PRÉDICTION DE DÉFAUTS DE PAIEMENT

Mahmoud JAHA

Université Côte d'Azur

Sous la direction du Pr Nicolas PASQUIER

OBJECTIF DU PROJET

Ce projet de data mining consiste à identifier un classifieur optimal qui nous permettrait de prédire un défaut de paiement de clients ayant contracté un prêt. Notre ensemble de données est composé de plusieurs informations considérées comme importantes sur 1500 clients. Nous allons essayer de créer plusieurs types de classifieurs, de les tester, de les comparer et enfin de conclure en choisissant celui dont la prédiction est la plus juste. L'objectif final est de réussir à réduire les coûts éventuels qu'impliquent l'accord de prêt à des clients susceptibles d'être en défaut de paiement.

Exploration des données

Dictionnaire des données

Variable	Type	Description	Domaine de valeurs
branch	Catégoriel	Code de la branche d'activité du client.	{3, 13, 15, 20, 25, 49, 60, 64, 68, 73, 74, 75, 76, 77, 91}
ncust	Entier	Nombre de clients dans la branche d'activité.	[1919, 4809]
customer	Entier	Numéro unique d'identification du client	[10010, 453800]
age	Entier	Age en nombre d'années	[18, 79]
ed	Catégoriel	Niveau d'éducation relativement au baccalauréat	{Niveau bac, Bac+2, Bac+3, Bac+4, Bac+5 et plus}
employ	Entier	Nombre d'années avec l'employeur actuel	[0, 63]
address	Entier	Nombre d'années à l'adresse actuelle	[0, 34]
income	Réel	Revenus du foyer en milliers de $	[12.0, 1079.0]
debtinc	Réel	Débit carte de crédit en milliers de $	[0.0, 40.7]
creddebt	Réel	Ratio Débit/Crédit (x100)	[0.00, 35.97]
othdebt	Réel	Autres dettes en milliers de $	[0.00, 63.47]
default	Booléen	Un défaut de payement a-t-il eu lieu ? Variable de classe.	{Oui, Non}

Ces 12 variables ne sont pas toutes utiles pour notre problème, il faut procéder à un nettoyage et à une classification de ces variables.

Paramétrage de l'apprentissage du classifieur :

- 1 variable cible : **default** variable de classe à prédire.
- 10 variables prédictives : **age, employ, address, etc** décrivant les caractéristiques testées.
- 1 variable ignorée : **customer** est le numéro d'identification client, cette information est inutile pour le classifieur.

Cette première étape de paramétrage nous permet de mettre en évidence l'inutilité de certaines informations dans un processus de prédiction algorithmique, ici l'identifiant du client. En effet, moins le modèle a de variables, plus la prédiction sera rapide et efficace. Il vaut donc mieux supprimer les variables "parasites". Par analogie, on peut aller plus loin en identifiant des relations d'égalité ou de corrélation entre plusieurs variables et en les ajustant pour réduire le nombre de variables totales. Ici, nous pouvons par exemple constater que la variable **branch** et **ncust** sont quasi-similaires, c'est-à-dire qu'une des variables donne une information que l'autre confirme toujours. On pourra alors en supprimer une des deux de nos ensembles de données sans "détruire" de l'information.

Fichiers de données

Fichier	Nbr instances	Classe?	Remarques
`Data Projet.csv`	1200	Oui	Instances dont la classe réelle est connue
`Data Projet New.csv`	300	Non	Instances à prédire

Dans un premier temps, la base de données que nous utiliserons est constituée des informations de 1200 clients ayant eu recours à un prêt avec la confirmation ou non de l'existence d'un défaut de paiement pour chaque client.

Dans un second temps nous utiliserons une base de données de 300 clients constituée des mêmes variables que la précédente, hormis l'information sur l'existence d'un défaut de paiement.

Cet ensemble de données est appelé “l’ensemble de test” et nous permettra de tester nos classifieurs.

Méthode d'évaluation des classifieurs

Dans le cadre de ce projet :

- Les clients prospects prédit dans la classe **default=False** recevront un accord pour emprunter. => classe positive
- Les clients prospects prédit dans la classe **default=True** ne recevront pas d’accord pour emprunter. => classe négative

L’objectif à terme est de réduire le coût des mauvaises décisions dans l’accord de prêt. Ainsi l’évaluation de la pertinence des classifieurs doit prendre en compte deux éléments :

1. Un taux de succès général assez élevé
2. La minimisation du taux de Faux Positifs ou en d’autres termes la proportion d'exemples négatifs incorrectement classés.

Ces éléments requièrent l’utilisation de la courbe ROC, d’AUC mais aussi d’une matrice de confusion.

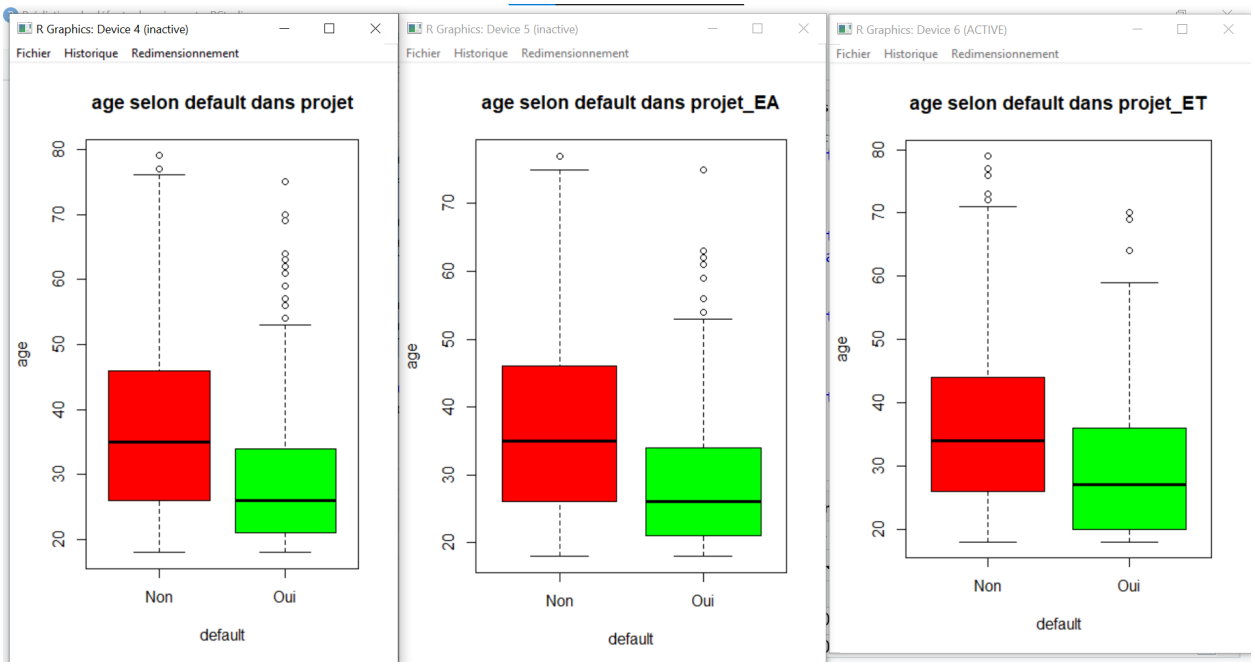
Création des ensembles d’apprentissage et de test

Pour créer les deux ensembles de données essentiels à la création des classifieurs, nous allons les extirper de la base de données “Data Projet.csv”.

Tout d’abord nous effectuons un échantillonnage aléatoire de la base de données initiale notée “projet” car nos données sont rangées selon le code de la branche d’activité du client.
Nous extirpons ensuite les 800 premières lignes de cette base échantillonnée afin de créer l’ensemble d’apprentissage “projet_EA” constitué du ⅔ de la data frame “projet”.
Puis nous créons “projet_ET” avec les 400 dernières lignes de cette même base initiale échantillonnée. Cet ensemble de données est appelé “l’ensemble de test” et nous permettra de tester nos classifieurs.

Vérification de pertinence des ensembles d'apprentissage et de test

On peut vérifier si les deux ensembles créés respectent les mêmes caractéristiques, afin d'être certains d'avoir des échantillons assez semblables. On peut à cet effet, comparer les distributions des classes et comparer les proportions des classes pour les valeurs des variables prédictives.



On voit ici, par exemple, que nos deux ensembles ont quasiment les mêmes proportions de la classe "default" pour la variable "age" par rapport à l'ensemble initial.

Création d'une nouvelle variable

Après observation de tous les nuages de points possibles, j'ai repéré une certaine relation entre debtinc et employ au regard de la classe default. J'ai essayé de créer une variable Quotient_debtinc_employ qui me semble être pertinente car un fort débit de carte est plus aisé à contrôler lorsqu'on a un emploi stable depuis plusieurs années, et les imprévues sont plus faciles

à amortir. Nous allons donc désormais tester nos classifieurs sur deux bases de données distinctes: celle de départ et une autre avec le **quotient debtinc/employ**.

Construction et évaluation des classifieurs

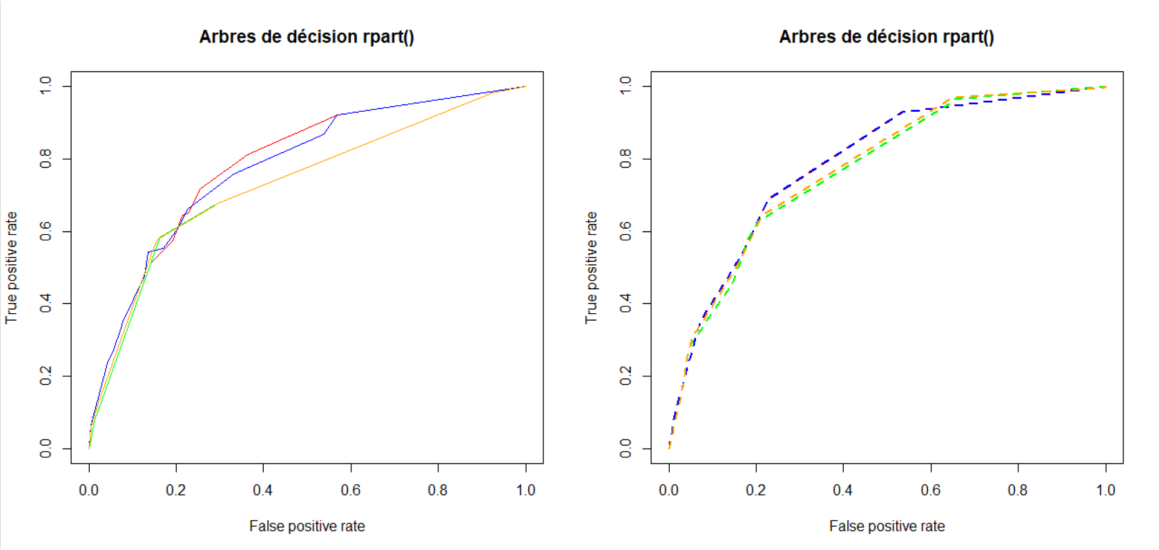
Pour augmenter nos chances de générer le meilleur classifieur, nous allons construire plusieurs types de classifieurs différents, selon plusieurs paramètrages distincts.

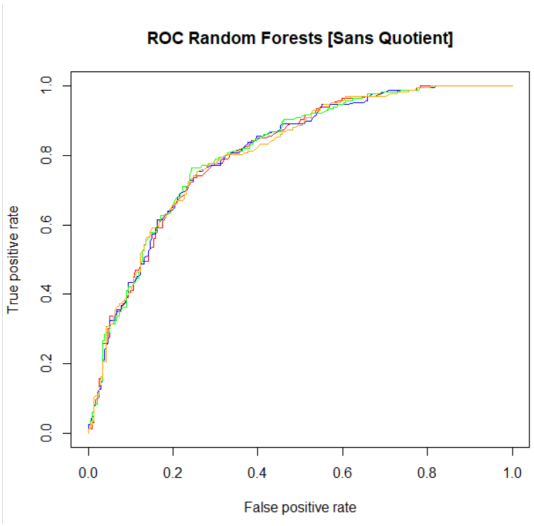
Voici les classifieurs supervisés que nous allons utiliser :

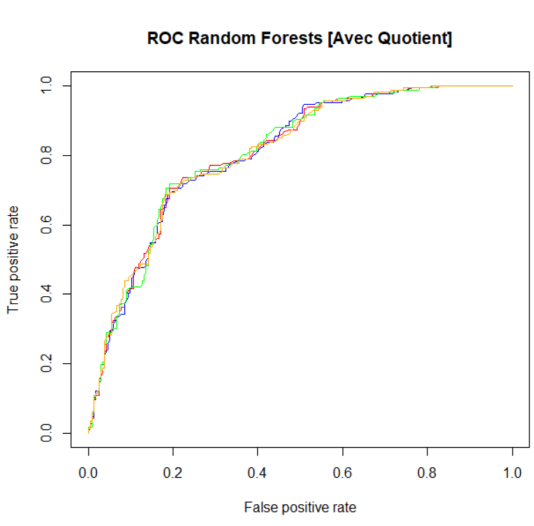
1. Arbres de décisions
2. Forêts d'arbres décisionnels aléatoires
3. K plus proches voisins
4. Support vector machines
5. Naive Bayes
6. Réseaux de neurones

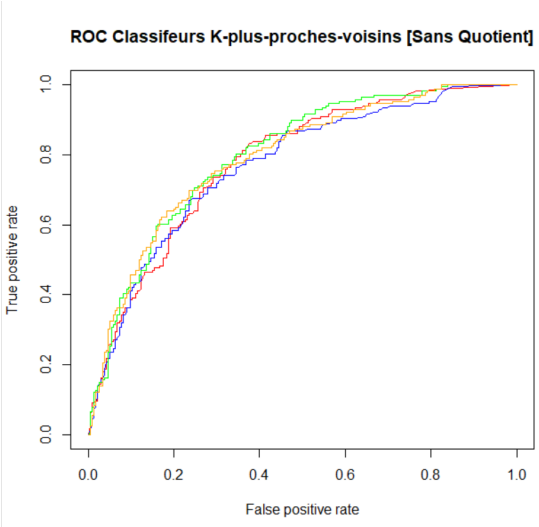
Pour aller plus vite dans la construction et l'évaluation de nos classifieurs, il est essentiel de créer des fonctions pour chacune des méthodes. La première étape de notre évaluation se porte donc sur les performances générales des classifieurs, à cet effet nous utiliserons donc le ROC et nous calculerons l'AUC (voir code R).

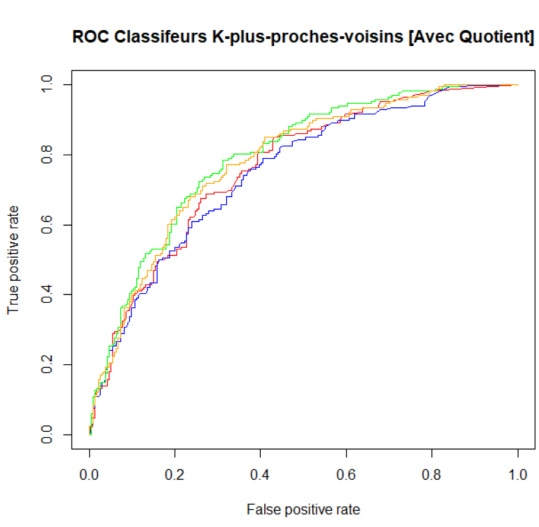
Voici les ROC de plusieurs types de classifieurs selon leur paramétrage :

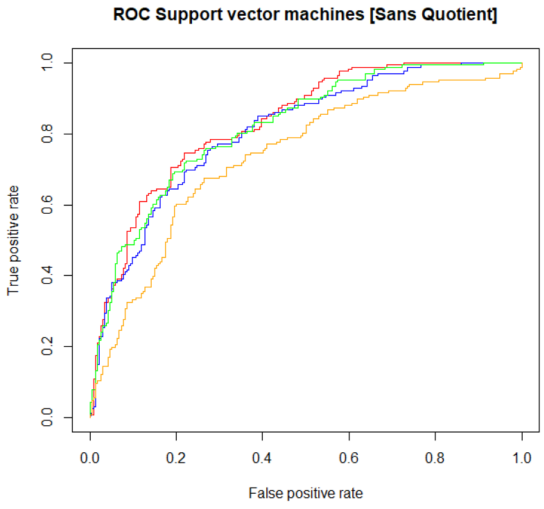


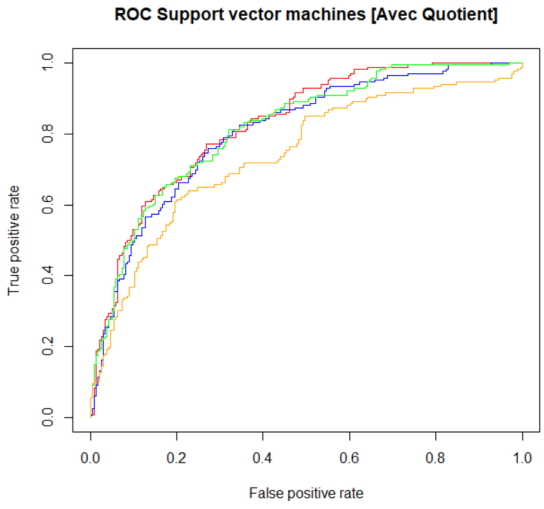

ROC Random Forests [Sans Quotient]
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

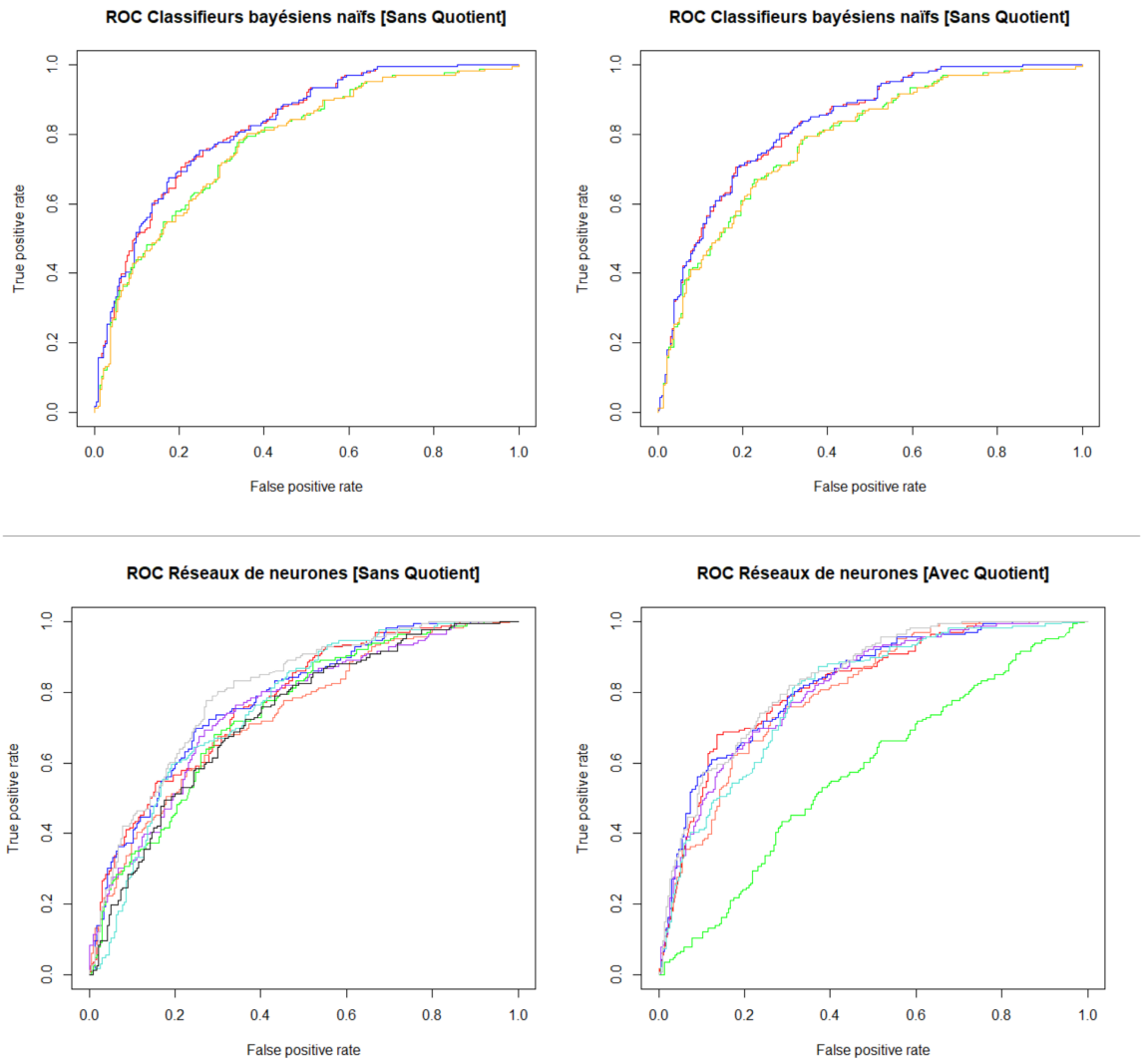

ROC Random Forests [Avec Quotient]
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Classifeurs K-plus-proches-voisins [Sans Quotient]
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Classifeurs K-plus-proches-voisins [Avec Quotient]
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Support vector machines [Sans Quotient]
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0


ROC Support vector machines [Avec Quotient]
True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0



A partir de ces courbes ROC et le calcul du AUC, on peut identifier pour chaque type de classifieurs supervisés, le meilleur paramétrage. Ainsi, nous avons retenus ces paramètrage suivants :

1. Arbres de décisions

   test_Q_rpart("gini", 5, TRUE)

   AUC=0.7894

2. Forêts d'arbres décisionnels aléatoires

   test_rf(500, 3, TRUE)

   AUC=0.8117

3. K plus proches voisins

   test_knn(20, 1, TRUE)

   AUC=0.7990

4. Support vector machines

   test_svm("linear", FALSE)

   AUC=0.8318

5. Naive Bayes

   test_Q_nb(0, FALSE, FALSE)

   AUC=0.8302

   test_nb(0, FALSE, FALSE)

   AUC=0.8233

6. Réseaux de neurones

   test_Q_nnet(25, 0.001, 100, TRUE)

   AUC=0.8364

Remarque : Lorsqu'il y a un "Q" dans la fonction, c'est qu'on utilise l'ensemble de données avec la variable "Quotient debtinc/employ". Ce qu'on peut affirmer, c'est que la création de cette variable a amélioré les performances de certains types de classifieurs (amélioration particulièrement important pour les classifieurs à Réseaux de neurones) mais pour d'autres, cela n'a rien changé et dans quelques rares cas la variable a réduit les performances des classifieurs.

Après comparaison, on distingue 3 types de classifieurs avec les meilleures performances globales : 4)Support Vector Machine, 5)Naive Bayes, 6)Réseaux de neurones. (AUC>0.82)

Maintenant, analysons le taux de Faux Positifs pour chacun de ces 3 types de classifieurs afin de trouver le type de classifieur avec le plus faible taux de Faux Positifs :

```
> test_svm("linear", FALSE, "red")
     svm_class
      Non Oui
  Non 206  28
  Oui  65 101
AUC =  0.831891669241067
```

Taux de FP=65/(65+101)=0.391

```
> test_nb(0, FALSE, FALSE, "red")
     nb_class
      Non Oui
  Non 145  89
  Oui  29 137
AUC =  0.82339614869735
```

Taux de FP=29/(29+137)=0.174

```
> test_Q_nnet(25, 0.001, 100, TRUE, "grey")
     nnQ_class
      Non Oui
  Non 192  42
  Oui  60 106
```

Taux de FP=60/(60+106)=0.361

Ce qu'on aperçoit ici avec les matrices de confusion, c'est que les taux de faux positifs sont largement inférieurs pour le classifieur de type Naive Bayes que pour les deux autres (et cela, qu'importe le paramétrage). Rappelons que notre objectif est de minimiser le coût des prêts accordés à des "mauvais" clients prévisibles.

Par conséquent, le choix final se portera sur un classifieur de type Naive Bayes avec le paramétrage suivant :

- Valeur du lissage de Laplace (paramètre laplace) = 0
- Utilisation d'un noyau d'estimation des densités ou non (paramètre usekernel) = NON
- Ensembles d'apprentissage et de test sans la variable "Quotient debtinc/employ"

Résultats de l'application du classifieur sélectionné

Voici les résultats de la fonction summary() sur le data frame “projet_P” (tableau des données de test avec les instances prédites) :

```
Prediction
Non:134
Oui:166

Probabilité.Non  Probabilité.Oui
Min.   :0.0000   Min.   :0.00000
1st Qu.:0.2375   1st Qu.:0.06436
Median :0.3928   Median :0.60723
Mean   :0.5139   Mean   :0.48609
3rd Qu.:0.9356   3rd Qu.:0.76250
Max.   :1.0000   Max.   :1.00000
```

Conclusion

Ce projet de prédiction des défauts de paiement s’est révélé être extrêmement formateur, tant sur le point technique que théorique. La recherche du classifieur optimal est un exercice très rigoureux et assez chronophage. Nous avons testé plusieurs modèles d’algorithme de classification supervisée afin de trouver les meilleurs modèles, leur paramétrage optimal, et enfin identifier l’unique classifieur qui répond au mieux aux problèmes initiaux (ici, réduction des coûts dûs aux défaut de paiement des emprunts bancaires). Par conséquent, chaque étape à son importance, et négliger l’une d’elles peut vite rendre l’exercice obsolète