UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

# Apprentissage Statistique Avancé

Laurène DAVID – Safa HAMDAN – Allan PENDANT

Juin 2022

# Table des matières

# Introduction

L'objectif de ce jeu de données bancaires est de concevoir un modèle de prédiction de défaut de paiement.

Ce jeu contient :

- 5 000 observations
- 21 variables dont :
    - 8 variables numériques (type int64)
    - 13 variables objet (type object)

Après vérification, cette base de donnée ne contient pas de valeurs manquantes.

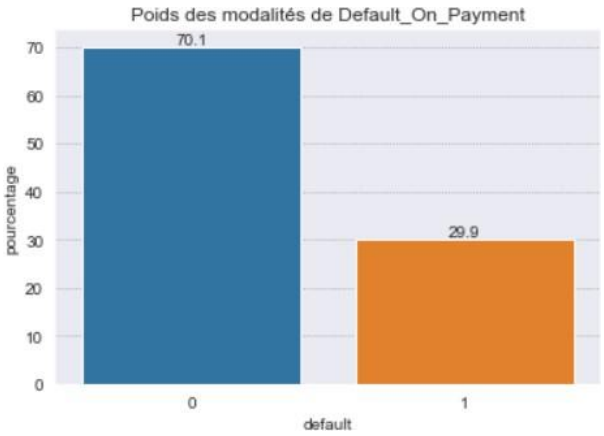
# I – Analyse Descriptives

## 1- Variable cible

La variable cible pour notre future modèle de prédiction étant « Default_On_Payment », effectuons-en une brève analyse

| | default | count | perct |
|---|---|---|---|
| **0** | 0 | 3505 | 70.1 |
| **1** | 1 | 1495 | 29.9 |

Descriptif de la variable default_On_Payment




Proportion des différentes modalités de la variable Default_On_Payment

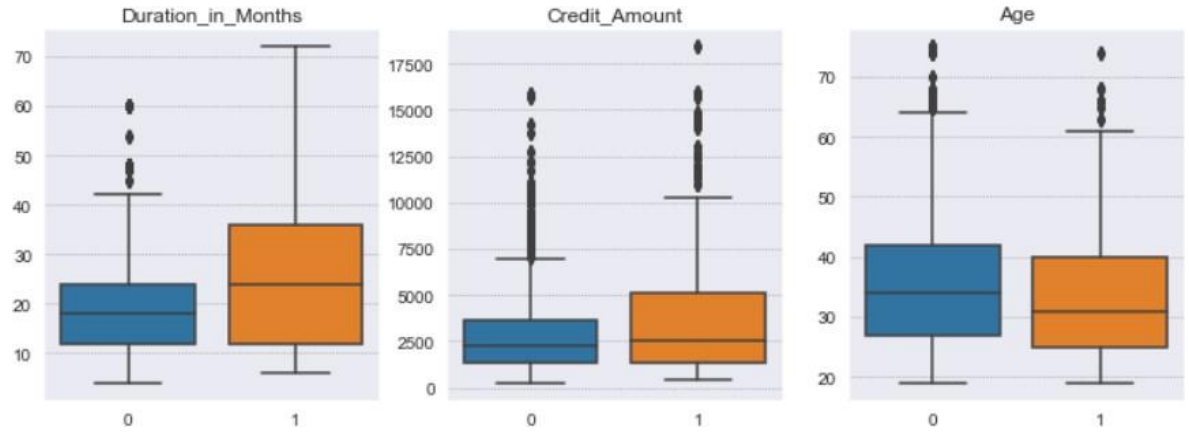
Nous avons 3505 observations avec un défaut de paiement noté 0 et 1495 noté 1. La variable cible est donc débalancée.

## 2- Variables numériques

Intéressons-nous maintenant aux variables numériques (nous ne nous intéressons pas à la variable « Customer_ID » .

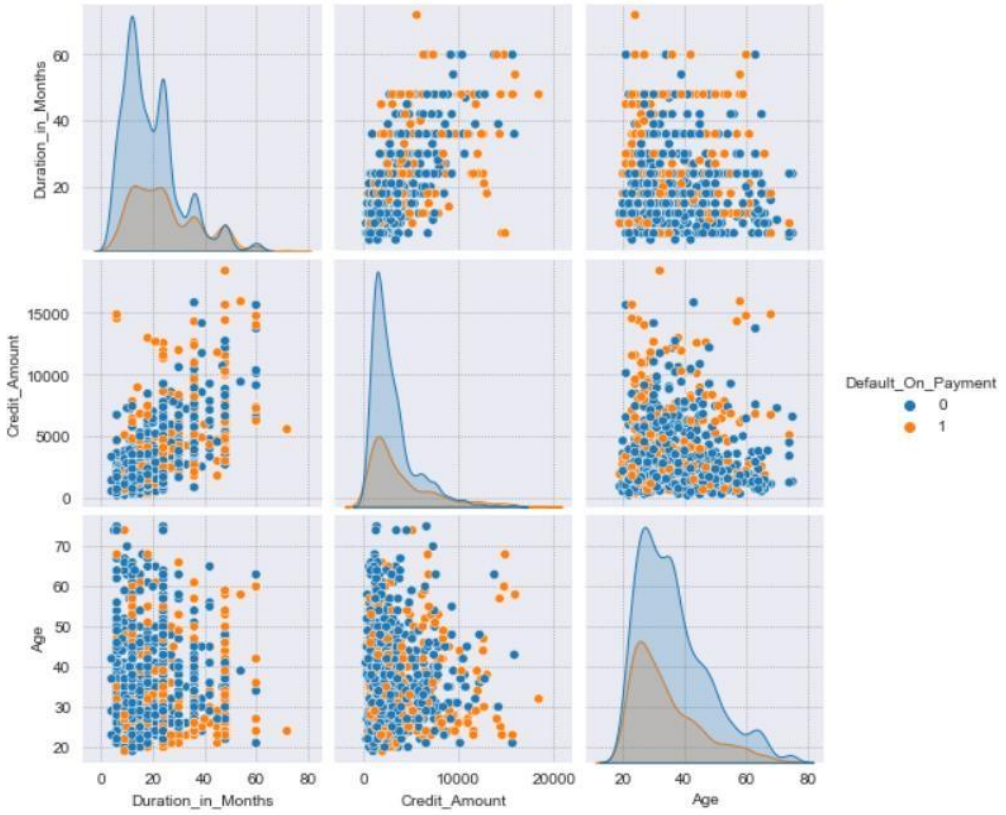
| | variable | nbr val uniques |
|---|---|---|
| **0** | Credit_Amount | 921 |
| **1** | Age | 53 |
| **2** | Duration_in_Months | 33 |
| **3** | Inst_Rt_Income | 4 |
| **4** | Current_Address_Yrs | 4 |
| **5** | Num_CC | 4 |
| **6** | Dependents | 2 |

Tableau du nombre de modalités par variable numérique



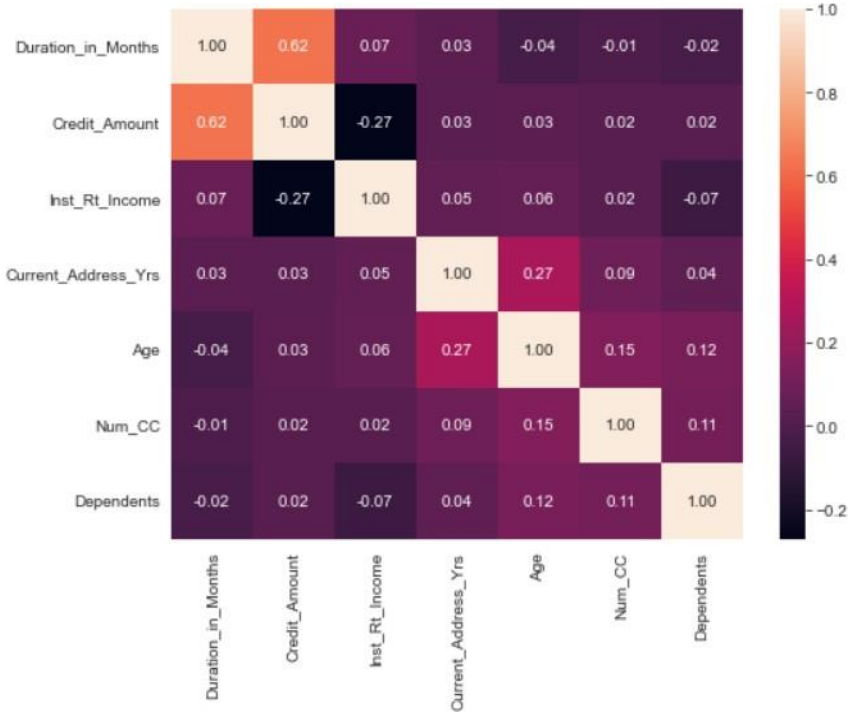

Boxplot des variables numériques à plus de 4 modalités

Nous observons que trois variables ont plus de 4 modalités. Étudions les plus attentivement :




Paireplot

Il semble y avoir une relation linéaire entre la variable Credit_Amount et Duration_in_Months. Analysons la corrélation entre les variables numériques




Matrice de corrélation

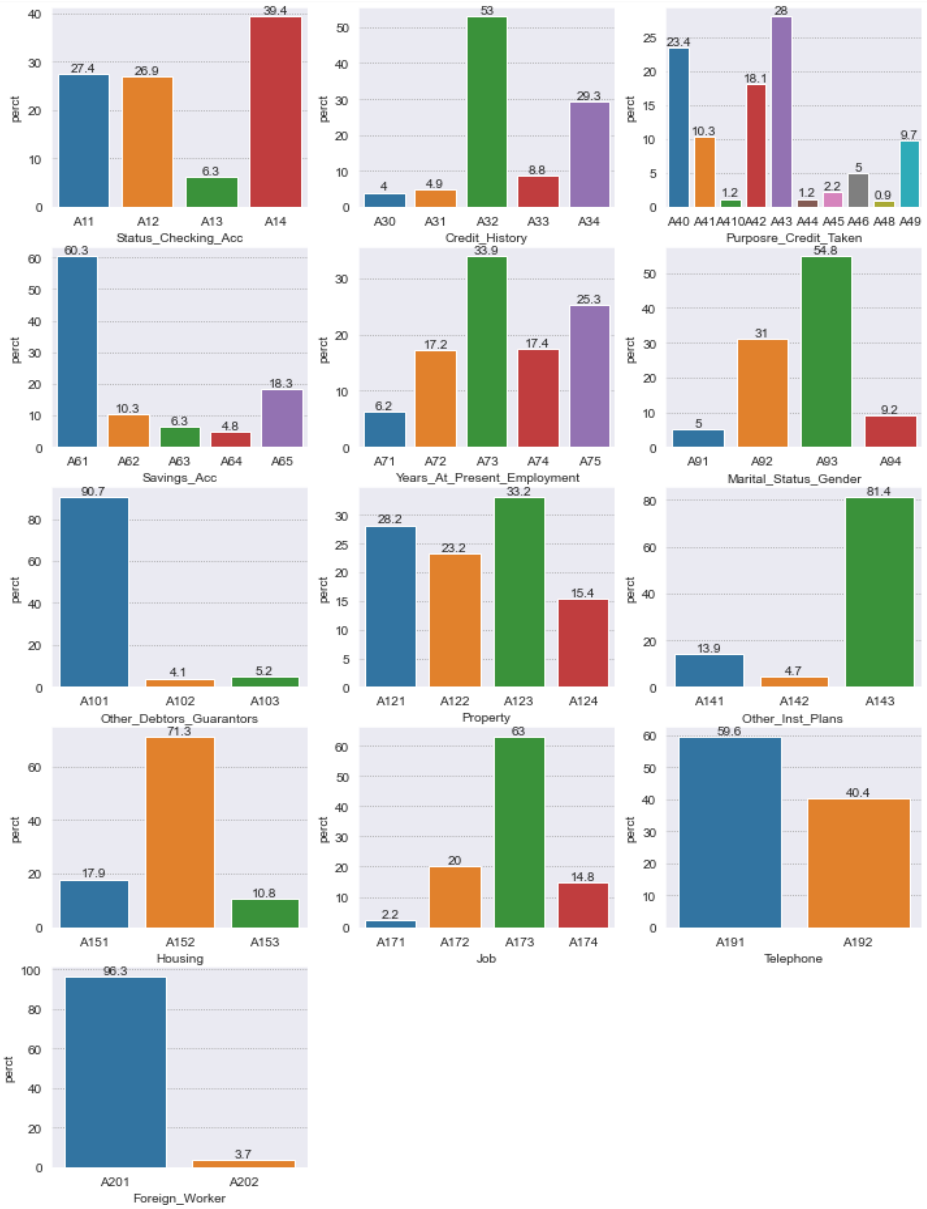
Nous voyons qu'effectivement ces deux variables sont corrélées à 62%. Cependant ce taux est trop faible pour décider de n'en garder qu'une.

### 3- Variables qualitatives non numériques

Analysons maintenant les variables qualitatives non numériques en regardant tout d'abord leur nombre de modalités et ensuite la proportion de chaque modalité au sein de ces variables.

| | |
|---|---|
| Status_Checking_Acc | 4 |
| Credit_History | 5 |
| Purposre_Credit_Taken | 10 |
| Savings_Acc | 5 |
| Years_At_Present_Employment | 5 |
| Marital_Status_Gender | 4 |
| Other_Debtors_Guarantors | 3 |
| Property | 4 |
| Other_Inst_Plans | 3 |
| Housing | 3 |
| Job | 4 |
| Telephone | 2 |
| Foreign_Worker | 2 |

Nombre de modalité par variables




Proportion de chaque modalités présentes pour chaque variable qualitative non numérique

Nous observons qu'il existe plusieurs modalités à faibles proportions. Celles-ci pourraient poser problème dans l'élaboration d'une table de contingence.

# II - Multiple Correspondance Analysis (MCA)

Dans cette prochaine partie, nous allons vous présenter la méthode de réduction de dimension que nous avons choisi, ainsi qu'expliquer le choix d'appliquer une MCA et non une FAMD pour ce jeu de données.

## 1- Découpage en train et test split

Avant d'appliquer notre méthode de réduction de dimension, nous avons construit les train et test set de notre modèle.

Cela va nous permettre d'appliquer la MCA séparément sur les jeux de données d'apprentissage et de test, afin de limiter la fuite d'information du train set vers le test set.

Puisque notre variable à prédire «Default_On_Payment» est débalancée, nous allons utiliser le paramètre « stratify » de la fonction train_test_split de sklearn pour maintenir les proportions de la variable target dans les deux jeu de données.

```
data_ = df.drop(columns=target)
target_ = df[target]

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data_, target_, test_size=0.33, stratify = df[target])
```

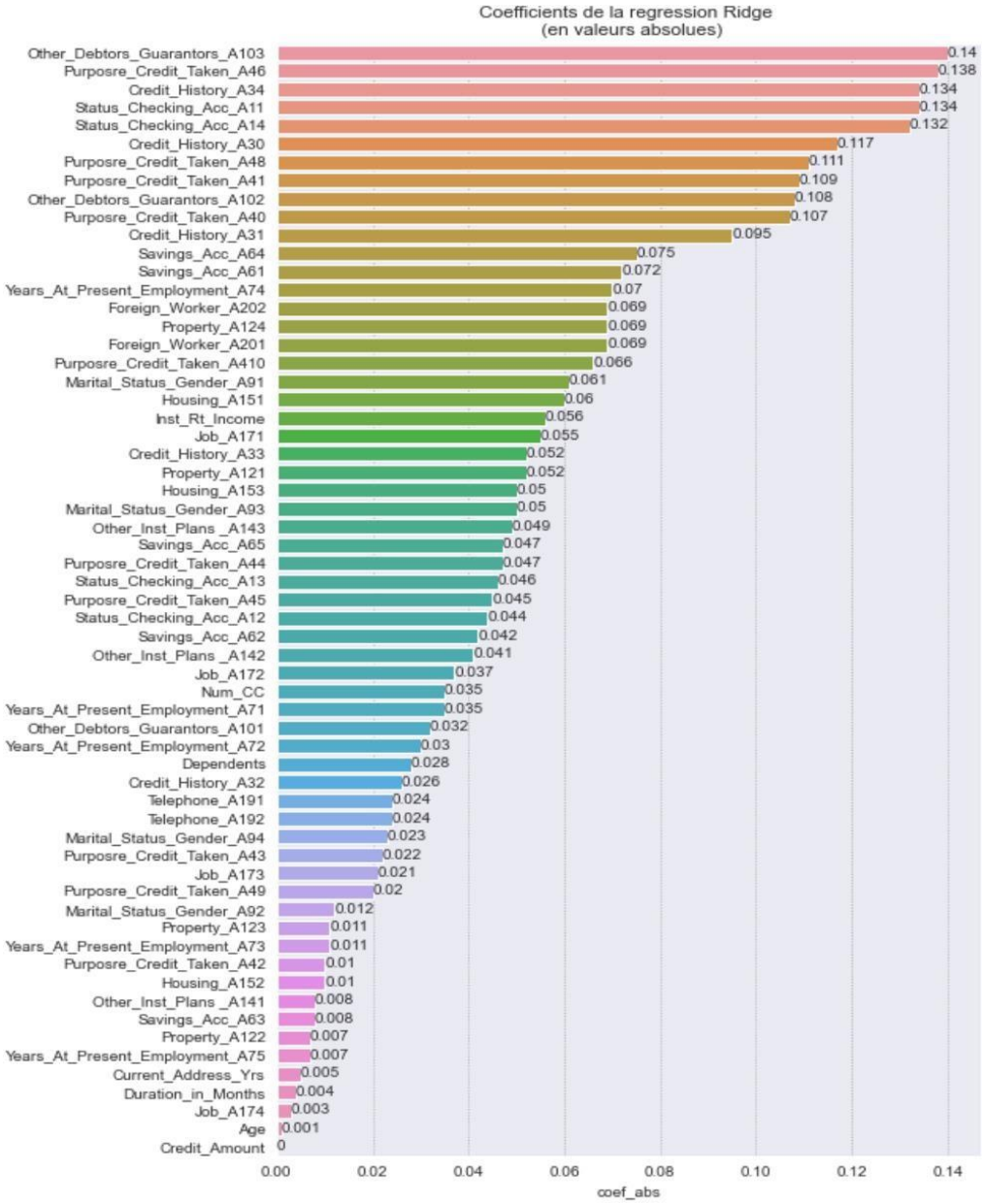
## 2- Sélection de variables

Nous avons commencé par construire la table de contingence 0/1 des variables qualitatives du jeu de données d'apprentissage. En concaténant cette table avec les variables quantitatives originales, nous nous retrouvons avec un jeu de données avec 61 variables.

Afin de limiter l'impact d'une augmentation du nombre de variable sur notre modèle, nous avons décidé d'opérer à de la sélection de variables, avant de réduire la dimension des données.

Nous avons utilisé les coefficients (en valeur absolue) d'une régression RIDGE pour sélectionner les variables pertinentes.

Les résultats de cette régression sont dans le graphique suivant :




Nous avons décidé de supprimer les 14 variables ayant un coefficient (en valeur absolue) inférieur à 0.01.

Parmi les variables supprimées, quatre des sept variables numériques du jeu de données sont présentes.

Nous avons gardé les trois variables numériques restantes (Dependents, Num_CC, Instant_Rt_Income) puisqu'elles sont suffisamment significatives pour être maintenu dans le modèle.

Nous avons tout de même remarqué que ces variables avaient un faible nombre de valeurs uniques, par rapport aux autres variables numériques.

| | variable | nbr val uniques |
|---|---|---|
| **0** | Credit_Amount | 921 |
| **1** | Age | 53 |
| **2** | Duration_in_Months | 33 |
| **3** | Inst_Rt_Income | 4 |
| **4** | Current_Address_Yrs | 4 |
| **5** | Num_CC | 4 |
| **6** | Dependents | 2 |

Nous les avons donc considérés comme des variables qualitatives avec un faible nombre de modalités. Ces variables ont été encodé par OneHotEncoding puis intégrées aux variables qualitatives restantes.

Nous avons ainsi obtenu une table de contingence construite à partir de variables entièrement qualitatives et qui nous a servi à appliquer notre Multiple Correspondance Analysis.

Voici un aperçu des cinq premières variables du tableau de contingence.

| | Status_Checking_Acc_A11 | Status_Checking_Acc_A12 | Status_Checking_Acc_A13 | Status_Checking_Acc_A14 | Credit_History_A30 |
|---|---|---|---|---|---|
| **864** | 0 | 0 | 0 | 1 | 0 |
| **3797** | 0 | 0 | 0 | 1 | 0 |
| **2081** | 0 | 0 | 0 | 1 | 0 |
| **3897** | 0 | 0 | 0 | 1 | 0 |
| **4089** | 1 | 0 | 0 | 0 | 1 |

Le jeu de données test a lui aussi été modifié pour correspondre au jeu de données d'apprentissage.

## 3- Application de la MCA

La Multiple Correspondance Analysis est une méthode de réduction de dimension destinée à réduire le nombre de variables qualitatives du modèle, sans large perte d'information. Cette méthode est utile dans notre cas puisque nous avons 59 variables au total, après création du tableau de contingence.

Comme nous l'avons énoncé précédemment, nous allons construire la nouvelle base de projection à partir du jeu de données d'apprentissage. Cette base va ensuite nous permettre de projeter le jeu de données test sur ce nouvel espace de dimension réduite.

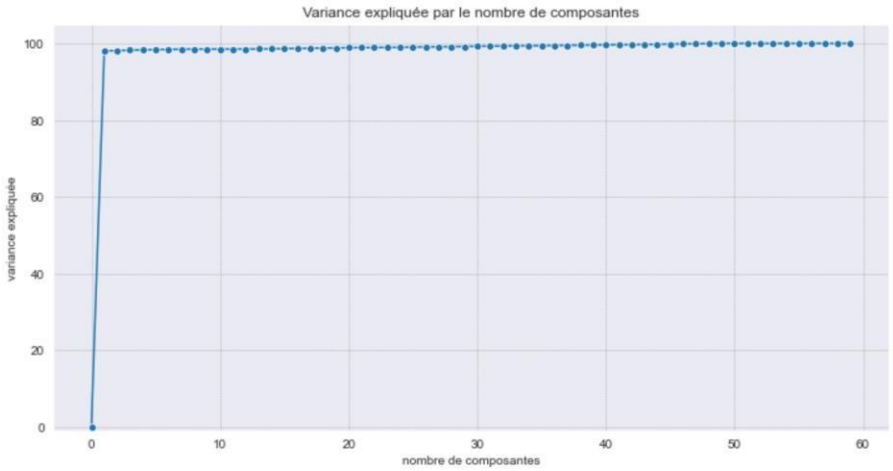
Nous avons décidé d'implémenter manuellement la MCA, à partir de l'algorithme disponible dans le cours d'Apprentissage Statistique Avancé.

Voici un aperçu des 10 premières composantes du jeu de données d'apprentissage.

| | Z1 | Z2 | Z3 | Z4 | Z5 | Z6 | Z7 | Z8 | Z9 | Z10 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.150085 | -5.960247 | -2.876377 | 2.146386 | -1.851153 | -0.584595 | -4.717614 | 0.389338 | -0.441322 | -3.412569 |
| **1** | 0.150228 | -4.976952 | -0.592712 | 2.416573 | 1.353843 | 6.596310 | -8.840323 | 0.122691 | 0.064613 | 0.212655 |
| **2** | 2.002501 | 0.587941 | 3.052753 | -5.532733 | -2.517500 | -3.498230 | 6.487602 | -2.898702 | 1.374582 | -0.333532 |
| **3** | 0.150208 | -4.131570 | 2.485451 | 4.713205 | -3.609178 | 7.369991 | -5.511872 | 0.231397 | -0.373779 | 0.000447 |
| **4** | -0.775928 | 0.033815 | 4.416404 | 5.863627 | 5.264007 | -4.191590 | -3.229549 | -0.322083 | -0.613138 | 0.219303 |

Nous avons utilisé la matrice de valeurs propres $U$ , ainsi que la matrice de poids $D_p^{-1}$ de notre première MCA pour projeter le jeu de données test dans cette nouvelle base.

Si on s'intéresse maintenant à la part de la variance expliquée par les composantes du jeu de données d'apprentissage, on obtient




La première composante semble déjà expliquée 98% de la variance du jeu de données, ce qui nous semble plutôt surprenant.

Nous verrons dans la prochaine partie comment nous avons choisi le nombre de composantes à garder dans les deux jeux de données.

# III- Modèle de Classification

L'objectif de cette partie du projet est de construire un modèle de prédiction.

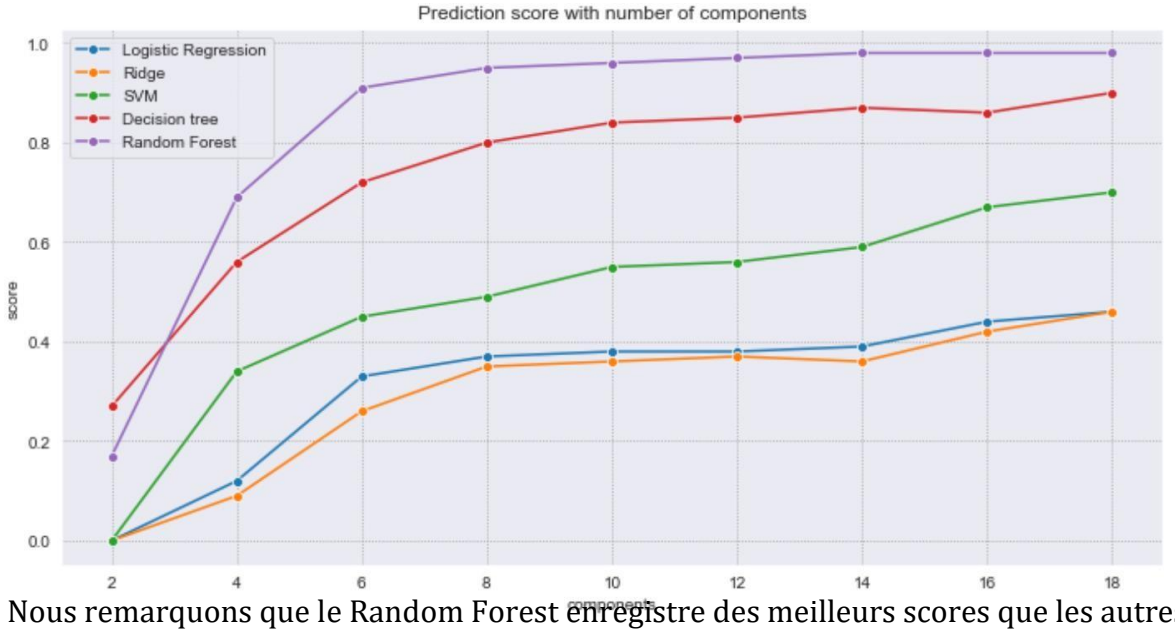
Il nous a donc semblé plus pertinent de sélectionner un nombre de composantes qui maximise la performance de notre modèle.

Pour quantifier la performance d'un modèle, nous avons choisi comme métrique le f1-score qui regroupe la Precision et le Recall en un même score. Il nous permettra d'identifier les modèles qui permettent de prédire correctement les deux modalités de notre variable target débalancée.

Nous avons choisi de tester cinq modèles de classification et d'estimer son f1-score sur le jeu de données test pour un nombre de composantes allant de 2 à 18.

Nous avons aussi laissé les paramètres par défaut pour les cinq modèles.

Nous obtenons les résultats suivants :




Nous remarquons que le Random Forest enregistre des meilleurs scores que les autres modèles de classification. Son f1-score sur le test set dépasse 0,91 à partir de 6 composantes.

Les scores du Random Forest sont déjà très élevé, même sans hyperparamétrage par GridSearch.

Nous avons ici un dataframe qui montre les scores des différents modèles.

| | components | Logistic Regression | Ridge | SVM | Decision tree | Random Forest |
|---|---|---|---|---|---|---|
| **0** | 2 | 0.00 | 0.00 | 0.00 | 0.27 | 0.17 |
| **1** | 4 | 0.12 | 0.09 | 0.34 | 0.56 | 0.69 |
| **2** | 6 | 0.33 | 0.26 | 0.45 | 0.72 | 0.91 |
| **3** | 8 | 0.37 | 0.35 | 0.49 | 0.80 | 0.95 |
| **4** | 10 | 0.38 | 0.36 | 0.55 | 0.84 | 0.96 |
| **5** | 12 | 0.38 | 0.37 | 0.56 | 0.85 | 0.97 |
| **6** | 14 | 0.39 | 0.36 | 0.59 | 0.87 | 0.98 |
| **7** | 16 | 0.44 | 0.42 | 0.67 | 0.86 | 0.98 |
| **8** | 18 | 0.46 | 0.46 | 0.70 | 0.90 | 0.98 |

Nous constatons aussi qu'à partir de 14 composantes, les f1-score du Random Forest semblent se stabiliser.

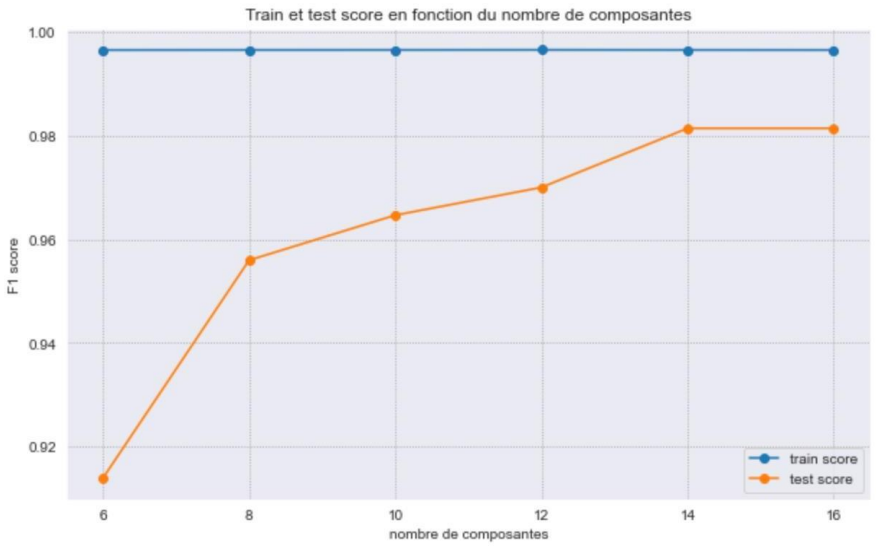
Pour faire notre choix définitif du nombre de composantes à garder, nous allons aussi calculer les train score du Random Forest.

Voici le code qui nous a permis de construire la liste des différents train et test score, avec comme métrique le f1-score.

```
f1_train = [f1_score(y_train, RandomForestClassifier().fit(X_mca_df.iloc[:,:i],y_train).predict(X_mca_df.iloc[:,:i])) for i in range(6,18,2)]
f1_test = [f1_score(y_test, RandomForestClassifier().fit(X_mca_df.iloc[:,:i],y_train).predict(X_mca_test_df.iloc[:,:i])) for i in range(6,18,2)]
```

Les jeux de données X_mca_df et X_mca_test_df correspondent au jeu de données d'apprentissage et de test, une fois transformés par le Multiple Correspondance Analysis.

Nous avons représenté les f1-score du jeu de données train et celui du test dans un même graph.




Le f1-score du train reste constant pour les différents nombres de composantes testés.

Nous pouvons donc choisir le modèle avec le meilleur test score, puisque c'est celui qui minimise le phénomène d'overfitting.

En regardant le dataframe ci-dessous, nous constatons qu'à 14 composantes, l'écart entre le train et le test score est minimisé. Nous avons donc décidé de fixer à 14 le nombre de composantes à garder dans la MCA

| | nb_composantes | train_score | test_score |
|---|---|---|---|
| **0** | 6 | 0.996 | 0.914 |
| **1** | 8 | 0.997 | 0.956 |
| **2** | 10 | 0.997 | 0.965 |
| **3** | 12 | 0.997 | 0.970 |
| **4** | 14 | 0.997 | 0.981 |
| **5** | 16 | 0.996 | 0.981 |

# Conclusion

Pour conclure, nous avons réussi à construire un modèle de classification qui prédit avec de bons résultats les deux modalités de « Default_on_Payment », même si celle-ci est débalancée.

Nous restons tout de même surpris par les résultats de notre MCA quant à la part significative de la variance expliquée par la première composante.