INPT

المعهد الوطني للبريد والمواصلات

ⵍⵎⵄⵀⴰⴷ ⵍⵡⴰⵟⴰⵏⵉ ⵏ ⵜⵓⵙⵙⵉⵔⵉⵏ ⴰⴷ ⵜⵉⵍⵉⴽⵓⵎⵓⵏⵉⴽⴰⵙⵢⵓⵏ

Institut National des Postes et Télécommunications

Projet Scoring - Filière AMOA INE2

Présenté à :

INSTITUT NATIONAL DES POSTES ET TELECOMMUNICATIONS

Sujet

Prévision des difficultés financières des clients d'une banque

---

Réalisé par :

Achraf KHAMRI
Khalil LAMHARCHI
Ziyad GHANEME
Abdessamad ABID
Walid ELASS

Encadré par :

Pr. Zineb EL AKKAOUI

Année universitaire :

2020/2021

Résumé

Dans le cadre de nos études au sein de l'INPT en deuxième année, nous - étudiants de la filière AMOA - étions amenés à réaliser un projet où nous devrions employer l'ensemble de nos acquis.

Ce projet s'inscrit dans le thème de l'analyse de données avancée et le Scoring. Divisée en plusieurs groupes, la filière AMOA était amenée à travailler afin de proposer une solution adéquate au sujet. Ce document contient les résultats du travail de notre groupe.

Durant la réalisation de ce projet, nous avons parcouru toutes les phases de le l'anaylse de données a savoir le prétraitement, la création de modèle de prévision et l'évaluation de la performance en s'appuyant sur diverses technologies, principalement Python.

Liste des abréviations

— **AMOA** - Assistance à maîtrise d'ouvrage
— **ACP** - Analyse des Composantes Principales
— **LDA** - Linear Discriminant Analysis
— **QDA** - Quadratic Discriminant analysis
— **SQL** - Structured Query Language
— **TPR** - True Positive Rate
— **FPR** - False Positive Rate
— **TNR** - True Negative Rate
— **FNR** - False Negative Rate
— **ROC** - Receiver Operating Characteristic

Liste des figures

Table des matières

1 Contexte du projet

1.1 Introduction

Le scoring est une technique qui permet d'affecter un score à un client ou prospect. Le score obtenu traduit généralement la probabilité qu'un individu réponde à une sollicitation marketing ou appartienne à la cible recherchée. Il mesure donc l'appétence pour l'offre potentielle. Le score peut également exprimer la valeur potentielle d'un client ou prospect ou l'espérance mathématique de gain qui peut lui être associée, voire un risque dans le cadre du scoring bancaire.

1.2 Cadre du projet

1.2.1 Présentation de l'éxistant

Nous avons à notre disposition une base de données regroupant les informations suivantes sur 150.000 emprunteurs :

- **SeriousDlqin2yrs**
  La personne était en désordre de paiement de 90 jours ou pire.
- **RevolvingUtilizationOfUnsecuredLines**
  Solde total sur les cartes de crédit et les marges de crédit personnelles à l'exception de l'immobilier et sans dette à tempérament comme les prêts automobiles divisé par la somme des limites de crédit.
- **Age**
  L'âge de l'emprunteur en années.
- **NumberOfTime30-59DaysPastDueNotWorse**
  Nombre de fois où l'emprunteur est en souffrance depuis 30 à 59 jours, mais pas pire au cours des 2 dernières années.
- **DebtRatio**
  Paiements mensuels de la dette, pension alimentaire, frais de subsistance divisés par le revenu brut mensuel.
- **MonthlyIncome**
  Revenu mensuel.
- **NumberOfOpenCreditLinesAndLoans**
  Nombre de prêts ouverts (à tempérament comme prêt automobile ou hypothécaire) et marges de crédit (par exemple, cartes de crédit).
- **NumberOfTimes90DaysLate**
  Nombre de fois où l'emprunteur est en souffrance depuis 90 jours au cours des 2 dernières années.
- **NumberRealEstateLoansOrLines**
  Nombre de prêts hypothécaires et immobiliers, y compris les marges de crédit hypothécaire.
- **NumberOfTime60-89DaysPastDueNotWorse**
  Nombre de fois où l'emprunteur est en souffrance depuis 60 à 89 jours, mais pas pire au cours des 2 dernières années.
- **NumberOfDependents**
  Nombre de personnes à charge dans la famille excluant l'emprunteur (conjoint, enfants, etc.)

Voici un aperçu des données en Excel

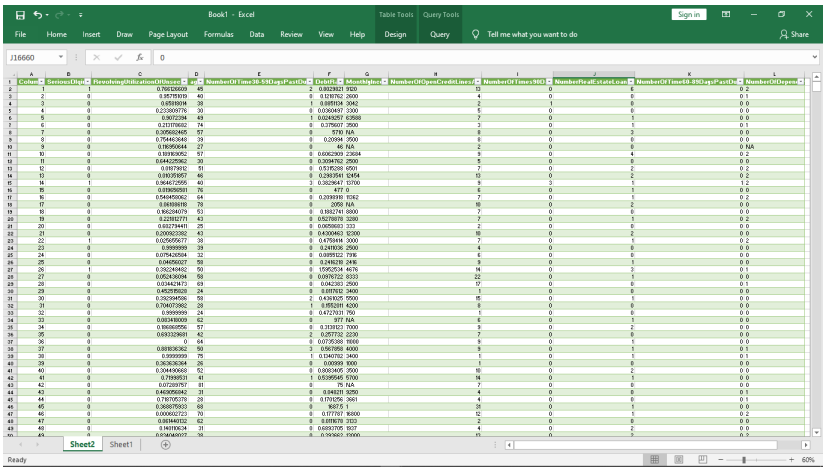

FIGURE 1 – Apercu des données

1.2.2 Problématique

Identifier les clients (emprunteurs) qui auraient des difficultés financières dans les deux années à venir afin de leur proposer des offres adaptées.

1.2.3 Solution

Construire des **modèles de prévision de score**. Ce score servira à attribuer **une classe** à chaque client. Les modèles construits vont exploiter les données des fichier *ScoringTraining.csv* et *Scoring-Test.csv* pour l'apprentissage et la validation, ainsi que pour l'évaluation des modèles afin d'en choisir le plus performant.

2 Techniques et environnement logiciel

2.1 Introduction

Ce chapitre sera dédié à la présentation des techniques et logiciels sur lesquelles on s'est basé pour réussir l'implémentation de notre solution.

2.2 Techniques

2.2.1 Nettoyage de données

Les données dans leur forme brute peuvent avoir plusieurs erreurs et valeurs manquantes. Pour amener le modèle de prévision à prendre des décisions correctes, il est nécessaire que les informations soient fiables et complètes. Nous allons nous servir de représentations graphiques de données statistiques, notamment les boites à moustaches dans le but de repérer les valeurs aberrantes de chaque variable. En ce qui concerne les valeurs manquantes, nous allons les remplir en utilisant une régression linéaire multiple stochastique.

2.2.2 Régression linaire Stochastique

C'est la solution que nous proposons pour traiter les valeurs manquantes, elle consiste à remplacer ces données par les valeurs prédites par un modèle de régression et répéter ce processus pour chaque variable. Un inconvénient majeur de cette méthode est que nous réduisons la variabilité de la variable imputée. En d'autres termes, puisque nous remplaçons les données manquantes par des sorties de régression, les valeurs prédites se trouvent au long de l'hyperplan de régression où les variables auraient en fait contenu du bruit / biais. Pour ajouter de l'incertitude aux valeurs des variables imputées, nous pouvons ajouter du bruit normalement distribué avec une moyenne de zéro et la variance égale à l'erreur-type des estimations de régression.[1]

2.2.3 Analyse discriminante

C'est un ensemble de méthodes statistiques de réduction de dimension qui vise à prédire l'appartenance d'un individu a des groupes prédéfinis à partir d'une série de variables prédictives. Le principe consiste à projeter les individus dans une direction permettant de mettre en évidence les groupes. À cette fin, Il faut privilégier la variance interclasse au détriment de la variance intraclasse considérée comme due au bruit. Dans la mise en œuvre de notre solution nous allons implémenter une analyse factorielle discriminante (AFD), une Analyse discriminante linéaire (LDA) et une analyse discriminante quadratique (QDA).

2.3 Environnement logiciel

2.3.1 Pandas

FIGURE 2 – Logo de Pandas

Pandas est une bibliothèque spécialisée dans la manipulation de données. Cette bibliothèque contient un ensemble de fonctions optimisées pour la gestion des Big Data. Elle permet de créer et d'exporter des tableaux de données à partir de fichiers texte (séparateurs, .csv, format fixe, compressé), binaires (HDF5 avec Pytable), HTML, XML, JSON, MongoDB, SQL. . . [2]

2.3.2 Microsoft SQL Server

FIGURE 3 – Logo de Microsoft SQL Server

SQL(sigle de Structured Query Language, en français langage de requête structurée) est un langage informatique normalisé servant à exploiter des bases de données relationnelles. La partie langage de manipulation des données de SQL permet de rechercher, d'ajouter, de modifier ou de supprimer des données dans les bases de données relationnelles.[3]

3 Mise en oeuvre de la solution

3.1 Phase de prétraitement

3.1.1 Préparation des données

Proportion de défauts

Le taux de défaut est un ratio calculant le pourcentage de non-paiement des échéances d'un emprunt[4] et se calcule à partir de la formule suivante [5] :

$$p = NombreDePrtsEnRetardOuEnDfautDePaiementDepuisPlusDeuxMois \div NombreDePrtsEnCours$$

Ces données sont disponibles dans les colonnes **NumberOfOpenCreditLinesAndLoans**, **NumberOfTimes90DaysLate**, **NumberRealEstateLoansOrLines**, **NumberOfTime60-89DaysPastDueNotWorse**. La requête SQL suivante permet d'effectuer ce calcul

```
SELECT ( Sum(numberoftimes90dayslate)
         + Sum(numberoftime60-89dayspastduenotworse) ) / ( Sum(numberofopencreditlinesandloans)
                                                           + Sum(numberrealestateloansorlines) )
FROM   dbo.scoringtraining
```

FIGURE 4 – Requête taux de défaut

Résultat : 5.35%

Données extrêmes

Une donnée extrême ou aberrante est une valeur ou une observation qui est distante des autres observations effectuées sur le même phénomène. Afin de détecter ces valeurs, on se sert des **boites à moustache** générées pour chaque variable. Le code permettant d'obtenir ces figures est disponible en annexe.

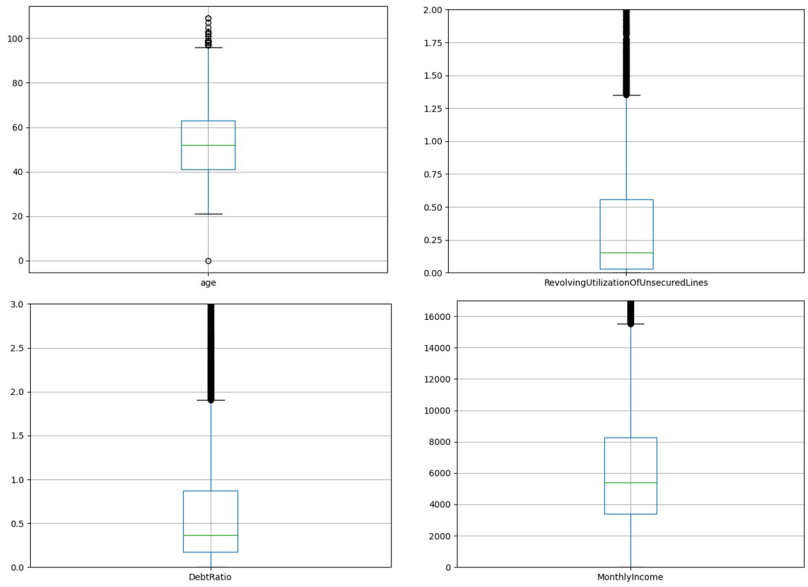

FIGURE 5 – Boites à moustaches

En se basant sur les graphes. Nous avons décidé de conserver les intervalles de données valides suivants :

Variable	Intervalle
RevolvingUtilizationOfUnsecuredLines	0-1.4
Age	20-98
NumberOfTime3059DaysPastDueNotWorse*	NombreCredits*24
DebtRatio	0-1.4
RevolvingUtilizationOfUnsecuredLines	0-1.9
MonthlyIncome	100-15800
NumberOfOpenCreditLinesAndLoans	0-20
NumberOfTimes90DaysLate*	NombreCredits*24
NumberRealEstateLoansOrLines	0-5
NumberOfTime6089DaysPastDueNotWorse*	NombreCredits*24
NumberOfDependents	0-4

***Ces variables ne doivent pas dépasser la valeur : (nombre de crédits) *24 puisque les données s'étalent en durée sur 2 années.**

FIGURE 6 – Tableau des intervalles de données valides

Valeurs manquantes

Le package **missingno** de python fournit un ensemble d'outils de visualisations de données manquantes flexibles et faciles à utiliser qui permettent d'obtenir un résumé visuel rapide de l'exhaustivité (ou de l'absence de celle-ci) d'un ensemble de données. [6]
L'application de ces outils sur notre base de données donne les résultats suivants :

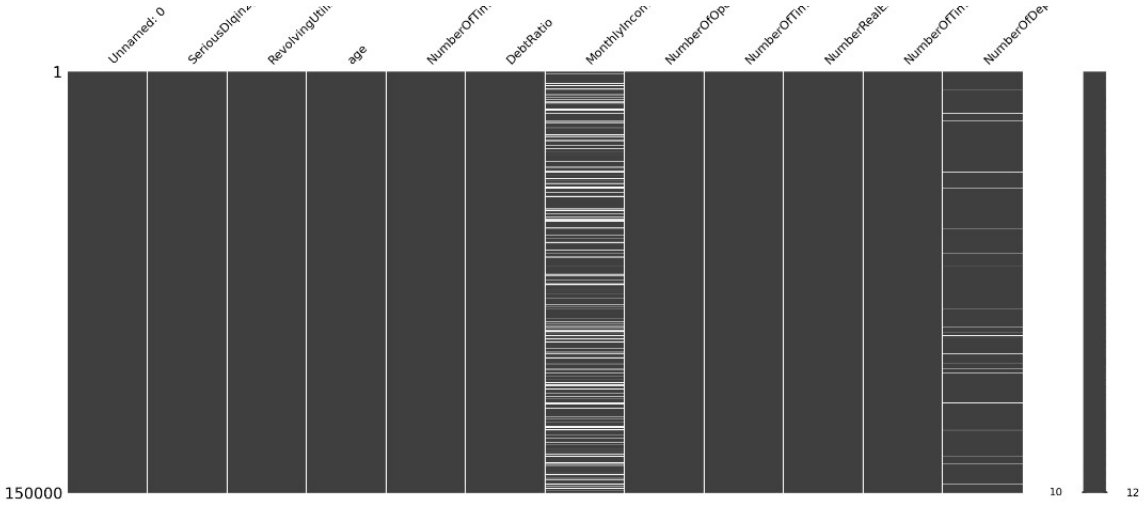

FIGURE 7 – Matrice de nullité

La matrice de nullité permet de visualiser la distribution des données manquantes. Remarquons donc que les variables concernées par ce problème sont **MonthlyIncome** et **NumberOfDependents** dans une moindre mesure. Nos remarques sont confirmées par la visualisation de la nullité par colonne

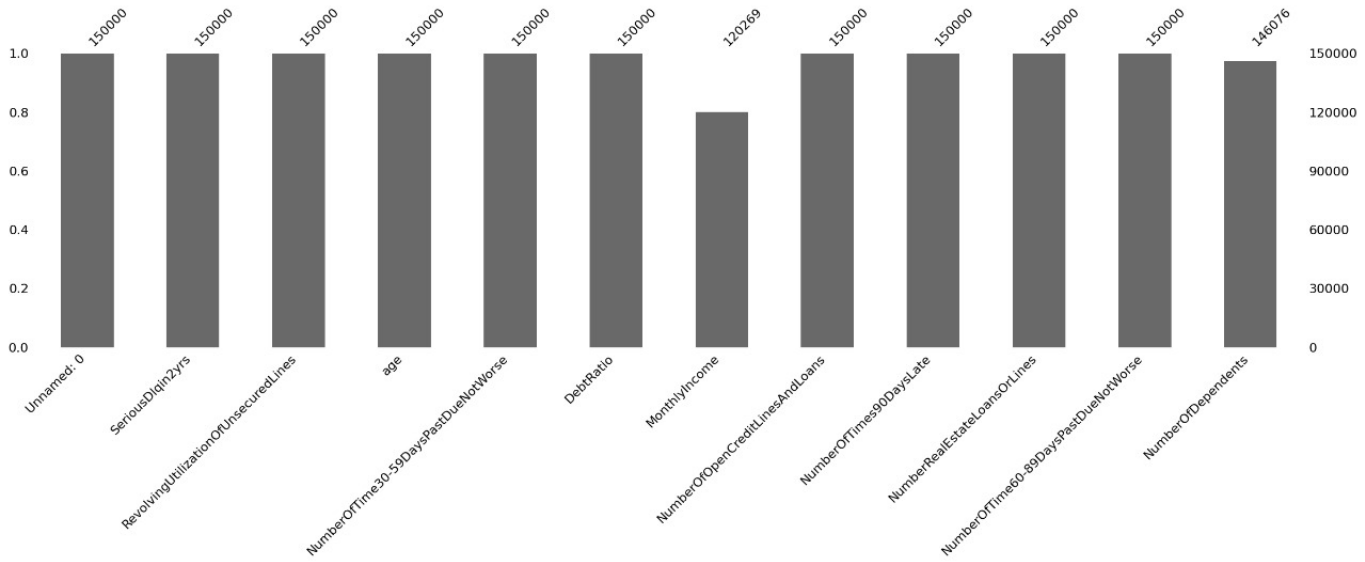

FIGURE 8 – Valeurs complètes par colonne

Pour assurer le bon apprentissage du modèle, et puisque la simple suppression des lignes avec des données manquantes entraînera à la perte de 20% de la taille initiale de la base de données, nous avons décidé de remplir ces valeurs nulles en appliquant une régression linaire stochastique sur la colonne **NumberOfDependents** et puis sur la colonne **MonthlyIncome**. Cette régression va deviner les valeurs manquantes en se basant sur les informations apportées par les variables sans nullité. Les scores R2 de ces régressions sont perspectivement 0.08 et 0.06.

Equilibrage des données d'apprentissage

Après la suppression des valeurs aberrantes, la base de données contient désormais 140.354 clients dont seulement 9025 ont **SeriousDlqin2yrs**=1. Afin d'équilibrer les données d'apprentissage, nous exécutons la requête SQL suivante :

```
SELECT *
FROM   dbo.data
WHERE  seriousdlqin2yrs = 1
UNION
SELECT TOP 9025 *
FROM   dbo.data
WHERE  seriousdlqin2yrs = 0
```

FIGURE 9 – Requête SQL de l'équilibrage

L'échantillon résultant comporte exactement le même nombre de clients ayant **SeriousDlqin2yrs**=1 que de clients avec **SeriousDlqin2yrs**=0 et a une taille globale de 18050 lignes.

Identification des meilleurs prédicteurs

On groupe la population suivant la variable **SeriousDlqin2yrs** On représente (boites à moustache) les deux classes pour chaque variable :

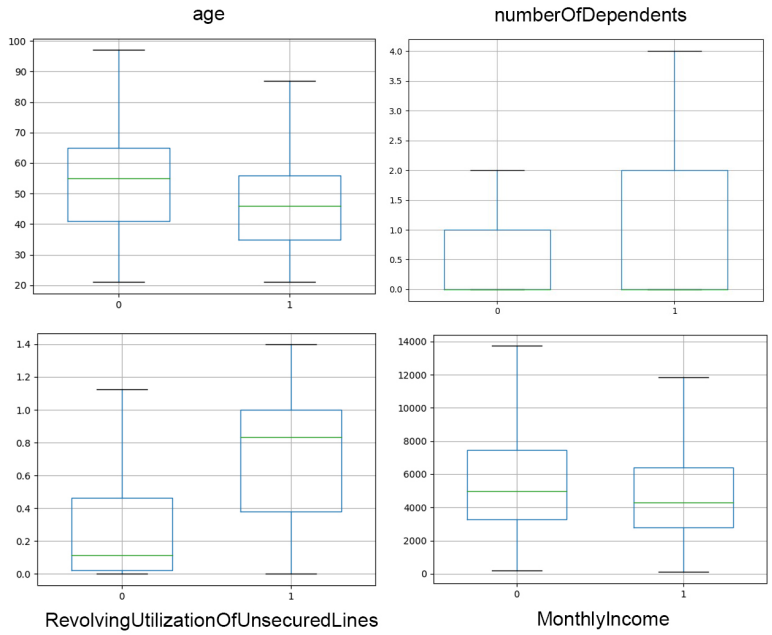

FIGURE 10 – Comparaison entre les 2 classes de clients

D'après ces représentations, nous estimons que les variables **RevolvingUtilizationOfUnsecuredLines** , **Age**, **MonthlyIncome**, **NumberOfDependents** et **NumberOfTimes90DaysLate** seront cruciales à l'identification des classes.

3.2 Modèle de prévision

Afin de valider nos modèles, nous avons choisi une méthode de validation croisée basée sur les fonctions *RepeatedStratifiedKFold* et *cross_val_score* du package **sklearnmodel_selection**. Le code est disponible en annexe.

3.2.1 Analyse factorielle discriminante

Malheureusement, nous n'avons pas trouvé d'outil prédéfini pour appliquer l'AFD,nous avons donc tenté de l'implémenter nous même. Nous avons écrit une fonction qui effectue l'apprentissage en se basant sur le modèle mathématique. La fonction calcule la barre de décision qu'on a trouvé égale à *c=-5.26e-5.*

3.2.2 Linear & Quadratic Disciminant Analysis

On applique une LDA sur les données de l'échantillon en spécifiant Y= **SeriousDlqin2yrs** la variable à expliquer en fonction des autres. Après l'apprentissage, on choisit 300 points aléatoirement et ont les visualise d'une façon à mettre en valeur la séparation des groupes.

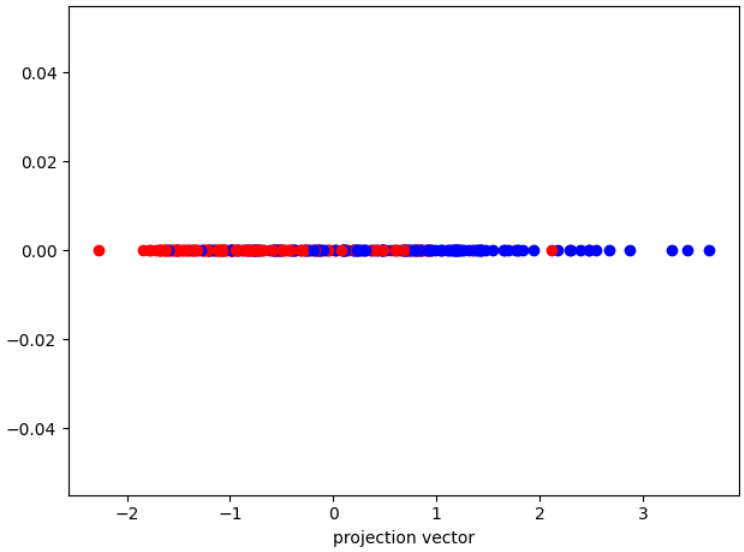

FIGURE 11 – Séparation des groupes en LDA

On effectue la QDA de la même manière.

3.2.3 Logistic Regression

On applique une régression logistique sur les données et on représente graphiquement chaque variable dans une figure.

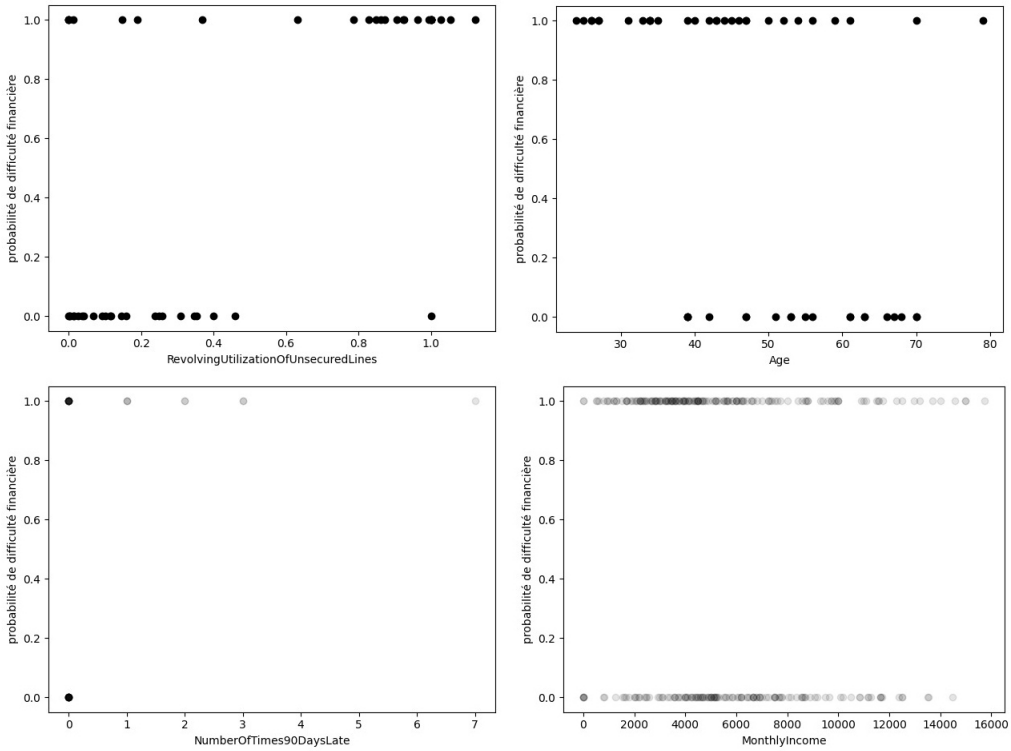

FIGURE 12 – Régression Logistique

Pour les variables **NumberOfTime90DaysLate** et **MonthlyIncome**, nous avons réduit l'opacité des points pour mieux distinguer les points superposées.
Remarquons que la variable **RevolvingUtilizationOfUnsecuredLines** prend des valeurs différentes pour chaque classe. Ceci peut être observé pour d'autres variables aussi : les personnes qui trouvent des difficultés financières (**SeriousDlqin2yrs**=1) sont généralement plus jeune et touchent en moyenne un salaire inférieur à ceux qui ne trouvent pas de problème.
Ces remarques confirment les hypothèses que nous avons fait dans la phase de prétraitement.

3.3 Phase d'évaluation et règle de décision retenue

3.3.1 Goodness of fit

On se sert de la méthode de validation définie préalablement pour calculer le score de précision de chaque modèle. Le tableau suivant décrit les résultats.

Technique	Score
LDA	0.757
QDA	0.742
Linear Regression	0.770

FIGURE 13 – Scores de précision des modèles

3.3.2 Pouvoir de prédiction

On applique nos 3 modèles sur les données du fichier *ScoringTest.csv* après l'avoir nettoyé des valeurs manquants. La fonction **confusion_matrix** permet d'obtenir les matrices de confusion suivantes :

	Prédit 0	Prédit 1
Correct 0	66666	13374
Correct 1	4	1356

FIGURE 14 – Matrice de confusion LDA

	Prédit 0	Prédit 1
Correct 0	70802	9238
Correct 1	3	1357

FIGURE 15 – Matrice de confusion QDA

	Prédit 0	Prédit 1
Correct 0	71268	8772
Correct 1	3	1357

FIGURE 16 – Matrice de confusion Logistic Regression

A partir de ces matrices de confusion on calcule le pouvoir de prédiction et on obtient les résultats suivants :

Technique	Score
LDA	0.835
QDA	0.886
Linear Regression	0.892

FIGURE 17 – Tableau des pouvoirs de prédiction

On trace également les courbes ROC de chaque modèle. Puisque la précision des modèles est très proche, les courbes ROC sont très similaires.

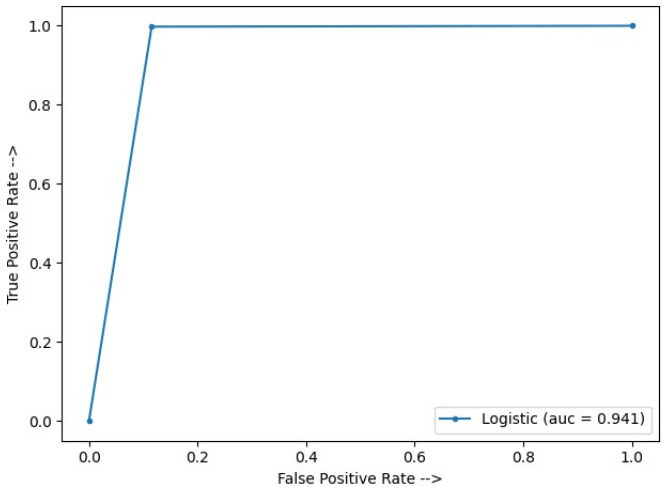

FIGURE 18 – Courbe ROC de Logistic Regression

3.3.3 Conclusion

A partir de ce résultats, on déduit que les 3 modèles donnent des résultats satisfaisants et sont capable de prédire les classes -surtout la classe 0- avec précision mais le modèle basé sur la **régression logistique** reste le plus performant.

Ce qui est logique et prévisible parce que on a les propriétés suivantes qui favorisent une régression logistique[7] :

- La variable à prédire Y est de nature dichotomique. Autrement dit,il s'agit d'un problème de classification binaire (oui/non).
- IL n'y a pas de valeurs aberrantes dans les données car elles ont été supprimé dans la phase de prétraitement.
- La corrélation entre les variables explicatives X est faible.

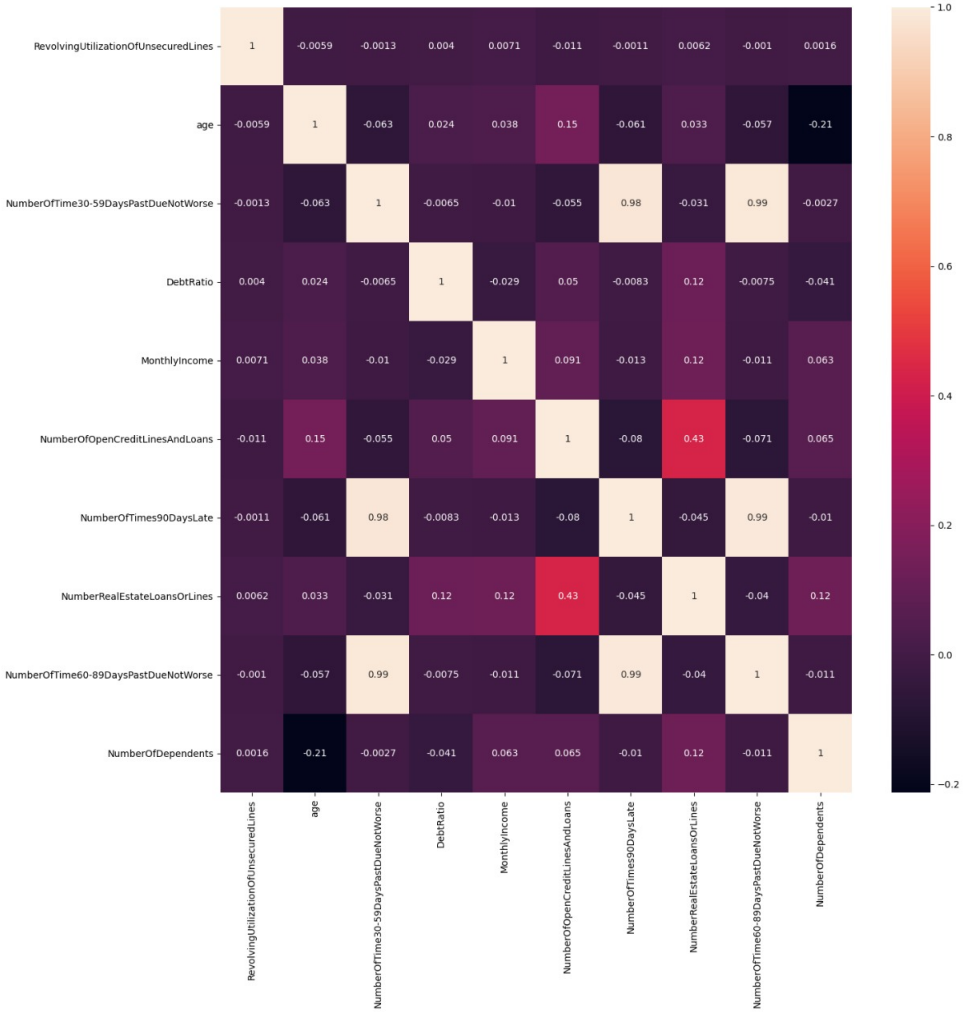

FIGURE 19 – Matrice de corrélation

4 Références bibliographiques

[1] Missing Data Imputation using Regression - https://www.kaggle.com/shashankasubrahmanya/missing-data-imputation-using-regression

[2] Pandas : Project Description - https://pypi.org/project/pandas/

[3] Structed Query Language - https://fr.wikipedia.org/wiki/Structured_Query_Language

[4] Notion De Taux De Défaut (Default Rate) - https://aide.wesharebonds.com/hc/fr/articles/203886952-Notion-de-taux-de-d%C3%A9faut-default-rate

[5] Taux de défaut : Comprendre cet indicateur si complexe - https://argent-et-salaire.com/taux-de-defaut/

[6] missingno - https://github.com/ResidentMario/missingno

[7] What is Logistic Regression ? - https://www.statisticssolutions.com/what-is-logistic-regression/

5 Annexe

— **Création d'une boite à moustaches**

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3
4  # changement de l'echelle de l'axe
5  plt.ylim(0, 3)
6  # lecture du fichier csv
7  data = pd.read_csv("ScoringTraining.csv")
8  # creation d'une boite à moustache
9  boxplot = data.boxplot(column
       ="NumberOfDependents")
10 plt.show()
```

— **Création de la matrice de nullité**

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  from quilt.data.ResidentMario import missingno_data
4  import missingno as msno
5
6  # lecture du fichier csv
7  data = pd.read_csv("ScoringTraining.csv")
8  collisions = missingno_data.nyc_collision_factors()
9  msno.matrix(data)
10 plt.show()
```

— **Création des boites à moustaches par classe**

```python
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("training_data.csv")
# splitting dataframe by groups
grouped = data.groupby(data.SeriousDlqin2yrs)
# grouping by particular dataframe column
data_0 = grouped.get_group(0)
data_1 = grouped.get_group(1)

ax = plt.axes()
# first boxplot
boxplot = data_0.boxplot(
    column="NumberOfDependents", positions=[1], widths=0.6, showfliers=False
)
# second boxplot
boxplot = data_1.boxplot(
    column="NumberOfDependents", positions=[2], widths=0.6, showfliers=False
)
ax.set_xticklabels(["0", "1"])
plt.savefig("NumberOfDependents_compare.png")
```

— Linear Discriminant Analysis

```
import pandas as pd
import matplotlib.pyplot as plt
import random
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# ouvrir le fichier csv
csv = pd.read_csv("C:\\Users\\pc\\Desktop\\training_data.csv")
# définition de la variable à prédire Y et des variables explicatives X
Y = csv.iloc[:, 1]
X = csv.iloc[:, 2:11]
# Définition du modèle
lda = LinearDiscriminantAnalysis(n_components=1)
# Apprentissage
score = lda.fit(X, Y).transform(X)
# Prévision de Y
y_pred = lda.predict(X)
# Evaluation du modèle
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(lda, X, Y, scoring="accuracy", cv=cv, n_jobs=-1)
print("Mean Accuracy: %.3f (%.3f)" % (numpy.mean(scores), numpy.std(scores)))
# Selection aléatoire de 300 points pour la visualisation
random_choices = [random.randint(0, len(score) - 1) for i in range(0, 300)]
score1 = [score[i] for i in random_choices]
y_train1 = [Y[i] for i in random_choices]
# Représentation graphique
for k, i in zip(score1, y_train1):
    if i == 1:
        plt.scatter(k, 0, c="b")
    else:
        plt.scatter(k, 0, c="r")
plt.xlabel("projection vector")
plt.show()
```

— Quadratic Discrimination Analysis

```
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
import  pandas as pd
from sklearn.model_selection import RepeatedStratifiedKFold, cross_val_score
# Memes étapes de LDA mais on change le type du modèle
csv=pd.read_csv("C:\\Users\\pc\\Desktop\\training_data.csv")
Y=csv.iloc[:,1]
X=csv.iloc[:,2:11]
qda=QuadraticDiscriminantAnalysis()
pred=qda.fit(X,Y)
y_pred=qda.predict(X)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(lda, X, Y, scoring='accuracy', cv=cv, n_jobs=-1)
print('Mean Accuracy: %.3f (%.3f)' % (numpy.mean(scores), numpy.std(scores)))
print("your accuracy is :"+str(accuracy))
```

— **Logistic Regression**

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
import random
from sklearn.metrics import accuracy_score
# Memes étapes de LDA mais on change le type de modèle
csv = pd.read_csv("C:\\Users\\pc\\Desktop\\training_data.csv")
Y = csv.iloc[:, 1]
X = csv.iloc[:, 2:11]
logistic_reg = LogisticRegression()
score = logistic_reg.fit(X, Y)
# On choisit une variable à visualiser
ruoul = csv.iloc[:, 2]
# Selection aléatoire de 50 points
random_choices=[random.randint(0,len(score)-1) for i in range(0,50)]
score1=[score[i] for i in random_choices]
y_train1=[ruoul[i] for i in random_choices]
# Représentation graphique
for k, i in zip(score, y_train1):
    plt.scatter(i, k, c="k")
plt.ylabel("probabilité de difficulté financière")
plt.xlabel("RevolvingUtilizationOfUnsecuredLines")
plt.show()
```

— **Prédiction des données de *ScoringTest.csv***

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, r2_score, confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

csv = pd.read_csv("C:\\Users\\Achraf\\PycharmProjects\\Scoring\\training_data.csv")
# le fichier test.csv contient les données de ScoringTest.csv + la colonne probabilité
# du fichier SampleScoring.csv
csv2 = pd.read_csv("C:\\Users\\Achraf\\PycharmProjects\\Scoring\\test.csv")
Y = csv.iloc[:, 1]
X = csv.iloc[:, 2:11]
X_test = csv2.iloc[:, 2:11]
Y_true_tmp = csv2.iloc[:, -1]
# Déduit les valeurs bool à partir des probabilités correctes
Y_true = []
for i in Y_true_tmp:
    if i > 0.5:
        Y_true.append(True)
    else:
        Y_true.append(False)
model = LogisticRegression()
# Apprentissage sur les données de training_data
score = model.fit(X, Y)
# prédition de Y sur le fichier ScoringTest
Y_pred_tmp = model.decision_function(X_test)
Y_pred = []
# Déduit les valeurs bool à partir des probabilités prédites
for i in Y_pred_tmp:
    if i > 0.5:
        Y_pred.append(True)
    else:
        Y_pred.append(False)
```

— **Courbe ROC**

```
logistic_fpr, logistic_tpr, threshold = roc_curve(Y_true, Y_pred)
auc_logistic = auc(logistic_fpr, logistic_tpr)
plt.figure()
plt.plot(
    logistic_fpr,
    logistic_tpr,
    marker=".",
    label="Logistic (auc = %0.3f)" % auc_logistic,
)
plt.xlabel("False Positive Rate -->")
plt.ylabel("True Positive Rate -->")

plt.legend()
plt.show()
```

— **Matrice de Confusion**

```
matrix = confusion_matrix(Y_true, Y_pred)
```

— **Matrice de Corrélation**

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sn

csv = pd.read_csv("C:\\Users\\Achraf\\PycharmProjects\\Scoring\\ScoringTraining.csv")
X = csv.iloc[:, 2:]
plt.figure(figsize=(15, 15))
corrMatrix = X.corr()
sn.heatmap(corrMatrix, annot=True)
plt.show()
```

— **Régression linéaire Stochastique**

```
import pandas as pd
import matplotlib.pyplot as plt
import missingno as msno
from sklearn import linear_model
from sklearn.svm import SVC
import numpy as np

data = pd.read_csv("ScoringTraining.csv")
colnames = list(data.columns)
missing_columns = ["NumberOfDependents",'MonthlyIncome']

random_data = pd.DataFrame(columns=["Ran" + name for name in missing_columns])

for feature in missing_columns:
    random_data["Ran" + feature] = data[feature + '_imp']
    parameters = list(set(data.columns) - set(missing_columns) - {feature + '_imp'})

    model = linear_model.LinearRegression()
    model.fit(X=data[parameters], y=data[feature + '_imp'])

    # Standard Error of the regression estimates is equal to std() of the errors of each estimates
    predict = model.predict(data[parameters])
    std_error = (predict[data[feature].notnull()] - data.loc[data[feature].notnull(), feature + '_imp']).std()

    # observe that I preserve the index of the missing data from the original dataframe
    random_predict = np.random.normal(size=data[feature].shape[0],
                                      loc=predict,
                                      scale=std_error)
    random_data.loc[(data[feature].isnull()) & (random_predict > 0), "Ran" + feature] = random_predict[
        (data[feature].isnull()) &
        (random_predict > 0)]

data.to_csv (r'C:\Users\Achraf\PycharmProjects\Scoring\data_reg.csv', index = False, header=True)
```

— **Analyse Factorielle Discriminante**

```python
import pandas as pd
import numpy as np
class AFD:
    def fit(self, X, y):
        n_features = X.shape[1]  # prendre le nombre de colone existent en X
        class_labels = np.unique(y)  # vrenvoi le valeur unique du vecteur y

        mean_overall = np.mean(X, axis=0)  # calculer g totale
        SW = np.zeros(
            (n_features, n_features)
        )  # creer une matrice de zero de n_features ligne et de n_features colone
        SB = np.zeros((n_features, n_features))
        gravities = []
        for c in class_labels:
            X_c = X[y == c]  # creer la matrice x_c depuis x lorsque y=c
            mean_c = np.mean(X_c, axis=0)  # calculer la gravites des classes
            SW = (X_c - mean_c).T.dot((X_c - mean_c))  # calculer W
            n_c = X_c.shape[0]
            # calculer B
            mean_diff = mean_c - mean_overall
            SB = n_c * (mean_diff).dot(mean_diff.T)
            gravities.append(mean_c)
        V = np.add(SB, SW)  # calculer v
        V_inv = np.linalg.inv(V)  # calculer linv de v
        u = np.dot(V_inv, np.subtract(gravities[0], gravities[1]))  # calculer u
        a = np.subtract(gravities[0], gravities[1])  # calculer a
        s1 = np.dot(np.transpose(gravities[0]), np.linalg.inv(SW))  # calculer s1
        s1 = np.dot(s1, a)
        s2 = np.dot(np.transpose(gravities[1]), np.linalg.inv(SW))  # calculer s2
        s2 = np.dot(s2, a)
        c = np.add(s1, s2) * 1 / 2  # calculer c
        print(c)
csv = pd.read_csv("C:\\Users\\pc\\Desktop\\training_data.csv")
Y = csv.iloc[:, 1]
X = csv.iloc[:, 2:11]
afd = AFD()
afd.fit(X, Y)
```