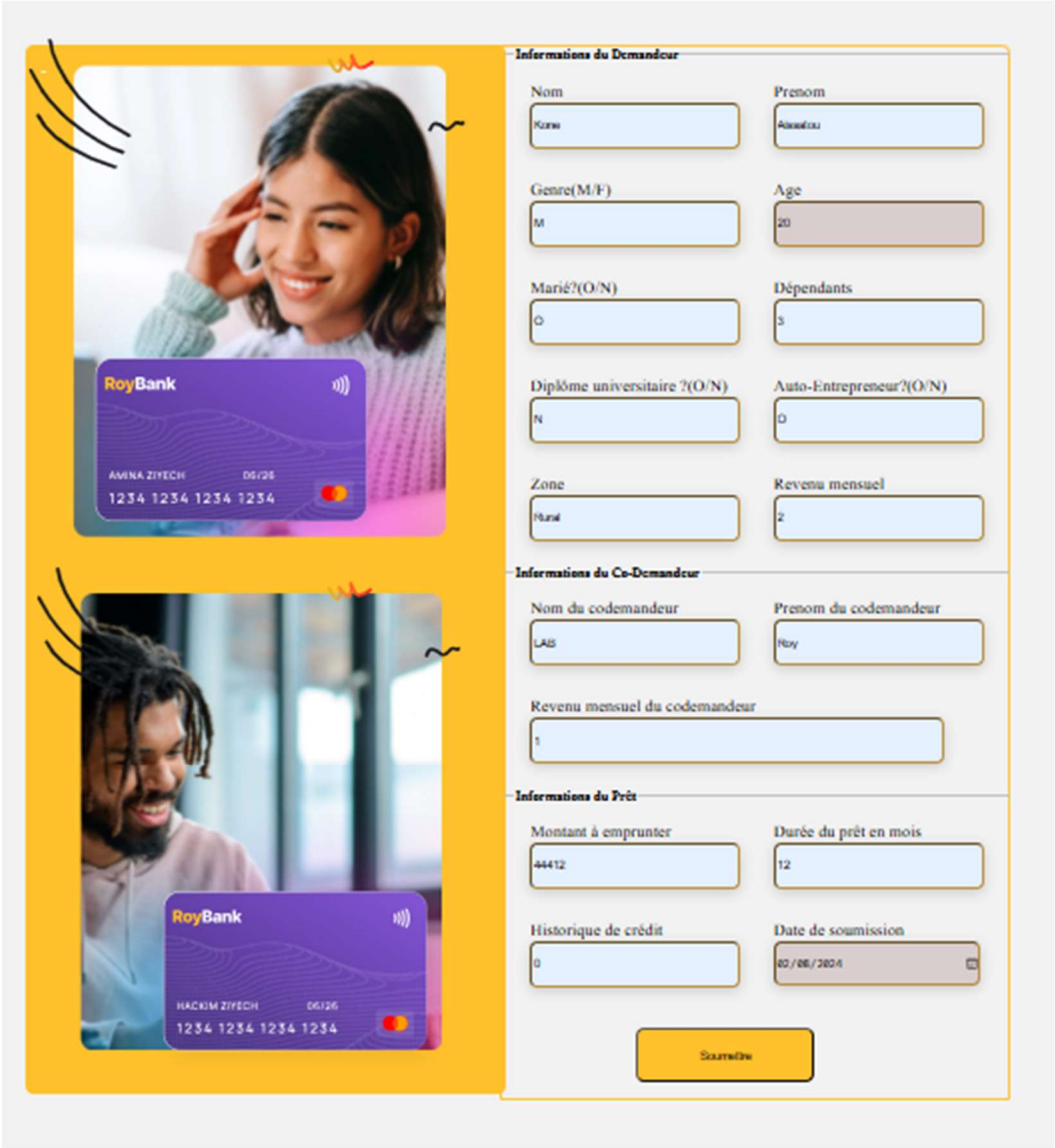
### *3.2.2. Page d'Accueil*

Une fois connecté, l'utilisateur est dirigé vers une page récapitulative contenant divers graphiques et tableaux présentant les statistiques globales de la clientèle de la banque. Les données sont récupérées en temps réel depuis la base de données MySQL et affichées sous forme de pourcentages et de listes.




### *3.2.3. Page de Demande de Prêt*

Cette page permet de soumettre une nouvelle demande de prêt. L'utilisateur entre les informations du client via un formulaire, et ces données sont ensuite utilisées pour effectuer une prédiction de l'éligibilité du prêt.




```python
# Route pour afficher la page formulaire
@app.route('/formulairepret', methods=['GET', 'POST'])
@login_required
def afficherPageFormulairePret():
    if request.method == 'POST':
        donnees_formulaire = recup_infosFormulaire()
        if donnees_formulaire:
            session['donnees_formulaire'] = donnees_formulaire
            return redirect(url_for('prediction'))  # Redirection après soumission du formu
        print("voici la session au niveau de afficherPageFormulairePret ",
        session)
    return render_template('formulairePret.html')
```

### *3.2.4. Page de Réponse à la Demande de Prêt*

Après soumission du formulaire de demande de prêt, cette page affiche le résultat de la prédiction (éligibilité ou non du prêt). Les données du formulaire sont traitées en utilisant un modèle prédictif développé avec des algorithmes comme KNN et la régression logistique.


```python
# Route pour traiter la prédiction
@app.route('/prediction', methods=['GET', 'POST'])
@login_required
def prediction():
    donnees_formulaire = session.get('donnees_formulaire', None)
    if donnees_formulaire:
        client_nom = f"{donnees_formulaire['prenom']} {donnees_formulaire['nom']}"
        montant_pret = donnees_formulaire['montant_pret']
        colonnes = ['nom', 'prenom', 'genre', 'marie', 'dependants', 'education', 'auto_ent
        df = pd.DataFrame([donnees_formulaire], columns=colonnes)
        predata = df.drop(["nom", "prenom"], axis=1)

        # Encoder les colonnes catégorielles
        colonnes_categorielles = ['genre', 'marie', 'dependants', 'education', 'auto_entrep
        for i in colonnes_categorielles:
            predata[i] = predata[i].astype("object")
            predata[i] = LabelEncoder().fit_transform(predata[i])

        # Charger le modèle et faire une prédiction
        with open("models/prediction_eligibilite_pret_modelRL_byRoyce.pkl", "rb") as modele
            modeleRL = pickle.load(modele)

        prediction = modeleRL.predict(predata)
        session['verdict'] = prediction[0]
```

### *3.2.5. Page de Gestion des Clients*

Cette page liste tous les clients enregistrés dans la base de données, avec la possibilité de cliquer sur un client pour afficher plus de détails. L'utilisateur peut ainsi consulter les informations personnelles de chaque client et leur historique de transactions.


```python
# Route pour afficher la page gestionClient (requiert une connexion)
@app.route('/gestionClient')
@login_required
def afficherPageGestionClient():
    infos_clients = recup_infosClient()
    return render_template('gestionClient.html', clients = infos_clients)
```

### *3.2.6. Page Détail Client*

Lorsqu'un utilisateur clique sur un client dans la page de gestion, il est redirigé vers une page spécifique au client. Cette page affiche des informations détaillées telles que les transactions, les demandes de prêt et leur statut.


```python
@app.route("/client/<int:id_client>")
def voirClient(id_client):
    conn = connexion_db()
    cursor = conn.cursor(dictionary=True)
    requete = "SELECT * FROM clients WHERE id_client = %s"
    cursor.execute(requete, (id_client,))
    client = cursor.fetchone()
    requete2 = """
    SELECT  t.nature, t.montant_transaction, t.date_transaction
    FROM transactions t
    JOIN clients c ON c.id_client = t.id_client
    WHERE t.id_client = %s
    ORDER BY t.date_transaction DESC;
    """
    cursor.execute(requete2, (id_client,))
    transactions = cursor.fetchall()
    cursor.close()
    conn.close()

    if client is None:
        return "Client non trouvé", 404
    print(transactions)
    return render_template('voirClient.html', client=client, transactions=transactions)
```

## *3.3. Authentification et Sécurité*

Pour protéger l'accès à l'application, un système d'authentification des utilisateurs a été mis en place avec Flask-Login. Seuls les utilisateurs authentifiés peuvent accéder aux différentes fonctionnalités de la plateforme. Chaque session utilisateur est gérée via un identifiant unique stocké dans la base de données.

## *3.4. Fonction JavaScript de Déconnexion*

Cette fonction utilise un script JavaScript pour gérer la session de l'utilisateur. Lorsqu'un utilisateur clique sur le bouton "Déconnexion", une requête est envoyée au serveur pour invalider la session en cours, puis redirige l'utilisateur vers la page de connexion.

```javascript
// deconnexion.
function deconnexion() {
    var confirmation = window.confirm("Voulez-vous vraiment vous déconnecter ?");
    if (confirmation) {
        alert("Vous êtes maintenant déconnecté !");
        window.location.href = '/';
    } else {
        alert("La déconnexion a été annulée.");
    }
}

```

## *3.5. Interaction avec la Base de Données*

Chaque page de l'application récupère les données nécessaires via des requêtes SQL exécutées en Python. Par exemple, la page d'accueil récupère les statistiques sur les clients et les prêts à l'aide de diverses requêtes SQL. La page de demande de prêt enregistre les informations soumises dans la base de données avant de rediriger l'utilisateur vers la page de prédiction.

## *3.5. Conclusion*

La conception de l'interface utilisateur a été guidée par des principes de simplicité et d'efficacité. En utilisant Flask pour le backend et HTML/CSS & JS pour le frontend, nous avons pu créer une plateforme conviviale et intuitive, tout en intégrant une base de données MySQL pour stocker et gérer les données des clients et des prêts.

# Chapitre 4 : Déploiement de l'application

## 4.1. Introduction à Docker

Docker est une plateforme qui permet de développer, expédier et exécuter des applications dans des conteneurs légers et isolés. Chaque conteneur embarque tous les composants nécessaires à l'exécution d'une application, y compris le code source, les bibliothèques et les dépendances. Docker permet donc de garantir que l'application fonctionne de la même manière, que ce soit sur l'environnement de développement ou en production, quel que soit le système d'exploitation sous-jacent.

## 4.2. Configuration de l'Environnement Docker

Pour le déploiement de l'application, plusieurs fichiers de configuration Docker ont été utilisés :

### *4.2.1 Fichier Dockerfile*

Le fichier Dockerfile est un script contenant les instructions nécessaires pour créer l'image Docker de l'application.

```
1
2   FROM python:3.9-slim
3
4   WORKDIR /app
5
6   COPY requirements.txt requirements.txt
7
8   RUN pip install -r requirements.txt
9
10  COPY . .
11
12  EXPOSE 5000
13
14  CMD ["python", "app.py"]
15
```

### *4.2.2 Fichier docker-compose.yml*

Le fichier docker-compose.yml permet de définir et de gérer plusieurs conteneurs Docker, comme celui de l'application Flask et de la base de données MySQL.

### *4.2.3 Fichier docker-compose.yml*

Le fichier docker-compose.yml permet de définir et de gérer plusieurs conteneurs Docker, comme celui de l'application Flask et de la base de données MySQL.

## 4.3. Processus de Déploiement avec Docker

Voici les étapes suivies pour déployer l'application avec Docker :

### *4.3.1 Construction de l'Image Docker*

La première étape consiste à construire l'image Docker à partir du fichier Dockerfile. Cette opération empaquette l'application ainsi que ses dépendances dans une image. Pour ce faire, la commande suivante a été exécutée dans le terminal à la racine du projet : « docker build -t roy_bank_app . »

### *4.3.2 Construction et lancement du conteneur Docker*

Une fois notre image créée, nous avons créé un conteneur à partir de cette image et l'avons lancé avec la commande : « docker run -p 5000:5000 roy_bank_app . » . Notre application est maintenant disponible via l'adresse « http://localhost:5000 »

# 3eme PARTIE : CADRE ANALYTIQUE

## 3.1. Analyse des Résultats

L'analyse des résultats obtenus à partir du modèle de prédiction et de l'application a permis d'évaluer l'efficacité de la solution mise en place. Après avoir formé les algorithmes de KNN et de RL, nous avons examiné leurs performances en termes de précision, rappel, et score F1, en utilisant des métriques standards d'évaluation des modèles de ML.

Le modèle KNN, tout en étant plus simple dans son fonctionnement, a montré des résultats satisfaisants pour certaines catégories de clients, notamment ceux ayant des transactions et des antécédents financiers bien documentés. Cependant, il est apparu que cet algorithme était sensible à la distribution des données et aux valeurs aberrantes, ce qui a parfois mené à des prédictions moins précises pour des clients avec des données partielles ou atypiques.

Quant à la régression logistique, elle a montré une meilleure performance globale, notamment en termes de précision et de rappel. Ce modèle s'est révélé plus robuste dans la classification des clients entre éligibles et non éligibles aux prêts.

## 3.2. Limites du Projet

Malgré les résultats globalement satisfaisants, plusieurs limites ont été identifiées :

- **Qualité et disponibilité des données** : La qualité du jeu de données utilisé a eu un impact direct sur la performance des modèles. La base de données était relativement limitée, avec des informations parfois manquantes ou incomplètes, ce qui a affecté les prédictions. Un accès à des données plus riches et diversifiées aurait permis de renforcer la robustesse du modèle.

- **Manque de variabilité des cas clients** : Le jeu de données a révélé une concentration de certains types de clients (par exemple, des clients urbains avec des revenus similaires), ce qui a pu biaiser les résultats. Des données plus variées concernant des clients venant de zones rurales ou semi-urbaines auraient pu améliorer la capacité du modèle à généraliser les prédictions.

- **Modèles prédictifs limités** : Le projet a utilisé des algorithmes relativement basiques (KNN et RL). Bien que ces algorithmes aient montrés de bonnes performances, des modèles plus avancés tels que les Random Forests ou les réseaux de neurones pourraient offrir des améliorations en termes de précision et de capacité de traitement de données plus complexes.

## 3.3. Perspectives d'Amélioration

Pour pallier les limites identifiées et améliorer la plateforme, plusieurs pistes peuvent être envisagées :

- **Enrichissement des données** : Une collecte de données plus diversifiée, incluant des informations sur une plus grande variété de clients et des données supplémentaires telles que les historiques de crédit ou les comportements d'épargne, permettrait d'améliorer les performances des modèles.

- **Implémentation de nouveaux modèles** : L'intégration de modèles plus sophistiqués comme les Random Forest, ou même des réseaux neuronaux permettrait d'explorer des approches non linéaires et d'augmenter la précision des prédictions. Ces modèles peuvent mieux capturer la complexité des relations entre les différentes variables.

## 3.4. Conclusion de la Partie Analytique

En résumé, l'analyse des résultats obtenus montre que les modèles développés offrent une bonne base pour prédire l'éligibilité aux prêts. Cependant, pour atteindre une solution pleinement fonctionnelle et exploitable à grande échelle, des améliorations sont nécessaires tant au niveau des données que des algorithmes et de l'infrastructure de l'application.

Ces perspectives d'amélioration constituent des pistes de développement futures pour continuer à renforcer la plateforme, en intégrant davantage de données et en optimisant la précision des prédictions pour répondre aux exigences croissantes du secteur bancaire.

# CONCLUSION

Ce rapport présente une solution de plateforme de prédiction d'éligibilité aux prêts bancaires, développée à travers trois grandes parties : le Cadre théorique, le Cadre pratique, et le Cadre analytique.

Dans la première partie, nous avons posé les bases théoriques du projet en explorant les concepts essentiels liés à la modélisation prédictive, aux systèmes d'information bancaires, ainsi qu'à l'utilisation des données clients pour la prise de décision. Nous avons également étudié les approches existantes dans le domaine des prêts bancaires et les défis posés par l'évaluation de l'éligibilité des clients, justifiant ainsi l'importance d'une solution automatisée.

La deuxième partie (Cadre pratique) a porté sur le développement effectif de la plateforme. Elle s'est articulée en trois chapitres.

Le premier chapitre était consacré à la construction du modèle de prédiction, en utilisant des algorithmes de ML tels que le KNN et la régression logistique. Nous avons décrit le processus de sélection des données, de prétraitement, et de validation des modèles.

Le deuxième chapitre s'est focalisé sur la gestion des données à travers MySQL, en définissant les tables et les relations entre les clients, les transactions, et les prêts.

Le troisième chapitre a détaillé le développement de l'interface utilisateur en HTML, CSS, JavaScript et Flask, pour fournir une expérience utilisateur intuitive et efficace.

Dans la troisième et dernière partie (Cadre analytique), nous avons analysé les résultats obtenus, les performances des modèles de prédiction, ainsi que les perspectives d'amélioration. Nous avons mis en avant les points forts de l'application, tout en soulignant les limites rencontrées, notamment la nécessité d'intégrer des données supplémentaires ou d'optimiser davantage le modèle prédictif pour une meilleure performance.

Ce projet, en plus d'apporter une solution pratique et fonctionnelle pour la gestion des prêts bancaires, ouvre des perspectives pour l'automatisation accrue des processus financiers à travers l'utilisation de l'intelligence artificielle et de la data science. Il constitue un pas important vers la modernisation des services bancaires, permettant une évaluation plus rapide, plus précise, et plus objective des profils clients.

Enfin, bien que le projet ait atteint ses objectifs initiaux, il présente des axes d'amélioration, notamment par l'ajout de nouvelles fonctionnalités et l'optimisation continue du modèle prédictif, offrant ainsi une solution évolutive pour répondre aux besoins futurs des institutions bancaires.

# WEBOGRAPHIE

**Flask Documentation** : https://flask.palletsprojects.com/

**W3Schools - HTML, CSS, JavaScript Tutorials**: https://www.w3schools.com/

**Stack Overflow**: https://stackoverflow.com/

**Sql.sh**: https://sql.sh/cours/

**Materialize CSS**: https://materializecss.com/

**Python.org** : https://www.python.org/

**GitHub**: https://github.com/

**Trello (Atlassian):** https://trello.com/

**Towardsdatascience** : https://towardsdatascience.com/predict-loan-eligibility-using-machine-learning-models-7a14ef904057

**Repo. Github** - analyse de variables : https://github.com/mridulrb/Predict-loan-eligibility-using-IBM-Watson-Studio

**Analytics vidhya**: https://www.analyticsvidhya.com/

**Statology**: https://www.statology.org/label-encoding-vs-one-hot-encoding/

**Practical python for datascience**: https://www.practicalpythonfordatascience.com/ap_seaborn_palette

**CSSgradient** : https://cssgradient.io/

**Scribbr** : https://www.scribbr.fr/methodologie/etat-de-lart/

**Datascientest**: https://datascientest.com/machine-learning-tout-savoir

**MySQL-Workbench** : https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html

**Figcomponents**: https://www.figcomponents.com/components/

**Repo. Github – méthodes de traitement valeurs manquantes** : https://github.com/matthewbrems/ODSC-missing-data-may-18/blob/master/Analysis%20with%20Missing%20Data.pdf

# ANNEXE

L'intégralité du projet est disponible sur Github. Que ce soit le code en général, les scripts, le rendu des pages…, il est possible de les consulter via le lien suivant : https://github.com/Royce-LAYINDE/project_roy_bank_manager .

# Table des matières