Université Mohammed V de Rabat
Ecole Supérieure de Technologie de Salé

RAPPORT DE PROJET DE FIN D'ÉTUDES

Préparé par : Mohammed Lahoua
Département : Informatique
Filière : Systèmes Informatiques et Réseaux

Encadré par :
Mr. Mounir Amraoui

Avril 2024

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude à Allah, le Miséricordieux, pour m'avoir accordé la force, la patience et la persévérance nécessaires pour mener à bien ce projet.

Je souhaite adresser mes plus sincères remerciements à Monsieur Amraoui pour son accompagnement, ses conseils avisés et son soutien constant tout au long de ce projet. Sa contribution a été inestimable et a grandement contribué à sa réussite.

Je voudrais également exprimer ma reconnaissance spéciale à Monsieur Badaoui pour son dévouement et son engagement tout au long des deux années de mes études, ainsi qu'à Monsieur Amraoui pour sa contribution précieuse. Leur guidance, leur enseignement et leur bienveillance ont été d'une importance capitale pour mon développement académique et professionnel.

Enfin, je tiens à remercier chaleureusement tous les autres professeurs qui ont enrichi mon parcours académique par leur expertise, leur enseignement et leur encouragement. Leur impact sur ma formation ne sera jamais oublié.

PLAN DU RAPPORT

Introduction

Présentation générale du projet

Le secteur bancaire est confronté à la tâche cruciale d'évaluer les risques associés à l'octroi de prêts. Afin d'optimiser ce processus, les banques cherchent à tirer parti des avancées en matière d'analyse de données et de modélisation prédictive. Notre projet vise ainsi à développer un modèle de machine learning capable de prédire si un demandeur de prêt est susceptible d'être approuvé ou non, en se basant sur un ensemble de caractéristiques financières et personnelles.

Contexte de l’étude

Face à une concurrence accrue sur le marché financier, les banques doivent non seulement attirer de nouveaux clients, mais aussi minimiser les risques de défaut de paiement. Dans ce contexte, la capacité à évaluer rapidement et avec précision la solvabilité des demandeurs de prêts devient un élément clé de la stratégie commerciale des institutions financières. Notre projet s'inscrit donc dans cette démarche d'optimisation des processus décisionnels bancaires.

Introduction

Objectif Principal du projet

L'objectif principal de notre projet est de concevoir, développer et évaluer un modèle prédictif robuste qui permettra à une banque de prendre des décisions éclairées concernant l'octroi de prêts. Ce modèle devra être en mesure de classer les demandeurs en fonction de leur probabilité d'acceptation ou de refus de prêt, en se basant sur des critères tels que le revenu, l'historique de crédit, l'état matrimonial, etc.

Etapes principales du projet

Notre projet se déroulera en plusieurs étapes clés, comprenant notamment :

- Collecte et nettoyage des données : Acquisition des données pertinentes et préparation de celles-ci pour l'analyse.
- Analyse exploratoire : Exploration approfondie des données pour en comprendre les tendances, les corrélations et les éventuels modèles.
- Réalisation du modèle : Développement et évaluation de plusieurs modèles de machine learning pour prédire l'éligibilité au prêt.
- Déploiement du modèle : Intégration du modèle sélectionné dans une application pratique pour une utilisation opérationnelle.

À travers ces étapes, notre objectif est de fournir aux banques un outil précieux pour améliorer leur processus de prise de décision en matière de prêts, contribuant ainsi à une gestion plus efficace des risques et à une meilleure satisfaction des clients.

Collecte et Nettoyage des Données

Définitions

Collecte de données : La collecte de données est le processus d'acquisition d'informations à partir de diverses sources, telles que des bases de données, des fichiers, des sites Web, des capteurs et des enquêtes. Elle vise à constituer un ensemble de données pertinent pour un objectif précis, comme la recherche scientifique, l'analyse commerciale ou le développement d'applications.
Nettoyage de données : Le nettoyage de données est le processus de préparation et de correction des données collectées pour les rendre utilisables pour l'analyse. Il vise à identifier et à corriger les erreurs, les incohérences et les valeurs manquantes dans les données.

Nécessité du nettoyage des données

Le nettoyage des données est une étape essentielle pour plusieurs raisons :

- Améliorer la qualité des données: Le nettoyage permet de corriger les erreurs et les incohérences, ce qui améliore la précision et la fiabilité des données.
- Augmenter l'efficacité de l'analyse: Des données propres et bien organisées facilitent l'analyse et permettent d'obtenir des résultats plus précis et plus significatifs.
- Assurer la cohérence des résultats: Le nettoyage des données garantit que les résultats de l'analyse sont cohérents et fiables.
- Éviter les biais: Des données non nettoyées peuvent contenir des biais qui peuvent influencer les résultats de l'analyse.

Collecte et Nettoyage des Données

Exemples de méthodes de nettoyage de données

- Traitement des valeurs manquantes: suppression des observations avec des valeurs manquantes, imputation des valeurs manquantes par la moyenne, la médiane ou la valeur la plus fréquente.
- Correction des erreurs: identification et correction des erreurs de saisie, de formatage ou de cohérence.
- Normalisation des données: transformation des données pour les mettre sur une échelle commune.
- Standardisation des données: conversion des données en un format standard.

Choix de la base de données

Nous avons opté pour une base de données disponible sur Kaggle pour sa taille significative, la présence de valeurs manquantes et la diversité des types de données (catégoriques et numériques). Ces caractéristiques en font une ressource idéale pour notre projet, permettant une exploration approfondie et la construction de modèles prédictifs robustes.

Analyse Exploratoire

Définition de l’analyse exploratoire

L'analyse exploratoire des données (AED) est une étape cruciale dans le processus d'analyse de données. Elle vise à explorer et à comprendre les caractéristiques d'un ensemble de données, identifier les structures et les relations cachées, et formuler des hypothèses à tester ultérieurement. L'AED est une approche itérative et visuelle qui permet de découvrir des insights précieux et de guider les analyses ultérieures.

Importance de l’AED

- Comprendre les données: L'AED permet de se familiariser avec les données, d'identifier les variables importantes et de comprendre leur distribution.
- Détecter des anomalies: L'AED peut mettre en évidence des valeurs aberrantes, des erreurs de saisie ou des incohérences dans les données.
- Formuler des hypothèses: L'AED permet de formuler des hypothèses sur les relations entre les variables et de les tester ultérieurement.
- Choisir les méthodes d'analyse: L'AED aide à choisir les méthodes d'analyse statistique les plus appropriées pour les données.

Analyse Exploratoire

Utilisation de la visualisation de données

Les visualisations de données sont un outil essentiel de l'AED. Elles permettent de représenter les données de manière graphique et intuitive, ce qui facilite leur compréhension et l'identification de patterns et de relations.

Exemples de visualisation de données

- Crédit history: Un histogramme peut être utilisé pour visualiser la distribution des scores de crédit history.
- Mariage: Un diagramme à barres peut être utilisé pour comparer le nombre de demandes de crédit acceptées/refusées pour les personnes mariées et célibataires.
- Revenu: Un nuage de points peut être utilisé pour visualiser la relation entre le revenu et le score de crédit.

Exemples de visualisation de relations entre variables

- Crédit history et revenu: Un nuage de points peut être utilisé pour visualiser la relation entre le score de crédit et le revenu. Une droite de régression peut être ajoutée pour visualiser la tendance linéaire.
- Mariage et acceptation du crédit: Un diagramme à barres peut être utilisé pour comparer le taux d'acceptation du crédit pour les personnes mariées et célibataires.

Réalisation du modèle

Définition d’une algorithme de machine learning

Un algorithme de Machine Learning (ML) est un ensemble d'instructions permettant à un ordinateur d'apprendre à partir de données et d'améliorer ses performances au fil du temps sans être explicitement programmé. Il s'agit d'un processus statistique qui permet à l'algorithme de faire des prédictions ou de prendre des décisions basées sur des données qu'il a déjà analysées.

Définition d’une division intelligente

Lorsque vous développez un modèle de science des données, vous lui apprenez à reconnaître des patterns dans vos données. Si vous utilisez la même base de données pour l'apprentissage et pour l'évaluation, le modèle risque de simplement mémoriser les données et de ne pas être capable de généraliser à de nouvelles données.
Une division intelligente de la base de données vise à garantir que la base de test est un échantillon représentatif de la base d'entraînement. Cela signifie que la base de test doit contenir des exemples de tous les types de données que le modèle est susceptible de rencontrer dans la réalité.

Méthode de division Stratified Shuffle Split

L'échantillonnage aléatoire stratifié est une méthode de division d'un ensemble de données en plusieurs sous-ensembles en suivant les étapes suivantes :

1. Stratification: Diviser l'ensemble de données en groupes (strates) en fonction de la variable cible.
2. Échantillonnage aléatoire: Dans chaque strate, prélever un échantillon aléatoire d'observations.
3. Répéter les étapes 1 et 2 pour chaque sous-ensemble (train, test, validation).

Réalisation du modèle

Algorithmes utilisées dans la conception du modèle

1. Logistic Regression (Régression logistique) :

- Type d'algorithme: Apprentissage supervisé, classification binaire
- Objectif: Prédire la probabilité d'un événement (ex: succès/échec, oui/non)
- Fonctionnement: Modélise la relation entre les variables indépendantes et la variable dépendante binaire à l'aide d'une fonction sigmoïde
- Avantages: Simple à comprendre et à interpréter, performant pour les données linéaires
- Inconvénients: Sensible aux valeurs aberrantes, moins performant pour les données non linéaires

2. KNeighborsClassifier (K plus proches voisins) :

- Type d'algorithme: Apprentissage supervisé, classification et régression
- Objectif: Prédire la classe d'un point en fonction des classes des K points les plus proches
- Fonctionnement: Ne nécessite pas d'apprentissage, utilise la distance euclidienne pour déterminer la similarité entre les points
- Avantages: Simple à comprendre et à mettre en œuvre, flexible et adaptable à différents types de données
- Inconvénients: Sensible au bruit et aux valeurs aberrantes, choix de K crucial pour la performance

Réalisation du modèle

Algorithmes utilisées dans la conception du modèle

3. DecisionTreeClassifier (Arbre de décision) :

- Type d'algorithme**:** Apprentissage supervisé, classification et régression
- Objectif: Déterminer des règles de décision pour prédire la classe d'un point
- Fonctionnement**:** Crée un arbre à partir des données en divisant l'espace en sous-ensembles
- Avantages**:** Facile à visualiser et à interpréter, robuste aux valeurs aberrantes
- Inconvénients**:** Complexité à mesure que l'arbre grandit, risque de surajustement (overfitting)

Définition de la fonction de précision

En apprentissage automatique, la fonction de précision, aussi appelée précision ou valeur prédictive positive, mesure la proportion de vrais positifs parmi les prédictions positives. En d'autres termes, elle évalue la justesse des prédictions positives.

- Une précision élevée signifie que la plupart des prédictions positives sont correctes.
- Une précision faible signifie qu'une grande partie des prédictions positives sont incorrectes.

Rôle de la précision :

- Évaluer la performance d'un modèle de ML
- Comparer différents modèles de ML
- Choisir le meilleur modèle pour un problème donné

Technologies utilisées

Base de données CSV

Un fichier CSV (Comma-Separated Values) est un format de fichier simple et largement utilisé pour stocker des données tabulaires. Chaque ligne du fichier représente une observation et chaque colonne représente une variable. Les valeurs sont séparées par des virgules.

Bibliothèques python utilisées

- Pandas: permet de charger la base de données CSV, de la nettoyer et de la manipuler.
- Matplotlib et Seaborn: permettent de créer des visualisations de données pour explorer les données et comprendre les relations entre les variables.
- NumPy: permet d'effectuer des calculs scientifiques sur les données.
- Pickle: permet de sérialiser le modèle de Machine Learning pour le stocker dans un fichier PKL.
- Scikit-learn: permet de créer et d'évaluer les modèles de Machine Learning.
- Flask: permet de créer une application web pour déployer le modèle de Machine Learning.

Détailles de la réalisation

Collecte et nettoyage des données

1. Source des données:

- Pour ce projet, j'ai utilisé une base de données CSV provenant de Kaggle.
- La base de données contient des valeurs manquantes (voir capture d'écran).

LP001052	Male	Yes	1	Graduate		3717	2925	151	360		Semiurba	N
LP001066	Male	Yes	0	Graduate	Yes	9560	0	191	360	1	Semiurba	Y
LP001068	Male	Yes	0	Graduate	No	2799	2253	122	360	1	Semiurba	Y
LP001073	Male	Yes	2	Not Gradu	No	4226	1040		360	1	Urban	Y
LP001086	Male	No	0	Not Gradu	No	1442	0	35	360	1	Urban	N
LP001087	Female	No	2	Graduate		3750		120	360	1	Semiurba	Y
LP001091	Male		1	Graduate		4166	3369	201	360		Urban	N
LP001095	Male	No	0	Graduate	No	3167	0	74	360	1	Urban	N
LP001097	Male	No	1	Graduate	Yes	4692	0	106	360	1	Rural	N
LP001098	Male	Yes	0		No	3500	1667	114	360	1	Semiurba	Y
LP001100	Male	No	3+	Graduate	No	12500	3000	320	360	1	Rural	N
LP001106	Male	Yes	0	Graduate	No	2275	2067		360	1	Urban	Y
LP001109	Male	Yes	0	Graduate	No	1828	1330	100		0	Urban	N

2. Vérification des valeurs manquantes:

- La fonction info() permet de vérifier la présence de valeurs manquantes dans chaque colonne (voir capture d'écran).
- La commande df.isnull().sum().any() retourne True si au moins une valeur est manquante.
- il faut aussi déterminer si il y a une répetition dans la colonne des identifiants et si il y a une valeur negative dans les variables numériques.

```
#lire la base de données
df=pd.read_csv('train_u6lujuX_CVtuZ9i.csv')

#déterminer la présence des valeurs manquantes
df.info()
print(f"le présence des valeurs manquentes dans cette base est :{df.isnull().sum().any()}")
```

Détailles de la réalisation

Collecte et nettoyage des données

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
le présence des valeurs manquentes dans cette base est :True
```

```python
#voir les variables catégoriques et déterminer c'est un ID est répété
print(df.describe(include='object'))

#voir les variables numériques et déterminer la présence des valeurs négatives
print(df.describe(include='number'))
```

Détailles de la réalisation

Collecte et nettoyage des données

```
les variables catégoriques :
        Loan_ID Gender Married Dependents Education Self_Employed Property_Area Loan_Status
count       614    601     611        599       614           582           614         614
unique      614      2       2          4         2             2             3           2
top     LP001002   Male     Yes          0  Graduate            No     Semiurban           Y
freq           1    489     398        345       480           500           233         422
les variables  numériques :
       ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History
count       614.000000         614.000000  592.000000         600.00000      564.000000
mean       5403.459283        1621.245798  146.412162         342.00000        0.842199
std        6109.041673        2926.248369   85.587325          65.12041        0.364878
min         150.000000           0.000000    9.000000          12.00000        0.000000
25%        2877.500000           0.000000  100.000000         360.00000        1.000000
50%        3812.500000        1188.500000  128.000000         360.00000        1.000000
75%        5795.000000        2297.250000  168.000000         360.00000        1.000000
max       81000.000000       41667.000000  700.000000         480.00000        1.000000
```

3. Traitement des valeurs manquantes:

- Séparation des variables:

J'ai séparé les données en deux catégories : variables catégoriques (ex: sexe, état matrimonial) et variables numériques (ex: revenu, montant du prêt).

```python
#séparation des variables catégoriques et numériques en des listes
cat_data=[]
num_data=[]
for i,c in enumerate(df.dtypes):
    if c==object:
        cat_data.append(df.iloc[:,i])
    else:
        num_data.append(df.iloc[:,i])

#transformer les listes en format de base de données
cat_data=pd.DataFrame(cat_data).transpose()
print(cat_data)
num_data=pd.DataFrame(num_data).transpose()
print(num_data)
```

Détailles de la réalisation

Collecte et nettoyage des données

```
      Loan_ID  Gender Married Dependents     Education Self_Employed Property_Area Loan_Status
0    LP001002    Male      No          0      Graduate            No         Urban           Y
1    LP001003    Male     Yes          1      Graduate            No         Rural           N
2    LP001005    Male     Yes          0      Graduate           Yes         Urban           Y
3    LP001006    Male     Yes          0  Not Graduate            No         Urban           Y
4    LP001008    Male      No          0      Graduate            No         Urban           Y
..        ...     ...     ...        ...          ...           ...           ...         ...
609  LP002978  Female      No          0      Graduate            No         Rural           Y
610  LP002979    Male     Yes         3+      Graduate            No         Rural           Y
611  LP002983    Male     Yes          1      Graduate            No         Urban           Y
612  LP002984    Male     Yes          2      Graduate            No         Urban           Y
613  LP002990  Female      No          0      Graduate           Yes     Semiurban           N

[614 rows x 8 columns]
     ApplicantIncome  CoapplicantIncome  LoanAmount  Loan_Amount_Term  Credit_History
0             5849.0                0.0         NaN             360.0             1.0
1             4583.0             1508.0       128.0             360.0             1.0
2             3000.0                0.0        66.0             360.0             1.0
3             2583.0             2358.0       120.0             360.0             1.0
4             6000.0                0.0       141.0             360.0             1.0
..               ...                ...         ...               ...             ...
609           2900.0                0.0        71.0             360.0             1.0
610           4106.0                0.0        40.0             180.0             1.0
611           8072.0              240.0       253.0             360.0             1.0
612           7583.0                0.0       187.0             360.0             1.0
613           4583.0                0.0       133.0             360.0             0.0

[614 rows x 5 columns]
```

- Remplacement des valeurs manquantes:

Pour les variables catégoriques, j'ai remplacé les valeurs manquantes par la valeur la plus fréquente en utilisant la fonction lambda.
Pour les variables numériques, j'ai remplacé les valeurs manquantes par la valeur suivante en utilisant la fonction bfill.
J'ai vérifié la présence de valeurs manquantes avant et après le remplacement pour m'assurer que le traitement a été effectué correctement.

Détailles de la réalisation

Collecte et nettoyage des données

```
print(f"le présence des valeurs catégoriques manquantes avant la renseignement est :{cat_data.isnull().sum().any()}")
print(f"le présence des valeurs numériques manquantes avant la renseignement est : {num_data.isnull().sum().any()} ")

#renseignement des valeurs catégoriques manquantes
print("rensignement des valeurs catégoriques manquantes :")
cat_data=cat_data.apply(lambda x:x.fillna(x.value_counts().index[0]))
print("renseignement des valeurs catégoriques manquantes avec succes.")
#renseignement des valeurs numériques manquantes
print("rensignement des valeurs numériques manquantes :")
num_data = num_data.bfill()
print("renseignement des valeurs numériques manquantes avec succes.")

print(f"le présence des valeurs catégoriques manquantes après la renseignement est :{cat_data.isnull().sum().any()}")
print(f"le présence des valeurs numériques manquantes après la renseignement est : {num_data.isnull().sum().any()} ")
```

```
ADMIN@Octa-Laptop MINGW64 ~/Desktop/EST-SIR/PFE-Data Science
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-Data Science/screen.py"
le présence des valeurs catégoriques manquantes avant la renseignement est :True
le présence des valeurs numériques manquantes avant la renseignement est : True
rensignement des valeurs catégoriques manquantes :
renseignement des valeurs catégoriques manquantes avec succes.
rensignement des valeurs numériques manquantes :
renseignement des valeurs numériques manquantes avec succes.
le présence des valeurs catégoriques manquantes après la renseignement est :False
le présence des valeurs numériques manquantes après la renseignement est : False
```

4. Transformation des variables catégoriques:

- J'ai transformé les variables catégoriques en valeurs numériques pour faciliter leur utilisation dans les modèles de Machine Learning.
- J'ai séparé la colonne Loan_Status de la base de données pour l'utiliser dans l'analyse exploratoire.

```
# Remplacer la colonne Loan_Status de Y et N a 1 et 0
target_value={'Y' :1, 'N' :0}
target=cat_data['Loan_Status']
cat_data.drop('Loan_Status',axis=1, inplace=True)
target=target.map(target_value)
print(target)
```

Détailles de la réalisation

Collecte et nettoyage des données

```
ADMIN@Octa-Laptop MINGW64 ~/Desktop/EST-SIR/PFE-Data Science
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-Data Science/screen.py"
0      1
1      0
2      1
3      1
4      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 614, dtype: int64
```

```python
#encodage des variables catégoriques
le=LabelEncoder()
for i in cat_data:
    cat_data[i]=le.fit_transform(cat_data[i])
#supprimer loan_id
cat_data.drop('Loan_ID',axis=1,inplace=True)
print(cat_data)
```

```
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-Data Science/screen.py"
     Gender  Married  Dependents  Education  Self_Employed  Property_Area
0         1        0           0          0              0              2
1         1        1           1          0              0              0
2         1        1           0          0              1              2
3         1        1           0          1              0              2
4         1        0           0          0              0              2
..      ...      ...         ...        ...            ...            ...
609       0        0           0          0              0              0
610       1        1           3          0              0              0
611       1        1           1          0              0              2
612       1        1           2          0              0              2
613       0        0           0          0              1              1

[614 rows x 6 columns]
```

Détailles de la réalisation

Collecte et nettoyage des données

5. Concaténation des variables:

- J'ai concaténé les deux listes de variables (catégoriques et numériques) ainsi que la colonne Loan_Status pour obtenir la base de données finale, qu'on va utiliser pour l'analyse exploratoire et l'entraînement de notre modèle.
- J'ai aussi vérifié la présence des valeurs manquantes.

```
#concatenation des variables dans une seule base
X=pd.concat([cat_data,num_data,target],axis=1)
#verification de la présence des valeurs manquantes
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             614 non-null    int32  
 1   Married            614 non-null    int32  
 2   Dependents         614 non-null    int32  
 3   Education          614 non-null    int32  
 4   Self_Employed      614 non-null    int32  
 5   Property_Area      614 non-null    int32  
 6   ApplicantIncome    614 non-null    float64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    float64
 9   Loan_Amount_Term   614 non-null    float64
 10  Credit_History     614 non-null    float64
 11  Loan_Status        614 non-null    int64  
dtypes: float64(5), int32(6), int64(1)
memory usage: 43.3 KB
```

Détailles de la réalisation

Analyse Exploratoire

1. Déterminer la variable cible:

- La première étape de l'analyse exploratoire est de déterminer la variable que nous voulons analyser. Dans ce cas, la variable cible est Loan_Status (accepté/refusé).

2. Visualiser la distribution de la variable cible:

- On peut utiliser la fonction value_counts pour afficher le nombre de crédits acceptés et refusés.
- On peut ensuite visualiser la distribution de la variable cible à l'aide d'un diagramme à barres.

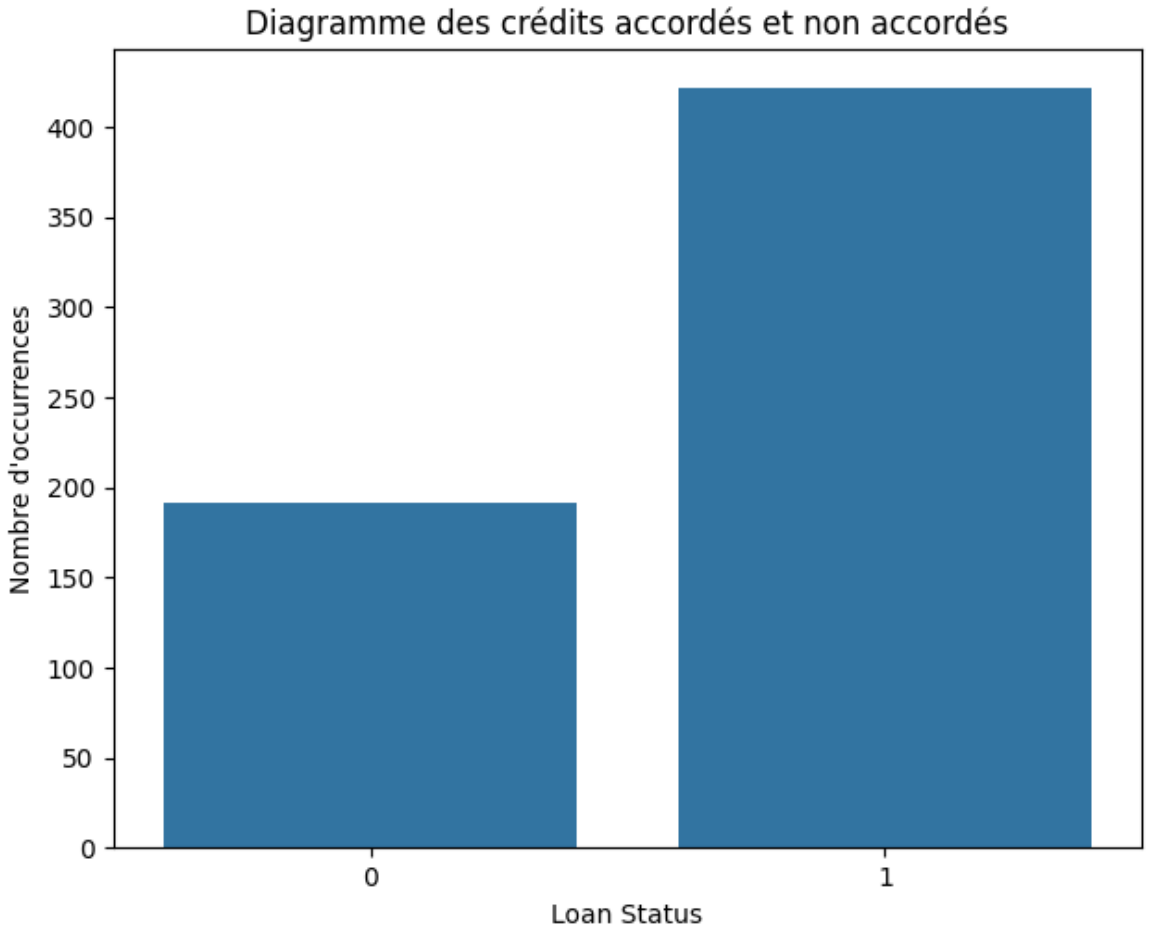
```
35  #visualisation de la variable target 'Loan_Status'
36  plt.figure(figsize=(8,6))
37  sns.countplot(x='Loan_Status', data=X)
38  plt.title("Diagramme des crédits accordés et non accordés")
39  plt.xlabel("Loan Status")
40  plt.ylabel("Nombre d'occurrences")
41  yes=(target.value_counts()[1]/len(target))*100
42  no=(target.value_counts()[0]/len(target))*100
43  print(f'le pourcentage des crédits accordés et:{yes}')
44  print(f'le pourcentage des crédits non accordés et:{no}')
45  plt.show()
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR

ADMIN@Octa-Laptop MINGW64 ~/Desktop/EST-SIR/PFE-Data Science
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-Data Science/screen.py"
le pourcentage des crédits accordés et:68.72964169381108
le pourcentage des crédits non accordés et:31.27035830618892
```

Détailles de la réalisation

Analyse Exploratoire



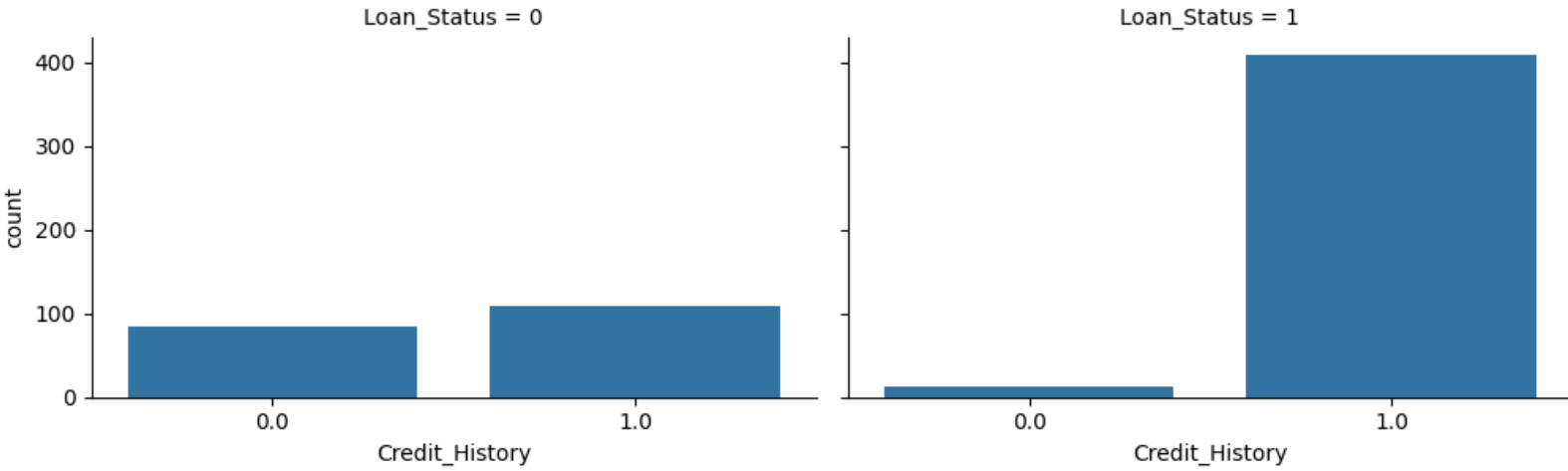
3. Analyser l'impact des variables individuelles:

- On peut ensuite analyser l'impact de chaque variable individuelle sur la variable cible.
- Par exemple, on peut visualiser la relation entre la variable Credit_History et la variable Loan_Status.
- On remarque que la majorité des personnes ayant un historique de crédit ont obtenu leur prêt, tandis que la majorité des personnes sans historique de crédit ont été refusées.

Détailles de la réalisation

Analyse Exploratoire

```
# Diagramme Comparaison Pour l'historique de credit
grid=sns.FacetGrid(X,col='Loan_Status',height=3.2,aspect=1.6)
grid.map(sns.countplot,'Credit_History')
plt.show()
```



4. Utilisation de la médiane:

- La médiane peut être utilisée pour comparer les valeurs numériques entre différentes catégories.
- Par exemple, on peut comparer la médiane du revenu des personnes dont le prêt a été accepté et la médiane du revenu des personnes dont le prêt a été refusé.

```
# les medians des valeurs
median=X.groupby('Loan_Status').median()
print(median)
```

```
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-Data Science/screen.py"
             Gender  Married  Dependents  Education  Self_Employed  Property_Area  ApplicantIncome  CoapplicantIncome  Loan
Loan_Status
0               1.0      1.0         0.0        0.0            0.0            1.0           3833.5              268.0
1               1.0      1.0         0.0        0.0            0.0            1.0           3812.5             1239.5
```

Détailles de la réalisation

Réalisation du modèle

1. Division de la base de données:

- Avant de développer le modèle, il est important de diviser la base de données en deux ensembles : un ensemble d'entraînement et un ensemble de test.
- Cette division permet d'évaluer la performance du modèle sur des données qu'il n'a pas encore vues.
- On utilise la fonction StratifiedShuffleSplit pour réaliser une division intelligente qui préserve la proportion de chaque classe dans les deux ensembles.

```
#séparation de la base de données en unes base de test et une base d'entrainement
y=target # ici target c'est la colonne de Loan_Status avec 1,0
sss = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train,test in sss.split(X,y):
    X_train,X_test=X.iloc[train],X.iloc[test]
    y_train,y_test=y.iloc[train],y.iloc[test]
print('X_train taille:', X_train.shape)
print('X_test taille:', X_test.shape)
print('Y_train taille:', y_train.shape)
print('Y_test taille:', y_test.shape)
```

```
ADMIN@Octa-Laptop MINGW64 ~/Desktop/EST-SIR/PFE-Data Science
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-
X_train taille: (491, 12)
X_test taille: (123, 12)
Y_train taille: (491,)
Y_test taille: (123,)
```

Détailles de la réalisation

Réalisation du modèle

2. Choix de l'algorithme de Machine Learning:

- Nous avons choisi trois algorithmes de Machine Learning :
  - Régression logistique: pour prédire la probabilité qu'un prêt soit accepté.
  - KNeighborsClassifier: pour prédire la classe d'un prêt en fonction des k observations les plus proches.
  - DecisionTreeClassifier: pour créer un arbre de décision pour prédire la classe d'un prêt.

3. Définition des fonctions utilisées :

- La précision (accuracy) est une mesure de performance d'un modèle de Machine Learning.
- Elle est définie comme le nombre de prédictions correctes divisé par le nombre total de prédictions.
- Nous avons défini une fonction qui permet d'entraîner, de tester et d'évaluer les modèles de Machine Learning.
- Cette fonction prend en entrée les données, l'algorithme de Machine Learning et la fonction de précision.

4. Entraînement et évaluation des modèles:

- Nous avons utilisé la deuxième fonction définie à l'étape 3 pour entraîner et évaluer les trois modèles de Machine Learning.
- Nous avons remarqué que le modèle de régression logistique a la meilleure performance avec une précision de 100%.

Détailles de la réalisation

Réalisation du modèle

```python
    #On va appliquer 3 algorithmes Logistic Regression, KNeighborsClassifier, DecisionTreeClassifier
models = {
    'LogisticRegression': LogisticRegression(random_state=42),
    'KNeighborsClassifier': KNeighborsClassifier(),
    'DecisionTreeClassifier': DecisionTreeClassifier(max_depth=1, random_state=42)
}
#la fonction de precision
def accu(y_true, y_pred, retu=False):
    acc=accuracy_score(y_true,y_pred)
    if retu:
        return acc
    else:
        print(f'la precision du modèle est: {acc}')
#Fonction pour entraîner, tester et évaluer les modèles
def train_test_eval(models, X_train, y_train, X_test, y_test):
    for name, model in models.items():
        print(name, ':')
        model.fit(X_train, y_train)
        accu(y_test, model.predict(X_test))
        print('-' * 30)
# Utilisation de la fonction avec vos données
train_test_eval(models, X_train, y_train, X_test, y_test)
```

```
la precision du modèle est: 1.0
------------------------------
KNeighborsClassifier :
la precision du modèle est: 0.6504065040650406
------------------------------
DecisionTreeClassifier :
la precision du modèle est: 1.0
------------------------------
```

Détailles de la réalisation

Réalisation du modèle

5. Réduction du nombre de variables:

- Avant de déployer le modèle, il est important de réduire le nombre de variables pour simplifier le processus de décision.
- Nous avons créé une base de données dérivée avec seulement 3 variables qui sont les plus importantes pour la prédiction du statut du prêt.
- Nous avons remarqué que le modèle de régression logistique a la meilleure performance encore une fois avec une précision de 85%.

```
#2eme BDD à partir de la première mais avec 4 colonnes
X_2 = X[['Credit_History', 'Married', 'CoapplicantIncome']]
sss = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train, test in sss.split(X_2, y):
    X_train, X_test = X_2.iloc[train], X_2.iloc[test]
    y_train, y_test = y.iloc[train], y.iloc[test]

print('X_train taille: ', X_train.shape)
print('X_test taille: ', X_test.shape)
print('y_train taille: ', y_train.shape)
print('y_test taille: ', y_test.shape)

train_test_eval(models, X_train, y_train, X_test, y_test)
```

Détailles de la réalisation

Réalisation du modèle

```
ADMIN@Octa-Laptop MINGW64 ~/Desktop/EST-SIR/PFE-Data Science
$ C:/Python312/python.exe "c:/Users/ADMIN/Desktop/EST-SIR/PFE-Data Science/screen.py"
X_train taille:  (491, 3)
X_test taille:  (123, 3)
y_train taille:  (491,)
y_test taille:  (123,)
LogisticRegression :
la precision du modèle est: 0.85365853658536585
------------------------------
KNeighborsClassifier :
la precision du modèle est: 0.6991869918699187
------------------------------
DecisionTreeClassifier :
la precision du modèle est: 0.8455284552845529
------------------------------
```

Détailles de la réalisation

Déploiement du modèle

1. Application de la régression logistique:

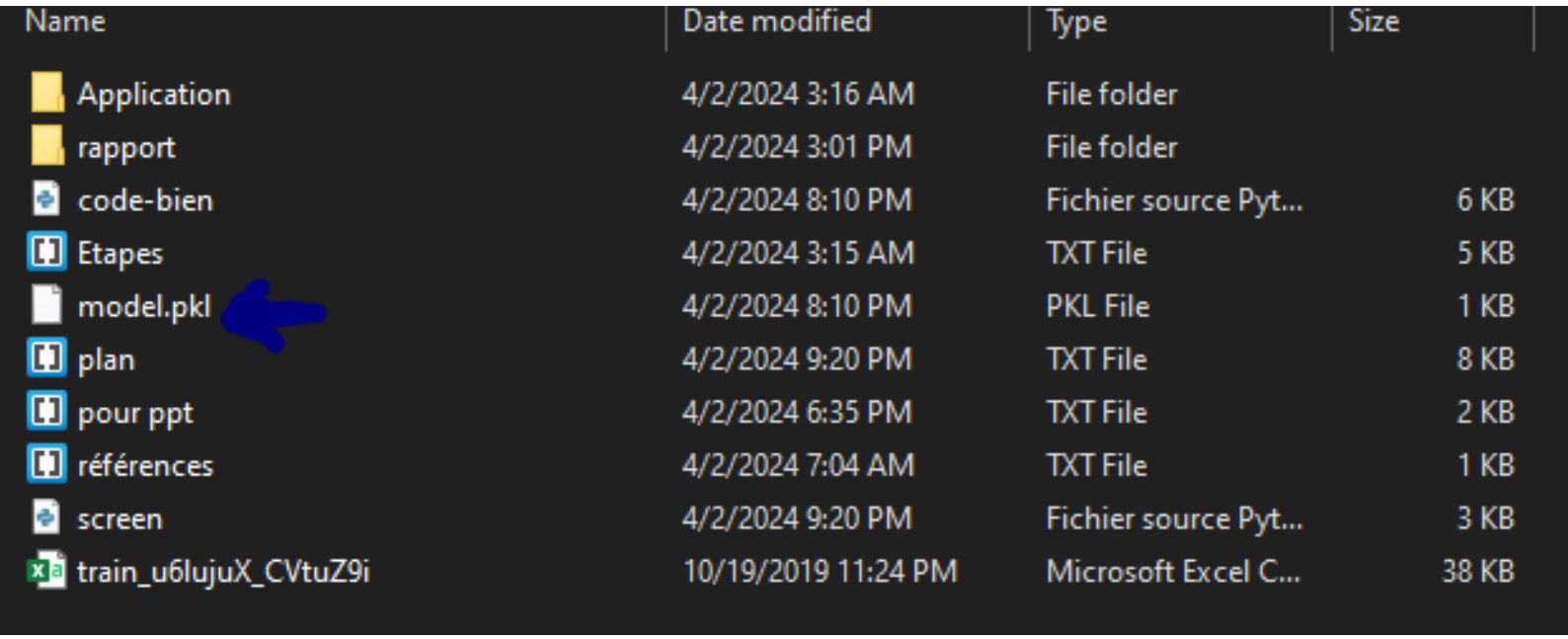
- Nous avons appliqué la régression logistique sur la deuxième base de données avec un nombre réduit de variables.
- Le modèle de régression logistique a la meilleure performance avec une précision de 85%.

2. Enregistrement du modèle:

- Nous avons enregistré le modèle sur un fichier PKL pour pouvoir le déployer.

```
#appliquer la regression logistique sur le 2ème base de données
Classifier=LogisticRegression()
Classifier.fit(X_2,y)

#enregistrer le modèle sur un fichier pkl
pickle.dump(Classifier,open('model.pkl','wb'))
```



Name	Date modified	Type	Size
Application	4/2/2024 3:16 AM	File folder	
rapport	4/2/2024 3:01 PM	File folder	
code-bien	4/2/2024 8:10 PM	Fichier source Pyt...	6 KB
Etapes	4/2/2024 3:15 AM	TXT File	5 KB
model.pkl	4/2/2024 8:10 PM	PKL File	1 KB
plan	4/2/2024 9:20 PM	TXT File	8 KB
pour ppt	4/2/2024 6:35 PM	TXT File	2 KB
références	4/2/2024 7:04 AM	TXT File	1 KB
screen	4/2/2024 9:20 PM	Fichier source Pyt...	3 KB
train_u6lujuX_CVtuZ9i	10/19/2019 11:24 PM	Microsoft Excel C...	38 KB

Détailles de la réalisation

Déploiement du modèle

3. Création de l'application:

- Nous avons créé une application web avec Python Flask, HTML, CSS et JavaScript.
- L'application permet à l'utilisateur de saisir les valeurs des variables et de recevoir une prédiction du statut du prêt.

Conclusion

Récapitulation des principales conclusions du projet:

- **Collecte et nettoyage des données:**
  - La base de données Kaggle a été collectée et nettoyée.
  - Les valeurs manquantes ont été remplacées par des valeurs appropriées.
  - Les variables catégoriques ont été transformées en valeurs numériques.
- **Analyse exploratoire:**
  - La variable cible Loan_Status a été analysée.
  - L'impact de chaque variable individuelle sur la variable cible a été étudié.
- **Réalisation du modèle:**
  - Trois modèles de Machine Learning ont été entraînés et évalués.
  - Le modèle de régression logistique a la meilleure performance avec une précision de 85%.
- **Déploiement du modèle:**
  - Le modèle de régression logistique a été déployé sur une application web.
  - L'application permet à l'utilisateur de saisir les valeurs des variables et de recevoir une prédiction du statut du prêt.

Conclusion

Importance des résultats obtenus pour la banque:

- **Amélioration de la prise de décision:**
  - Le modèle peut aider la banque à prendre des décisions plus précises concernant l'octroi de prêts.
  - Le modèle peut aider à réduire le risque de défaut de crédit.
- **Augmentation de l'efficacité:**
  - Le modèle peut automatiser le processus d'évaluation des demandes de prêt.
  - Le modèle peut aider à réduire le temps et les ressources nécessaires pour traiter les demandes de prêt.
- **Amélioration de la satisfaction client:**
  - Le modèle peut fournir aux clients une réponse plus rapide à leurs demandes de prêt.
  - Le modèle peut améliorer l'expérience client en simplifiant le processus de demande de prêt.

En conclusion, ce projet a permis de développer un modèle de Machine Learning qui peut être utilisé par la banque pour améliorer la prise de décision, l'efficacité et la satisfaction client.

Références

**source de la base de données utilisée :
https://www.kaggle.com/datasets

**data cleaning :
https://www.ibm.com/topics/data-quality
https://en.wikipedia.org/wiki/Data_cleaning

**analyse exploratoire :
https://www.ibm.com/topics/exploratory-data-analysis
https://en.wikipedia.org/wiki/Exploratory_data_analysis

**algorithmes machine learning :
https://en.wikipedia.org/wiki/Logistic_regression
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm
https://en.wikipedia.org/wiki/Decision_tree_learning
https://fr.wikipedia.org/wiki/Apprentissage_automatique
https://www.datacamp.com/courses/sampling-in-python

**biblithéques python :
pandas: https://pandas.pydata.org/docs/
matplotlib.pyplot: https://matplotlib.org/
seaborn: https://seaborn.pydata.org/
pickle: https://docs.python.org/3/library/pickle.html
sklearn: https://scikit-learn.org/stable/
numpy: https://numpy.org/doc/stable/