Université de LILLE
Faculté des Sciences et Techniques

Méthodes d'Apprentissage : Analyse des données et Machines Learning

Master 2
Ingénierie Statistique et Numérique

Professeur : M Céline Duval

Préparé par : Khalil AL-SAYED

le 30 Novembre 2021

Table des matières

.

1 Etude préliminaire

1.1 Analyse descriptive

Dans cette partie nous allons analyser la base de données default.xls dont le but de construire un bon model permettant de prédire le defaut du payement.
Sur le jeu de données comporte 30000 individus pour lesquelles 23 variables explicatives sont observées (X1 à X23) ainsi que une variable binaire Y que l'on cherche à prédire.
Nos variables sont :

- X1 : Montant du crédit (Unité, dollar) incluant les dépenses des membres de la famille.
- X2 : Genre (1= homme, 2 = femme).
- X3 : Education (1 = secondaire ; 2 = université ; 3 = supérieur ; 4 = autre).
- X4 : Situation familiale (1 = marié ; 2 = célibataire ; 3 = autre).
- X5 : Age (années).
- X6 à X11 : Historique de payements passés sur les derniers mois (d'avril à septembre 2005) X6 : septembre, X7 : août,... X11 : avril,
  Valeurs :
  -1= payé,
  1 = payé avec 1 mois de retard,
  2 = payé avec 2 mois de retard,... ,
  8 = payé avec 8 mois de retard,
  9 = payé avec 9 mois de retard et plus,

- X12 à X17 : Relevé de facturation (Unité, dollar) sur les derniers mois (d'avril à septembre 2005)
  X12 : septembre,
  X13 : août,
  ...
  X17 : avril,

- X18 à X23 : Montant de payement précédents (Unité, dollar) X18 :
  septembre,
  X19 :août,
  ...
  X23 : avril.

Dans un premier temps on va recoder les variables qualitatives en facteurs, à l'aide de la fonction **as.factor**, et pour plus de lisibilité, on va renommer les variables.
Voici un aperçu de ces données :

	genre	education	situation.familiale	X6	X7	X8	X9	X10	X11	montant.credit	age	defaut.payment
1	2	2	1	2	2	-1	-1	-2	-2	20000	24	1
2	2	2	2	-1	2	0	0	0	2	120000	26	1
3	2	2	2	0	0	0	0	0	0	90000	34	0
4	2	2	1	0	0	0	0	0	0	50000	37	0
5	1	2	1	-1	0	-1	0	0	0	50000	57	0
6	1	1	2	0	0	0	0	0	0	50000	37	0

	X13	X14	X15	X16	X17	X18	X19	X20	X21	X22	X23	defaut.payment
1	3102	689	0	0	0	0	689	0	0	0	0	1
2	1725	2682	3272	3455	3261	0	1000	1000	1000	0	2000	1
3	14027	13559	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
4	48233	49291	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
5	5670	35835	20940	19146	19131	2000	36681	10000	9000	689	679	0
6	57069	57608	19394	19619	20024	2500	1815	657	1000	1000	800	0

FIGURE 1 – Aperçu de quelques données

Donc aprés toutes ces transformations faisons un **summary** et un **str** pour verifier si les variables sont bien representées :
On remarque pour les variables :

Nous remarquons que dans la variable **X2 : genre**, on a le nombre d'hommes est 11888 tandis que le nombre de femmes est 18112, donc le nombre de femmes est supérieur à celui des hommes.

FIGURE 2 – Representation graphique de nombre des femmes et hommes dans notre base de données

Pour la variable **X3 : education**, on a 10585 étudiants en secondaire (1), 14030 étudiants universitaires(2), 4917 étudiants en supérieur (3) et 123 autres(4), on remarque que les étudiants universitaires sont les plus nombreux suivis des étudiants en secondaire et enfin on a les étudiants en supérieur suivi des autres catégories d'étudiants.

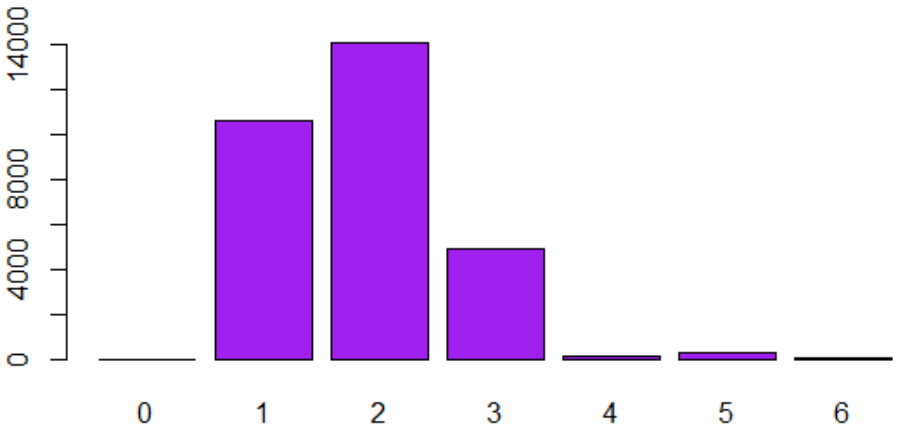


FIGURE 3 – Representation graphique de la variable education

Pour la variable **X4 : situation familiale** , on a 13659 mariés(1), 15964 célibataires (2) et 323 autres(3), donc le nombre de célibataires est le plus important de 15964 suivi du nombre de mariés de 13659 et enfin on a les autres genres de situation familiale qui sont de 323 seulement.

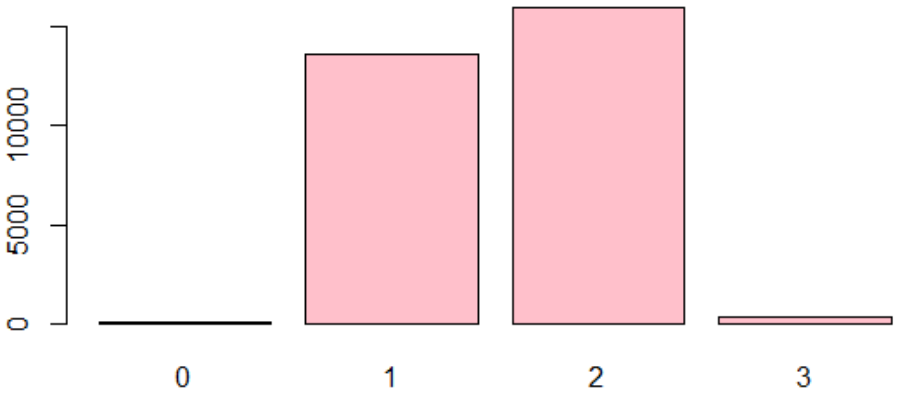


FIGURE 4 – Representation graphique de la variable situation familiale

Dans les variables de **X6 à X11 : Historique de paiements passés du mois d'avril à septembre 2005** on a fait une étude sur la variable X6 : septembre, X7 : août et X11 : Avril, les résultats sont les suivants :
Pour le mois de septembre (X6), on a 5686 payés (-1), 3688 payés avec un mois de retard(1), 2667 payés avec deux mois de retard (2).
Pour le mois d'août (X7), on a 6050 payés (-1), 3927 payés avec deux mois de retard(2).
Pour le mois d'avril (X11), on a 5740 payés(-1), 2766 payés avec deux mois de retard(2).

Pour la variable **X1 : montant du crédit** varie d'une valeur minimale de 10000 dollars à une valeur maximale de 1000000 dollar avec une moyenne de 167484 dollars.

Pour la variable **X5 : age**, on a l'âge minimal est 21 ans tandis que l'âge maximal est 79 ans avec un âge moyen de 35 ans.

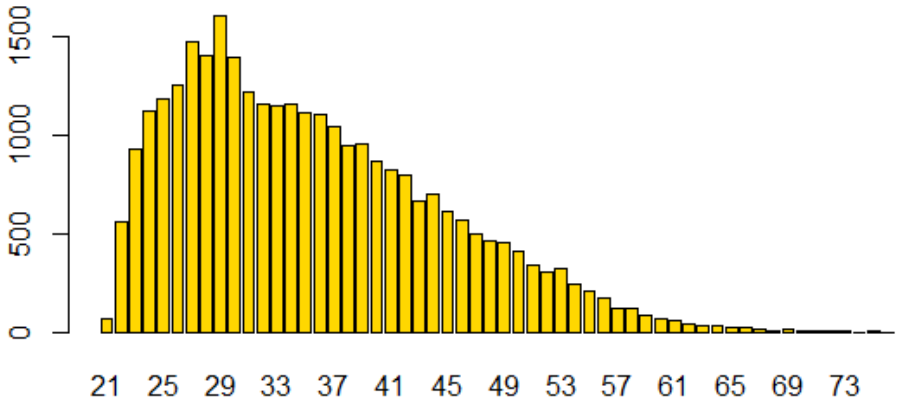


FIGURE 5 – Representation graphique de la variable age

Dans les variables de**X12 à X17 : relevé de facturation** durant les mêmes mois, du mois d'avril à septembre 2005, on a fait une étude sur la variable X12 : septembre, X13 août et X17 : avril , les résultats sont les suivant :
Pour le mois de septembre **(X12)**, on a un minimum de -165580 dollars et un maximum de 964511 dollars avec une moyenne de 51223 dollars.
Pour le mois d'août **(X13)**, on a un minimum de -69777 dollars et un maximum de 983931 dollars avec une moyenne de 49179 dollars.
Pour le mois d'avril **(X17)**, on remarque un minimum de -339603 dollars et un maximum

de 961664 dollars, la moyenne est de 38872 dollars.

Dans les variables de**X18 à X23 : Montant de paiements précédents** , on a fait une étude sur la variable X18 : septembre, X19 : août et X23 avril, les résultats sont les suivants :
Pour le mois de septembre (X18), on constate que le montant varie de 0 dollar à 873552 dollars avec une moyenne de 5664 dollars.
Pour le mois d'août (X19), on voit que le montant varie de 0 dollar à 1684259 dollars avec une moyenne de 5921 dollars.
Enfin, pour le mois d'avril, on a le montant minimal est de 0 dollar et le montant maximal est de 528666 dollars avec une moyenne de 5215,5 dollars.

On a Y c'est le défaut de paiement, si $Y = 1$ il y a défaut de paiement, si $Y = 0$ il n'y a pas défaut de paiement, on remarque que le nombre de défaut de paiement est de 6636 tandis que 23364 n'ont pas de défaut de paiement.

On remarque d'après la fonction str qu'on a d'une part 9 variables explicatives qualitatives qui sont sous forme de factor : genre, éducation, situation.familiale, et les variables de X6 à X11 ; d'autre part on a 14 variables explicatives quantitatives qui sont sous forme de num : montant.crédit, age et les variables de X12 à X23 ; de plus, notre variable Y Boolean qui est sous forme de factor.

On commence notre analyse par la variable :
genre :

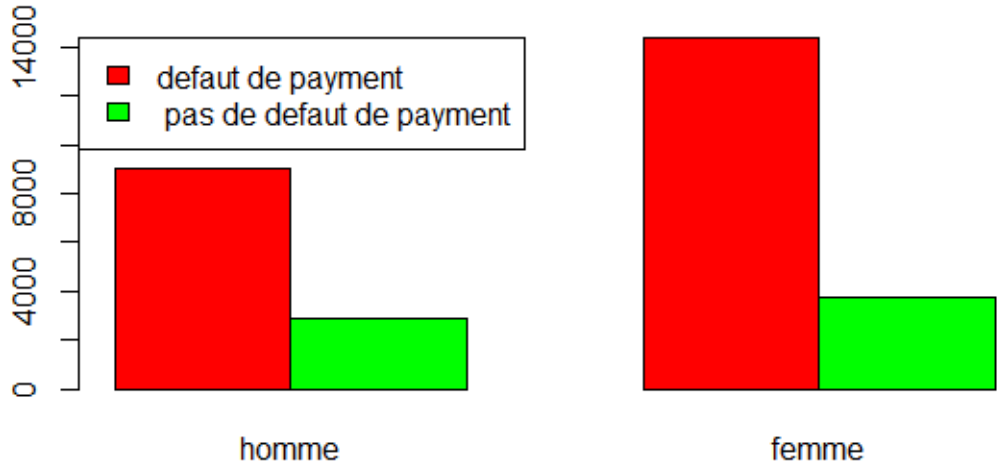


FIGURE 6 – Représentation du defaut de payement entre les femmes et les hommes

On remarque d'aprés le graphe ci-dessus que le nombre de "défaut de paiement" des femmes qui est de 14349 est supérieur à celui des hommes qui est de 9015 avec une différence de 5334 entre les deux sexes, de même le nombre de "pas de défaut de paiement" des femmes de 3763 est supérieur à celui des hommes qui est de 2873 avec une différence de 890 entre les deux.
Ce qui est logique puisque le nombre de femmes dans notre étude est plus grand que celui des hommes.

education :

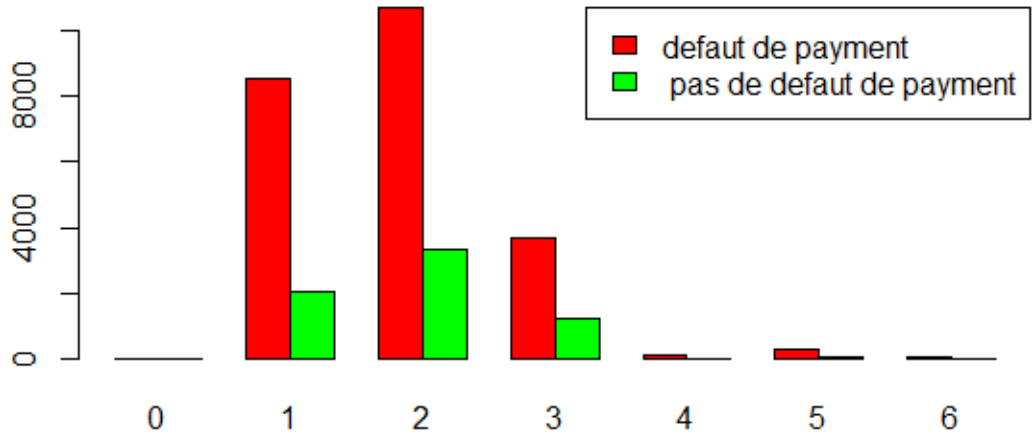


FIGURE 7 – Représentation de defaut de payement par rapport a l'éducation

On remarque que le nombre de "défaut de paiement" le plus grand est celui des étudiants universitaires avec un nombre de 10700 étudiants suivi de 8549 étudiants en secondaire puis de 3680 étudiants en supérieur et enfin 116 des autres étudiants.
De même, le nombre de "non défaut de paiement" pour les étudiants universitaires est le plus élevé avec un nombre de 3330 étudiants suivi de 2036 étudiants en secondaire puis de 1237 étudiants en supérieur et enfin 7 des autres étudiants.

situation familiale :

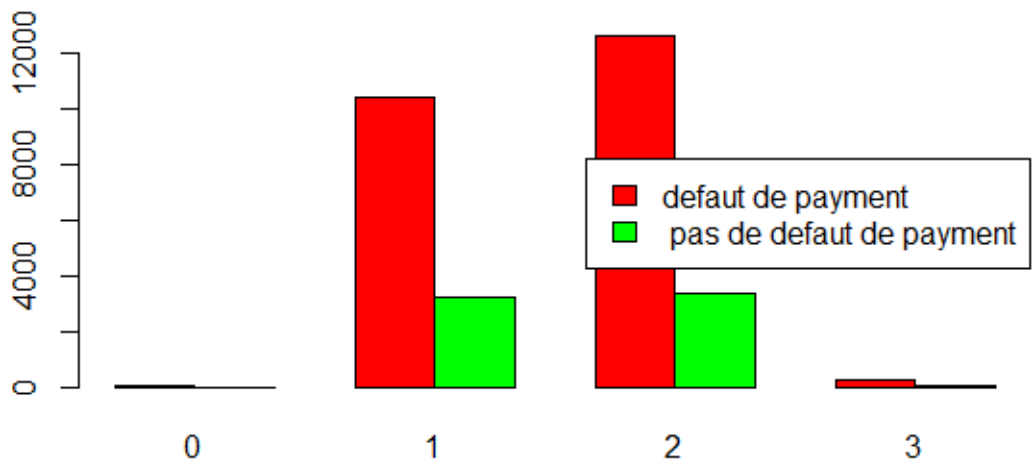


FIGURE 8 – Représentation de defaut de payement par rapport a la situation familiale

On remarque que le nombre de "défaut de paiement" des célibataires est légèrement supérieur à celui des mariés et largement supérieur à celui des personnes avec d'autres types de situations familiales ; 12623 célibataires ont un défaut de paiement suivi de 10453 de mariés et 239 autres.
De même, on remarque que le nombre de "non défaut de paiement" est légèrement plus élevé pour les célibataires que pour les mariés et largement supérieur à celui des personnes avec d'autres situations familiales, on trouve 3341 célibataires n'ont pas un défaut de paiement suivi de 3206 mariés et 84 autres.

1.2 Corrélation entre les variables

On va regarder s'il y'a une corrélation entre les variables :

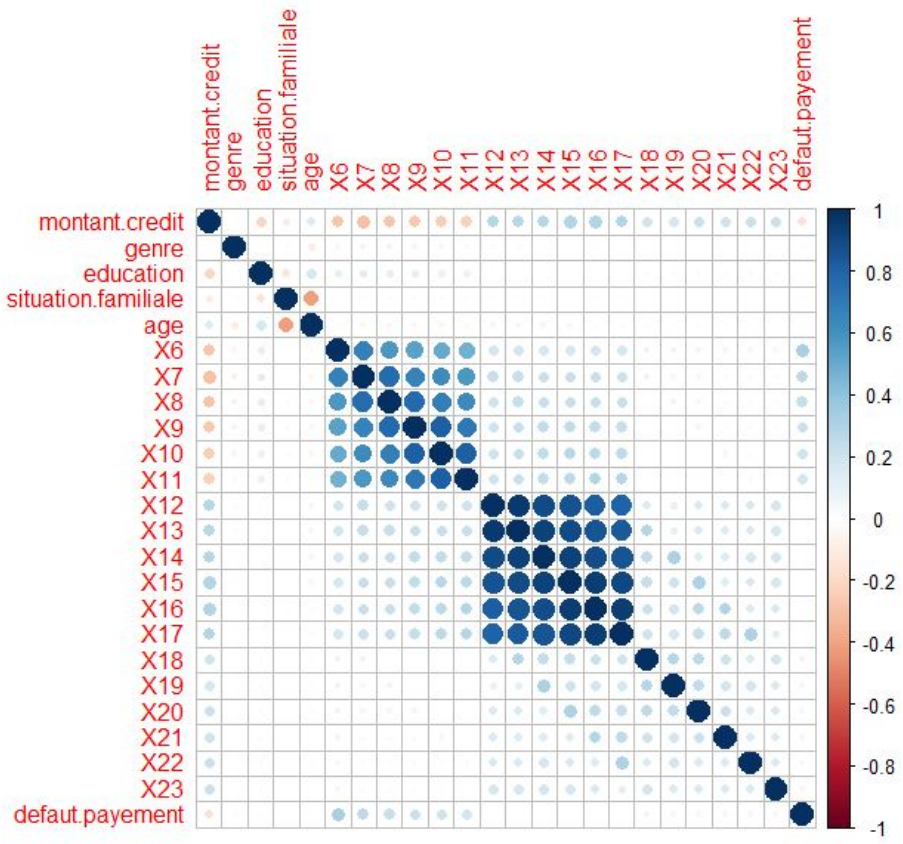


FIGURE 9 – Représentation de la correlation entre les variables

En effet, on remarque une forte corrélation positive entre les variables explicatives quantitatives, entre les variables de X12 à X17, qui sont les variables constituant les relevés de facturation sur les derniers mois, du mois d'avril au mois de septembre 2005, la corrélation est d'autant plus forte entre les mois successives par exemple entre X12 et X13 ou X16 et X17.

On remarque aussi une corrélation positive entre les variables explicatives qualitatives, entre les variables de X6 à X11, une corrélation moins forte que celle des variables explicatives quantitatives, ces variables représentent l'historique de paiements passés sur les derniers mois, du mois d'avril au mois de septembre 2005, cette corrélation est plus forte entre les mois successives par exemple entre X10 et X11.

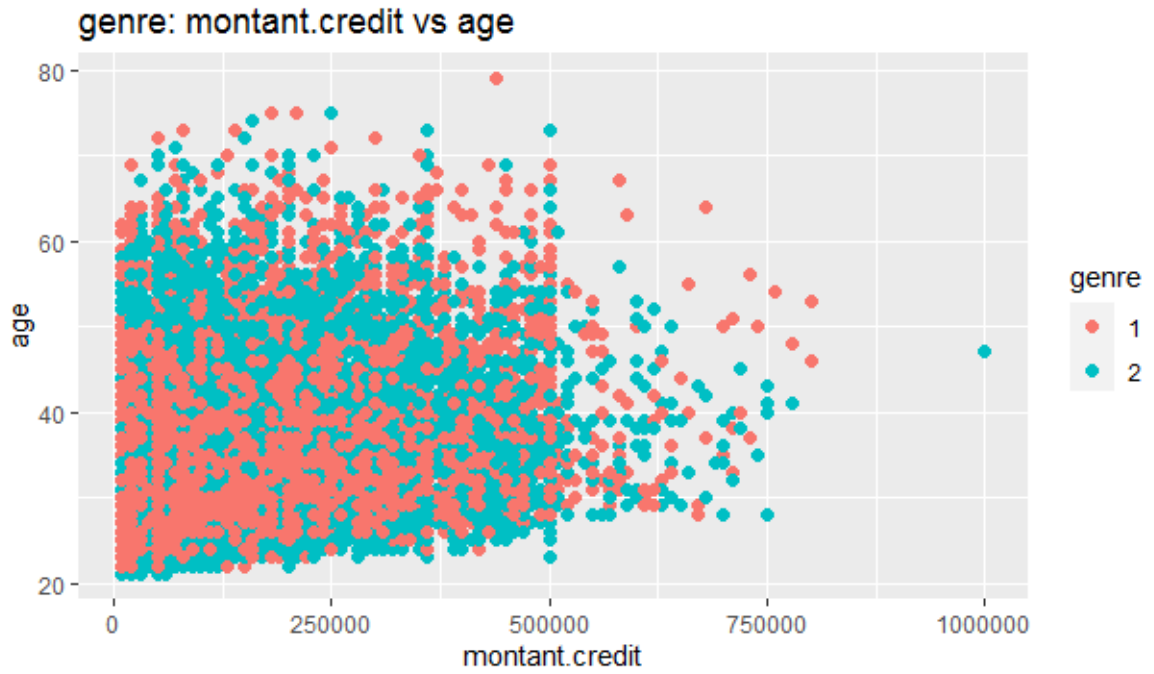


FIGURE 10 – Représentation graphique du montant.credit en fonction de l'age de chaque genre

On remarque que plus l'âge augmente plus le montant de crédit diminue ; entre 21 et 60 ans on voit que les deux sexes ont un montant de crédit de 0 dollar à 500000 dollars. néanmoins on trouve des outliers par exemple la valeur 2 = femme avec un montant de crédit de 1000000 ou valeur 1 = homme à 80 ans avec un montant de crédit près de 500000.

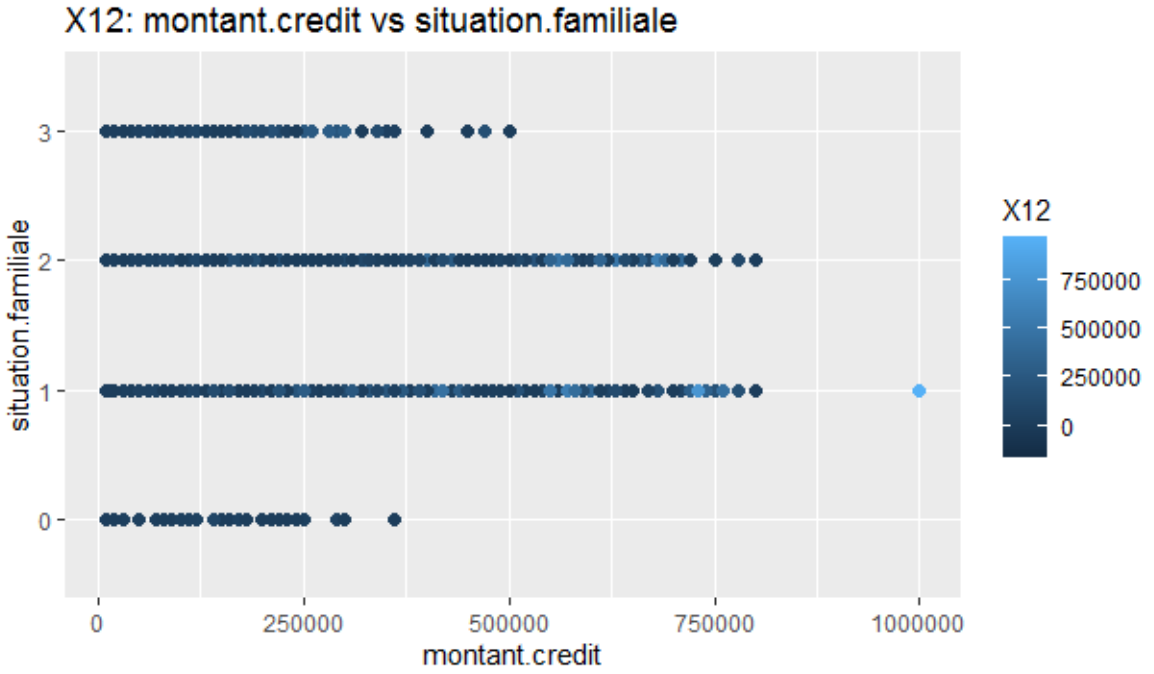


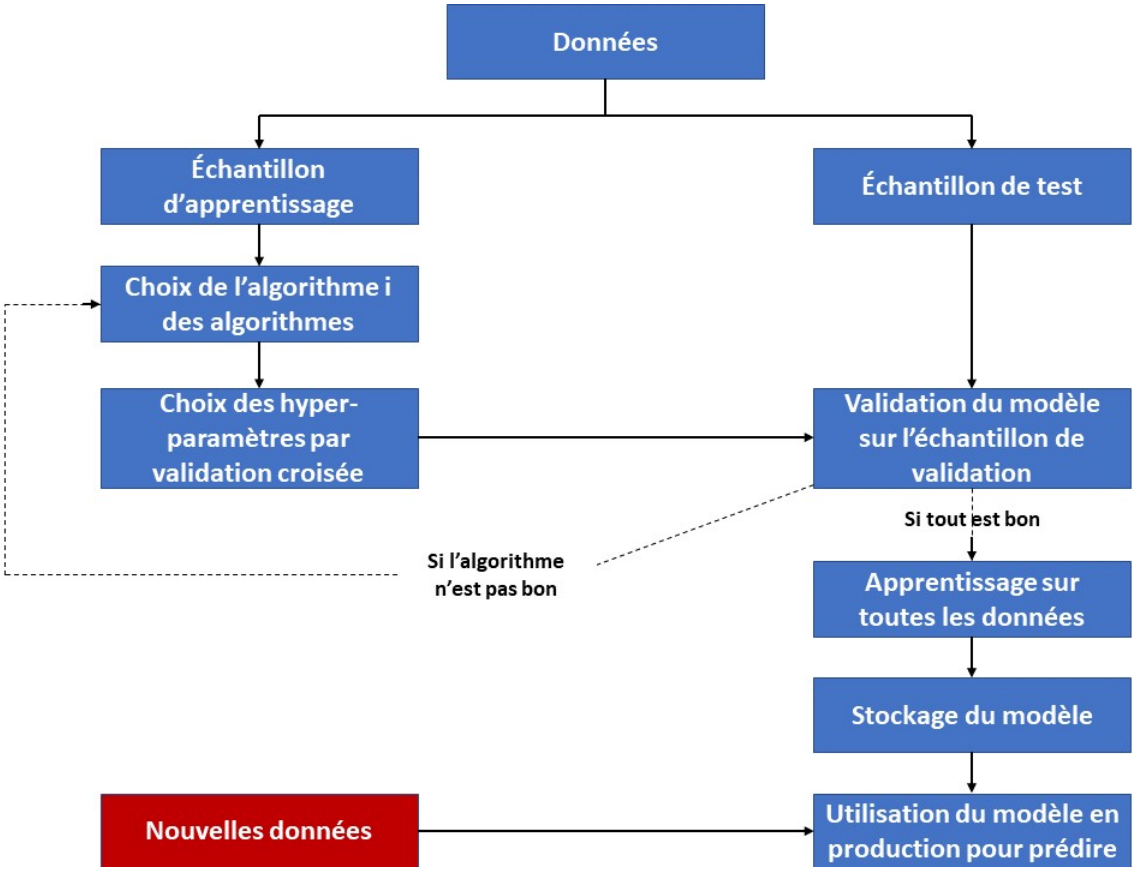
FIGURE 11 – Représentation graphique du montant.credit en fonction de la situation familiale du relevé de facturation du mois de septembre

On remarque dans ce graphique que dans le mois de septembre 2005 les célibataires et les mariés ont le montant de crédit le plus haut, ils sont alors facturés au même montant.

2 Algorithmes de classifications

2.1 Le processus de traitement en machine learning

Une fois que nous avons construit notre jeu de données (question1 au dessus), le processus est décrit dans la figure ci-dessous :



2.2 Construction de l'échantillon test et apprentissage

Pour la séparation, on utilise la fonction train_test_split() de Scikit-Learn, cette fonction permet de créer automatiquement autant de structures que nécessaire à partir de nos données. Elle utilise une randomisation des individus et ensuite une séparation en fonction d'un paramètre du type test_size, la taille de l'échantillon de test sera dans notre cas 30% de la taille du jeu de données initial car notre jeu de données est grand alors il on doit prendre un échantillon de test grand afin de bien tester la robustesse de notre modèle.Voilà le code en python :

```
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 27 02:05:50 2021

@author: khali
"""
```

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
data = pd.read_excel("C:/Users/khali/Downloads/default(1).xls")
data1=data.rename(columns={'X1':'m.credit','X2':'genre','X3':'
    education','X4':'sit.familiale','X5':'age','X6':'h.p.9','X7':'h.p
    .8','X8':'h.p.7','X9':'h.p.6','X10':'h.p.5','X11':'h.p.4','X12':'
    r.f.9','X13':'r.f.8','X14':'r.f.7','X15':'r.f.6','X16':'r.f.5','
    X17':'r.f.4','X18':'m.p.9','X19':'m.p.8','X20':'m.p.7','X21':'m.p
    .6','X22':'m.p.5','X23':'m.p.4'})
data2 = data1.drop('Y', axis = 1)
target = data1.Y

# In[1]: Base Test et base apprentissage

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data2, target,
    test_size=0.3)
```

Listing 1 – Sample Code Listing Python

2.3 Choix de l'algorithme et de l'hyperparamètres

Le choix des algorithmes de machine learning adaptés est un point clé.

Ce choix se fait sur un certain nombre de propriétés, principalement liées à notre données et à l'utilisation qui sera faite de cet algorithme.

Pour l'ajustement des hyperparamètres d'un modèle on doit utiliser Scikit-Learn qui propose une classe GridSearchCV permettant d'implémenter cette recherche d'hyperparamètres.

Nous allons faire varier les hyperparamètres de ces modèles pour trouver la meilleure combinaison en termes d'aire sous la courbe ROC.

1- K plus proches voisins : C'est une méthode de classification où il s'agit de prédire une variable qualitative (cas de notre données), cette méthode est basée sur la proximité entre les observations des données. cette algorithme est dit paresseux car son apprentissage est très rapide, c'est la partie prédiction qui est plus lente. on doit choisir la distance et le nombre de voisins comme hyperparamètres.

```
# In[3]:  kpp voisins
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
parameters = {'n_neighbors': list(range(1,50,2)),"weights":['uniform
    ','distance']}
# La fonction GridSearchCV automatise la recherche  d un  optimum
    parmi les hyperparam tre , elle utilise notamment la validation
    crois e .
knn = KNeighborsClassifier()
gridknn = GridSearchCV(knn, parameters, scoring="roc_auc",cv=5)
gridknn.fit(X_train, y_train)
```

```
gridknn.best_params_
```

Listing 2 – Sample Code Listing Python

2- Arbres de décision : C'est une méthode de classification permettant de construire un arbre de classification. Les arbres de décision sont des outils pratiques pour la visualisation des résultats. Ils sont néanmoins complexes à paramétrer et fortement soumis au sur-apprentissage.

on doit choisir le type de la fonctions d'impureté (Gini ou Shannon), max_depth, min_samples_split et min_samples_leaf comme hyperparamètres.

on trouve comme resultat les hyperparamètres (criterion="entropy",max_depth=5,min_samples_leaf=4,min_samples_split=5).

```
# In[4]: decision_tree

from sklearn.tree import DecisionTreeClassifier, plot_tree
param={"criterion":["gini","entropy"],"max_depth":list(range(1,10)),
    "min_samples_split":range(2,10),"min_samples_leaf":range(1,5)}
tree= GridSearchCV(DecisionTreeClassifier(),param,scoring="roc_auc",
    cv=5)

tree.fit(X_train, y_train)
tree.best_estimator_
```

Listing 3 – Sample Code Listing Python

3- Forêts aléatoires : C'est une méthode de classification de la famille des méthodes d'agrégation (ensemble) basée sur la multiplication des arbres de décision. C'est une méthode très stable et rapide, surtout en environnement parallélisé.

on doit choisir le n_estimators (n échantillons Bootstrap) et le max_depth comme hyperparamètres.

on trouve comme resultat les hyperparamètres (n_estimators=1000,max_depth=9).

```
# In[5]: foret_aleatoire

from sklearn.ensemble import RandomForestClassifier

param_rf={"max_depth":list(range(9,20)),"n_estimators"
    :[500,1000,2000,3000]}
modele_grid=GridSearchCV(RandomForestClassifier(),param_rf,scoring="
    roc_auc",cv=5)
modele_grid.fit(X_train, y_train)
modele_grid.best_params_
```

Listing 4 – Sample Code Listing Python

4- Modèle de mélange Gaussien : C'est une méthode de classification basée sur hypothèse de marginales gaussiennes, adapté si les variables sont continues (et un peu plus). Si on pense que le modèle est homoscedastique (c'est notre cas) alors il faut faire l'analyse discriminante linéaire, sinon on fait une analyse discriminante quadratique.

Cette méthode est sensible aux individus extrêmes.

on doit choisir le solver comme hyperparamètres, on trouve comme resultat le hyperparamètre solver="svd".

```
# In[6]: methode MMG

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
gridMMG = dict()
gridMMG['solver'] = ['svd', 'lsqr', 'eigen']
modele_gridMMG=GridSearchCV(LinearDiscriminantAnalysis(),gridMMG,
    scoring="roc_auc",cv=5)
modele_gridMMG.fit(X_train, y_train)
modele_gridMMG.best_params_
```

Listing 5 – Sample Code Listing Python

5- Régression logistique : C'est une méthode de classification pour deux classes ou plus permettant de modéliser la probabilité d'appartenance à une classe, elle est plutôt orienté vers l'explication que la prédiction.

on doit choisir le solver, le penalty et C parameter comme hyperparamètres.

on trouve comme resultat les hyperparamètres (solver="liblinear",penalty="l1",C=0.01).

```
# In[7] Regression logistique

from sklearn.linear_model import LogisticRegression
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['none','l1','l2','elasticnet']
c_values = [100, 10, 1.0, 0.1, 0.01]
gridlogit = dict(solver=solvers,penalty=penalty,C=c_values)
modele_gridlogit=GridSearchCV(LogisticRegression(),gridlogit,scoring
    ="roc_auc",cv=5)
modele_gridlogit.fit(X_train, y_train)
modele_gridlogit.best_params_
```

Listing 6 – Sample Code Listing Python

6- Gradient boosting machine (GBM) : C'est une méthode de classification de la famille des méthodes d'agrégation (ensemble) basée sur l'enchaînement de classifieurs simples améliorant la qualité du classifieur global.

les hyperparamètres permettent de gérer le principal problème de cette approche : le sur-apprentissage.

on doit choisir le n_estimators, le learning_rate, le subsample et le max_depth comme hyperparamètres.

on trouve comme resultat les hyperparamètres (n_estimators =200, learning_rate =0.1, max_depth=3, subsample =0.7).

```
# In[8]: Gradient boosting machine (GBM):
from sklearn.ensemble import GradientBoostingClassifier
n_estimators = [10, 100,200]
learning_rate = [0.001, 0.01, 0.1]
subsample = [0.5, 0.7, 1.0]
```

```
max_depth = [3, 7, 9]
paramboost = dict(learning_rate=learning_rate, n_estimators=
    n_estimators, subsample=subsample, max_depth=max_depth)
grid_boost = GridSearchCV(GradientBoostingClassifier(),paramboost,
    cv=5, scoring="roc_auc")
grid_boost.fit(X_train, y_train)
grid_boost.best_params_
```

Listing 7 – Sample Code Listing Python

2.4 Les indicateurs pour valider un modèle

La partie validation d'un modèle d'apprentissage est extrêmement important. L'objectif d'un modèle d'apprentissage est de prédire une valeur la plus proche possible de la réalité. les indicateurs de qualité du modèle qu'on doit utiliser ici sont **la courbe ROC** et **l'aire sous la courbe ROC**.

la courbe ROC : lorsque les classes sont fortement déséquilibrées on ne peut pas compter sur la matrice de confusion pour la validation du modèle car cette matrice devient dure à interpréter. La courbe ROC est là pour combler ce défaut. Elle est en fait la proportion de vrais positifs en fonction de la proportion de faux positifs.

Il s'agit en fait de représenter le rappel (recall) en fonction de (1-spécifité) sur une courbe en faisant varier le seuil de classification (C'est-à-dire le point à partir duquel une observation est considérée comme positive).

l'aire sous la courbe ROC : La courbe ROC est un indicateur important mais on préfère souvent une valeur plutôt qu'une courbe afin de comparer nos modèles. Pour cela, on utilise l'aire sous la courbe ROC (AUC). Cette aire est calculée directement à partir de la courbe ROC. Ainsi, un modèle aléatoire aura une AUC de 0.5 et un modèle parfait aura une AUC de 1.

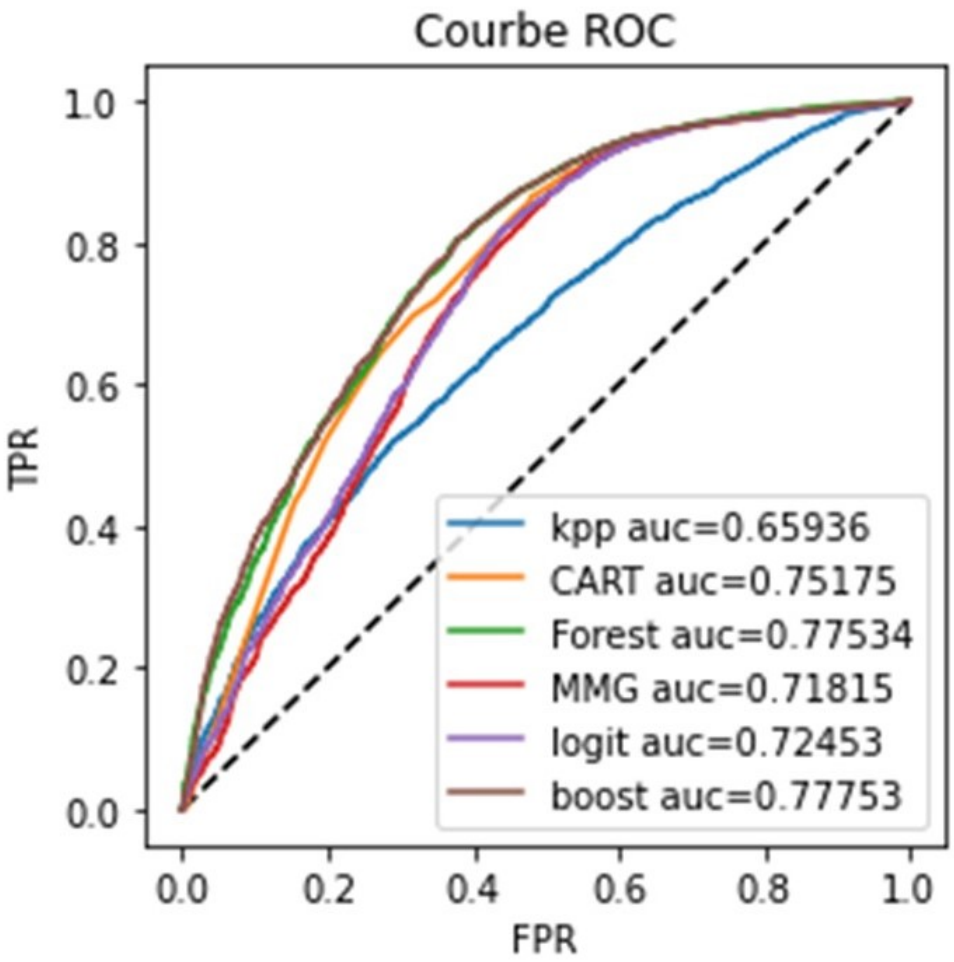
nous allons faire maintenant le courbe ROC Pour nos six modèles :

```
# In[3]:  kpp voisins

knn = KNeighborsClassifier()
gridknn = GridSearchCV(knn, parameters, scoring="roc_auc",cv=5)
gridknn.fit(X_train, y_train)
gridknn.best_params_
probas_kpp = gridknn.predict_proba(X_test)
fpr0, tpr0, thresholds1 = roc_curve(y_test, probas_kpp[:, 0],
    pos_label = gridknn.classes_[0] ,  drop_intermediate=False)
auc_kpp = auc(fpr0, tpr0)

# In[4]: decision_tree

clf = DecisionTreeClassifier(criterion="entropy",max_depth=5,
    min_samples_leaf=4,min_samples_split=5)
clf.fit(X_train, y_train)
clf.predict(X_test)
plt.figure(figsize=(20,20))
plot_tree(clf)
plt.show()
```

```
probas1 =clf.predict_proba(X_test)
fpr1, tpr1, thresholds0 = roc_curve(y_test,probas1[:, 0], pos_label=
    clf.classes_[0], drop_intermediate=False)# A REMPLIR
fpr1.shape
auc_tree = auc(fpr1, tpr1)

# In[5]: foret_aleatoire

forest = RandomForestClassifier(n_estimators=1000,max_depth=9)
forest.fit(X_train,y_train)
forest.score(X_test, y_test)
probas2 = forest.predict_proba(X_test)

fpr2, tpr2, thresholds2 = roc_curve(y_test, probas2[:, 0], pos_label
    =forest.classes_[0], drop_intermediate=False)


auc_for = auc(fpr2, tpr2)

# In[6]: methode MMG

lda = LinearDiscriminantAnalysis(solver="svd")
lda.fit(X_train, y_train)
lda.score(X_test,y_test)
probas3 = lda.predict_proba(X_test)
fpr3, tpr3, thresholds3 = roc_curve(y_test, probas3[:, 0], pos_label
    =lda.classes_[0], drop_intermediate=False)
auc_lda = auc(fpr3, tpr3)

# In[7] Regression logistique

SK_logit = LogisticRegression(max_iter =1000, penalty='l1',
    fit_intercept=True,solver='liblinear',C=0.01)
SK_logit.fit(X_train, y_train)
SK_logit.score(X_test,y_test)
probas4 = SK_logit.predict_proba(X_test)
fpr4, tpr4, thresholds4 = roc_curve(y_test, probas4[:, 0], pos_label
    =SK_logit.classes_[0], drop_intermediate=False)
fpr0.shape

auc_SK = auc(fpr4, tpr4)

# In[8]: Gradient boosting machine (GBM):

boost = GradientBoostingClassifier(n_estimators=200,learning_rate
    =0.1,max_depth=3,subsample=0.7)
boost.fit(X_train, y_train)
boost.score(X_test,y_test)
probas5 = boost.predict_proba(X_test)
fpr5, tpr5, thresholds5 = roc_curve(y_test, probas5[:, 0], pos_label
    =boost.classes_[0], drop_intermediate=False)
fpr0.shape

auc_boost = auc(fpr5, tpr5)
```

Listing 8 – Sample Code Listing Python

Donc pour nos six modèles, nous avons :



Donc d'après le courbe ROC on choisit le modèle ayant la plus grand aire sous la courbe ROC (auc), donc le meilleur modèle c'est le modèle Gradient boosting machine (GBM) (boost auc =0.77753) avec les hyperparamètres (n_estimators =200, learning_rate =0.1, max_depth=3, subsample =0.7).

On peut encore tracer le tableau de l'erreur de prévision pour chaque modèle :

modèle	L'erreur de prédiction (score)
kpp	21,67%
Arbre de décision	17,61%
Foret aleatoire	17,62%
MMG	18,16%
Regression logistique	18,68%
Gradient boosting machine (GBM)	17,45%

Remarque : le code dans ce pdf Ce n'est pas le code complet, il faut voir le fichier python et le fichier R .

Pour calculer l'erreur de prévision de chaque modèle, il faut calculer au debut la matrice de confusion M (voir mon notre fichier python), puis pour calculer l'erreur de prévision de ce modèle il faut calculer 1 -(D[0,0]+D[1,1])/(taille de l'echantillon test).

3 Conclusion

En somme, dans ce projet on a fait une statistique descriptive où on a montré la corrélation entre les variables afin de bien analyser la base de données et comprendre les différentes variables explicatives ensuite on a fait tourner sur les données les algorithmes de classifications vus en cours qui nous semblent pertinents et on a trouver que gradient adaboost machine est la meilleure méthode.