

*Sujet:*

PRÊT SÛR : RÉGRESSION  
LOGISTIQUE AU SERVICE DE LA  
SÉLECTION DE PRÊTS BANCAIRES

Réalisé par : CHEMLAL Ismail et TOULBA Boubacar

Encadré par : Professeur BADAoui Fadoua



# Table des matières

1	Introduction . . . . .	ii
2	Étude descriptive des variables . . . . .	iii
2.1	Variables catégorielles . . . . .	iii
2.2	Variables numériques . . . . .	iv
2.3	Étude de la variable cible . . . . .	v
3	Préparation et nettoyage des données . . . . .	vi
3.1	Transformation de la variable <code>length</code> . . . . .	vii
3.2	Suppression des variables redondantes et inutiles . . . . .	viii
3.3	Traitement des variables catégorielles et numériques rares . . . . .	x
3.4	Transformation des variables indépendantes et traitement des valeurs manquantes . . . . .	xi
4	Modèle de Régression Logistique . . . . .	xiii
4.1	Division du Jeu de Données . . . . .	xiii
4.2	Construction du Modèle de Régression Logistique . . . . .	xiii
4.3	Évaluation du modèle de régression logistique . . . . .	xiv
4.4	Évaluation du modèle par la courbe ROC . . . . .	xv
4.5	Optimisation du seuil de classification . . . . .	xvi
5	Conclusion . . . . .	xvii

# 1 Introduction

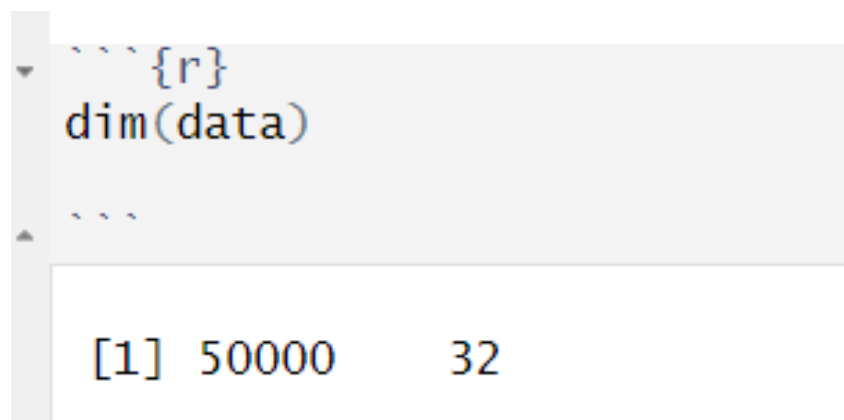
Ce rapport montre comment une banque peut utiliser un modèle informatique pour prédire si un prêt sera remboursé ou non. L'idée est simple : si la banque arrive à détecter à l'avance les prêts risqués (qui ne seront pas remboursés), elle peut les refuser et éviter de perdre de l'argent.

Pour cela, on utilise une méthode appelée régression logistique, qui permet de prévoir deux résultats possibles : soit le prêt est bon, soit il est mauvais. Grâce à ce modèle, on peut estimer la probabilité qu'un prêt soit bon. Par exemple, si la probabilité est supérieure à 0,5, on considère que le prêt est bon.

Avec ce système, on arrive à bien identifier 97

Le modèle utilisé est simple, facile à comprendre, et basé sur plusieurs caractéristiques du client (comme le revenu, le montant du prêt, etc.). Il peut déjà donner de très bons résultats, mais il peut aussi être amélioré dans le futur avec plus de données ou des techniques plus avancées.

En résumé, utiliser ce type de modèle permet à la banque de mieux choisir à qui prêter de l'argent, de réduire les pertes et d'augmenter fortement ses profits.



```
dim(data)
[1] 50000 32
```

FIGURE 1.1 – `dim(data)`

Voici le lien pour notre code R : [Notre code R sur GitHub](#)

## 2 Étude descriptive des variables

### 2.1 Variables catégorielles

Nom de variable	Description	Type
<b>term</b>	Durée du prêt (ex : 36 mois, 60 mois)	Catégorielle
<b>grade</b>	Note de crédit assignée au prêt	Catégorielle
<b>employment</b>	Profession de l'emprunteur	Catégorielle
<b>length</b>	Ancienneté professionnelle	Catégorielle
<b>home</b>	Statut de logement (RENT, MORTGAGE, etc.)	Catégorielle
<b>verified</b>	Statut de vérification du revenu	Catégorielle
<b>status</b>	Statut actuel du prêt (Fully Paid, Charged Off...)	Catégorielle
<b>reason</b>	Raison du prêt (debt consolidation, etc.)	Catégorielle
<b>state</b>	État de résidence de l'emprunteur	Catégorielle

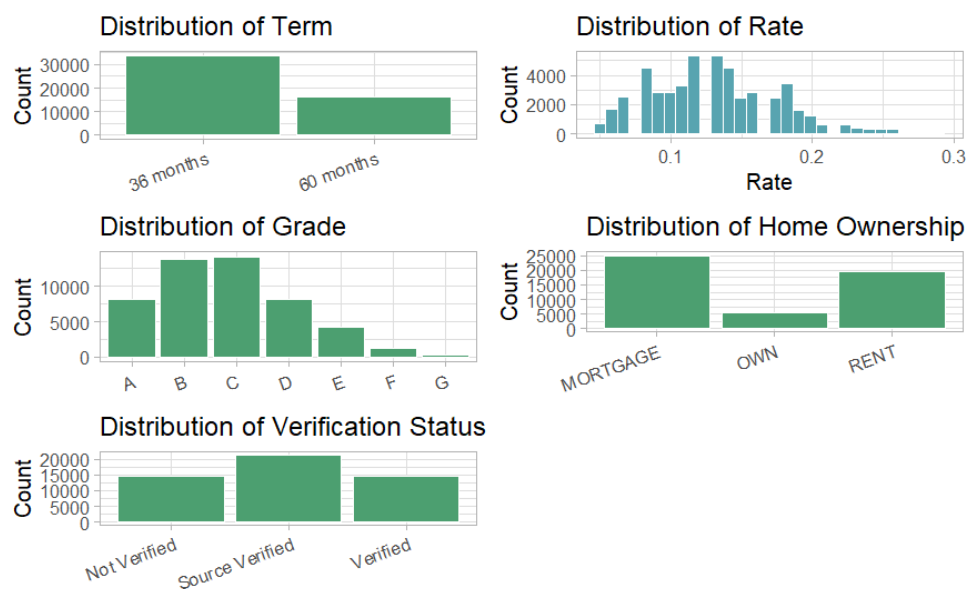


FIGURE 2.1 – Histogrammes pour variables catégoriques

## 2.2 Variables numériques

Nom de variable	Description	Type
<b>loanID</b>	Identifiant unique du prêt	Numérique
<b>amount</b>	Montant du prêt	Numérique
<b>rate</b>	Taux d'intérêt appliqué	Numérique
<b>payment</b>	Paieement mensuel	Numérique
<b>income</b>	Revenu annuel de l'emprunteur	Numérique
<b>debtIncRat</b>	Ratio dette/revenu	Numérique
<b>delinq2yr</b>	Délai depuis une délinquance sur 2 ans	Numérique
<b>inq6mth</b>	Nombre de demandes de crédit sur les 6 derniers mois	Numérique
<b>openAcc</b>	Nombre de comptes ouverts	Numérique
<b>pubRec</b>	Enregistrements publics négatifs	Numérique
<b>revolRatio</b>	Ratio d'utilisation du crédit renouvelable	Numérique
<b>totalAcc</b>	Nombre total de comptes de crédit	Numérique
<b>totalPaid</b>	Total remboursé par l'emprunteur	Numérique
<b>totalBal</b>	Solde total dû	Numérique
<b>totalRevLim</b>	Limite totale de crédit renouvelable	Numérique
<b>accOpen24</b>	Nombre de comptes ouverts sur 24 mois	Numérique
<b>avgBal</b>	Solde moyen par compte	Numérique
<b>bcOpen</b>	Nombre de comptes bancaires ouverts	Numérique
<b>bcRatio</b>	Ratio d'utilisation des cartes bancaires	Numérique
<b>totalLim</b>	Limite de crédit totale	Numérique
<b>totalRevBal</b>	Solde de crédit renouvelable total	Numérique
<b>totalBcLim</b>	Limite totale des cartes de crédit bancaire	Numérique
<b>totalIllim</b>	Limite de crédit à tempérament total	Numérique

## 2.3 Étude de la variable cible

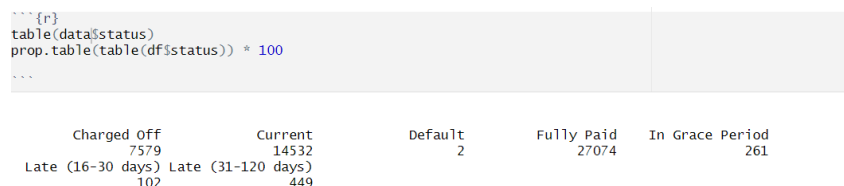


FIGURE 2.2 – Répartition des statuts de prêt

La variable `status` représente l'état actuel de chaque prêt dans le jeu de données. Elle prend différentes modalités, notamment :

- `Fully Paid` : le prêt a été entièrement remboursé par l'emprunteur.
- `Charged Off` : le prêt a été considéré comme irrécouvrable.
- `Current` : le prêt est en cours de remboursement.
- `In Grace Period` : le prêt est dans sa période de grâce.
- `Late (16-30 days)` : le remboursement est en retard entre 16 et 30 jours.
- `Late (31-120 days)` : le remboursement est en retard entre 31 et 120 jours.
- `Default` : le prêt est officiellement en défaut.

Pour simplifier l'analyse et permettre la mise en œuvre de modèles prédictifs supervisés, nous avons transformé cette variable en une variable binaire appelée `default`, selon les règles suivantes :

- `default = 1` si le statut est `Charged Off` (prêt en défaut),
- `default = 0` si le statut est `Fully Paid` (prêt remboursé).

Les autres observations, correspondant à des prêts encore actifs ou en retard, ont été exclues de cette première analyse.

Cette transformation permet de formuler un problème de classification binaire, où l'objectif est de prédire si un prêt donné sera remboursé ou non.

### 3 Préparation et nettoyage des données

Avant de procéder à l'analyse et à la modélisation, un prétraitement des données est nécessaire afin d'assurer la qualité et la pertinence des observations utilisées. Le code suivant a été utilisé pour filtrer et transformer les données du jeu `data` :

```
data = data %>% filter
(status == "Fully Paid" | status == "Charged Off" | status == "Default")
data = data %>% mutate
(response = case_when(status == "Fully Paid" ~ "Good", TRUE ~ "Bad"))
data$response <- as.factor
(data$response)
```

L'interprétation de ce code est la suivante :

- **Filtrage des observations** : la première ligne conserve uniquement les prêts ayant un statut final clairement identifiable : *"Fully Paid"* (remboursé), *"Charged Off"* (perte enregistrée) ou *"Default"* (défaut de paiement). Les autres statuts intermédiaires ou ambigus sont exclus pour ne garder que les cas utiles à une analyse binaire du risque de crédit.
- **Création de la variable cible** : une nouvelle variable `response` est introduite pour distinguer les "bons" emprunteurs (*"Good"*) – ceux qui ont remboursé intégralement – des "mauvais" (*"Bad"*) – ceux ayant fait défaut ou dont le prêt a été radié.
- **Conversion en facteur** : la variable `response` est convertie en facteur (variable catégorielle), ce qui est indispensable pour les modèles de classification supervisée en R.

Cette étape permet donc de transformer les données brutes en un jeu propre et directement exploitable pour des analyses de type prédictif, comme la modélisation du risque de défaut.



## 3.1 Transformation de la variable `length`

Dans le jeu de données, la variable `length` indique l'ancienneté professionnelle de l'emprunteur sous forme textuelle (par exemple : "`2 years`", "`< 1 year`", "`10+ years`"). Pour permettre une exploitation quantitative, cette variable a été transformée en valeurs numériques selon le code suivant :

```
data = data %>% mutate(length = case_when(  
  length == "< 1 year" ~ 0,  
  length == "1 year" ~ 1,  
  length == "2 years" ~ 2,  
  length == "3 years" ~ 3,  
  length == "4 years" ~ 4,  
  length == "5 years" ~ 5,  
  length == "6 years" ~ 6,  
  length == "7 years" ~ 7,  
  length == "8 years" ~ 8,  
  length == "9 years" ~ 9,  
  length == "10+ years" ~ 10,  
  TRUE ~ NA_real_  
))
```

L'objectif est de convertir les chaînes de caractères en une échelle numérique ordinale représentant le nombre d'années d'expérience. Cette transformation facilite l'analyse statistique et l'utilisation de modèles prédictifs. La valeur "`< 1 year`" est codée comme 0, tandis que "`10+ years`" devient 10. Toute modalité inconnue ou absente est convertie en valeur manquante (NA).

Ce recodage rend la variable exploitable pour des analyses numériques (

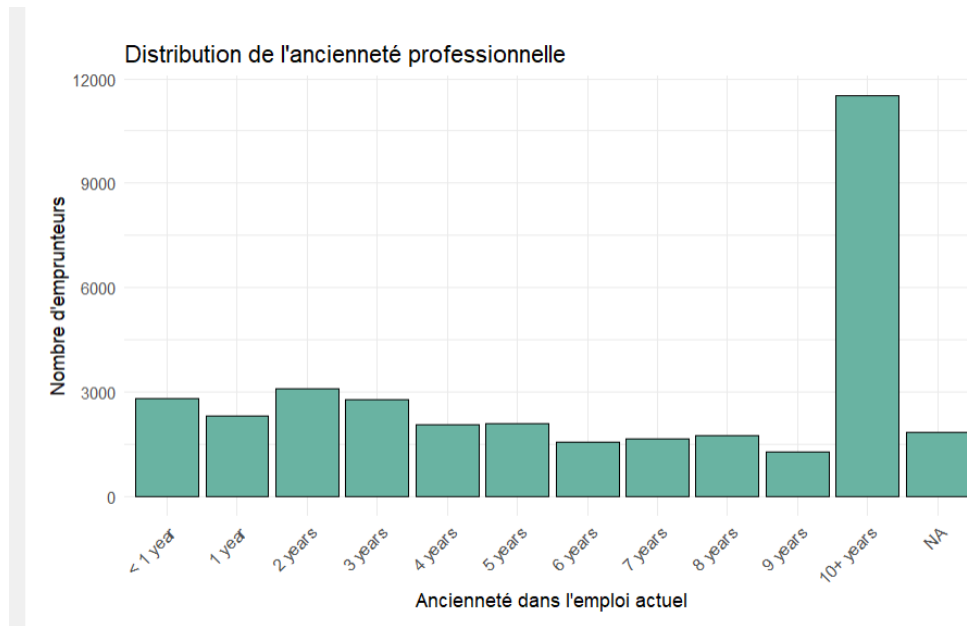


FIGURE 3.1 – Histogramme de l'ancienneté professionnelle des emprunteurs (en années).

## 3.2 Suppression des variables redondantes et inutiles

Lors de l'analyse exploratoire des données, nous avons observé que certaines variables dans le jeu de données présentaient des relations fortes ou des problèmes de cardinalité élevée, ce qui compliquait leur utilisation dans le cadre de la modélisation. En particulier, les variables suivantes ont été identifiées comme étant redondantes ou difficiles à modéliser.

### 2..1 Corrélation entre le montant du prêt et les paiements mensuels

Une forte corrélation a été observée entre la variable `loan amount` (montant du prêt) et la variable `monthly payment` (paiement mensuel), ce qui les rend redondantes. Un nuage de points a été tracé pour visualiser cette relation, comme montré dans la figure 3.2. En raison de cette forte corrélation linéaire, il a été décidé de supprimer la variable `payment` du jeu de données pour éviter la multicolinéarité et simplifier le modèle.

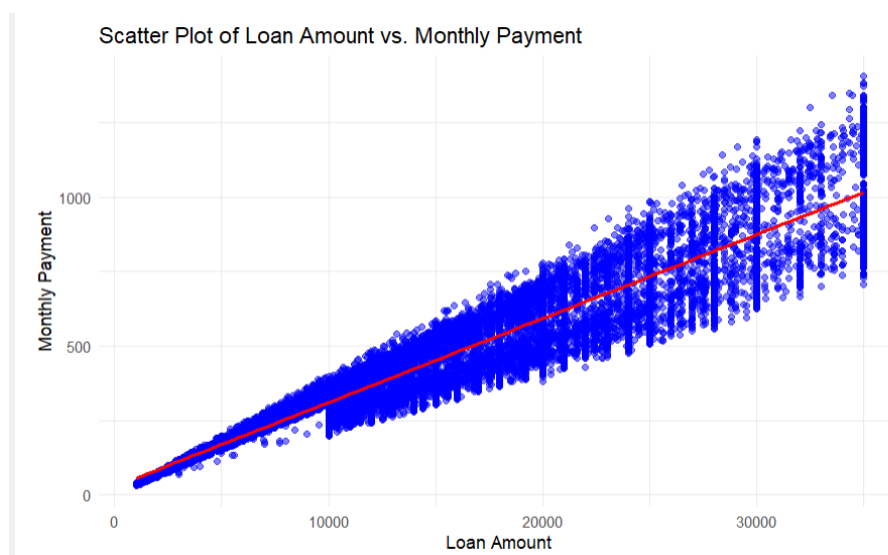


FIGURE 3.2 – Nuage de points entre le montant du prêt et les paiements mensuels.

```
{r}
# 2. Calcul de la corrélation linéaire entre 'amount' et 'payment'
correlation <- cor(data$amount, data$payment)
cat("Correlation between loan amount and monthly payment:", correlation, "\n")

Correlation between loan amount and monthly payment: 0.9524533
```

FIGURE 3.3 – corolaltion.

## 2..2 Problème de cardinalité avec la variable employment

La variable `employment` contient un grand nombre de catégories uniques, dont une majorité ne dispose que d'une seule observation. Par exemple, 12,572 catégories de la variable `employment` n'ont qu'une seule observation, et seulement 1,238 catégories ont deux observations. Cette diversité excessive rend difficile l'utilisation de cette variable pour la modélisation, car elle introduit une variabilité excessive et peu d'information utile. En raison de cette problématique de cardinalité, la variable `employment` a également été supprimée du jeu de données.

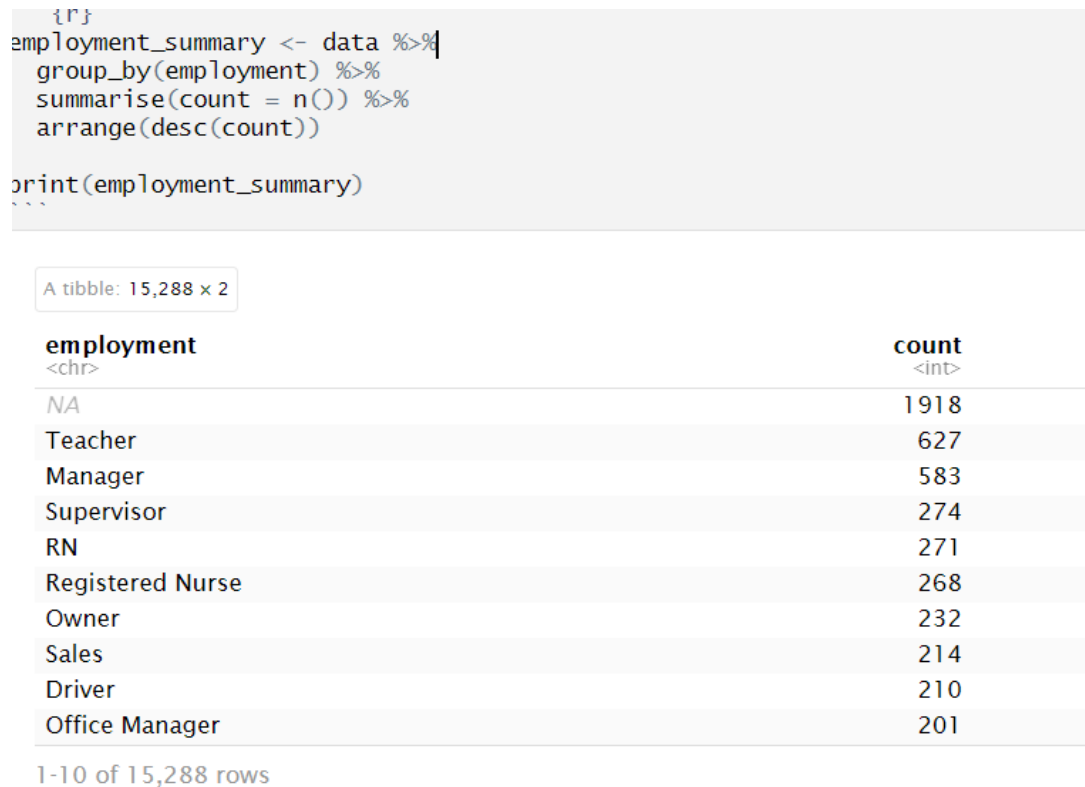


FIGURE 3.4 – summary.

### 2..3 Conclusion

La suppression de ces deux variables (`payment` et `employment`) permet de simplifier le modèle en réduisant la multicolinéarité et en éliminant des variables peu informatives. Le jeu de données est ainsi plus facile à gérer pour les étapes suivantes de modélisation.

## 3.3 Traitement des variables catégorielles et numériques rares

Afin d'améliorer la qualité du modèle prédictif et de réduire la complexité des données, certaines transformations ont été appliquées aux variables du jeu de données `loans` :

- **Variable `accOpen24`** : Cette variable représente le nombre de comptes ouverts au cours des 24 derniers mois. Très peu d'individus ont ouvert plus de 20 comptes durant cette période. Par conséquent, toutes les valeurs strictement supérieures à 19 ont été regroupées sous la valeur 20, de manière à réduire

l'impact des cas extrêmes.

- **Variable state** : Certains États sont représentés par un nombre très faible de demandes de prêt. Pour éviter le surapprentissage lié à des catégories peu fréquentes, les regroupements suivants ont été effectués :
  - États avec moins de 150 demandes regroupés sous la catégorie 0T (648 cas).
  - États avec un nombre de demandes compris entre 150 et 174 regroupés sous la catégorie 02 (308 cas).
  - États avec un nombre de demandes compris entre 175 et 199 regroupés sous la catégorie 03 (518 cas).

Ces regroupements permettent de limiter la cardinalité de la variable `state` tout en conservant l'information pertinente.

```

{r}
data2 = data2 %>%
  mutate(accOpen24 = case_when(accOpen24 > 19 ~ 20, TRUE ~ as.numeric(accOpen24)))

other1 = data2 %>% group_by(state) %>% summarise(count = n()) %>%
  filter(count < 150) %>% pull(1)
other2 = data2 %>% group_by(state) %>% summarise(count = n()) %>%
  filter(count >= 150 & count < 175) %>% pull(1)
other3 = data2 %>% group_by(state) %>% summarise(count = n()) %>%
  filter(count >= 175 & count < 200) %>% pull(1)
data2 = data2 %>% mutate(state = case_when(state %in% other1~"0T",
                                           state %in% other2~"02",
                                           state %in% other3~"03",
                                           TRUE~state))

```

FIGURE 3.5 – summary.

## 3.4 Transformation des variables indépendantes et traitement des valeurs manquantes

Certaines variables quantitatives présentent une forte asymétrie ou des valeurs extrêmes susceptibles de nuire à la performance des modèles prédictifs. Pour y remédier, une transformation logarithmique a été appliquée à plusieurs variables. La transformation  $\log(x + 1)$  est utilisée lorsque des valeurs nulles sont présentes, afin d'éviter les valeurs infinies :

- `income`, `totalRevLim`, `totalBal`, `avgBal`, `bcOpen`, `pubRec`, `inq6mth`, `delinq2yr`

Cette transformation permet de réduire la variance et de rendre la distribution de ces variables plus proche de la normale, ce qui est favorable pour de nombreux algorithmes.

Ensuite, les valeurs manquantes restantes ont été identifiées à l'aide de la fonction `sapply`. Les variables `revolRatio`, `bcOpen` et `bcRatio` présentaient respectivement 15, 360 et 384 valeurs manquantes. Toutes les observations contenant des valeurs manquantes ont été supprimées à l'aide de la fonction `drop_na()`, afin de garantir l'intégrité des analyses futures.

```
data2 = data2 %>% drop_na()
```

```
## {r}
data2 = data2 %>% mutate(income = log(income), totalBal = log(totalBal+1), totalRevLim = log(totalRevLim),
  avgBal = log(avgBal+1), bcOpen = log(bcOpen+1), pubRec = log(pubRec+1),
  inq6mth = log(inq6mth+1), delinq2yr = log(delinq2yr+1))

## {r}
sapply(data2, function(x) sum(is.na(x)))
```

loanID	amount	term	rate	payment	grade	employment	length	home	income	verified	status
reason	state	debtIncRat	delinq2yr	inq6mth	openAcc	pubRec	revolRatio	totalAcc	totalPaid	totalBal	totalRevLim
accOpen24	avgBal	bcOpen	bcRatio	totalLim	totalRevBal	totalBcLim	totalIllim	response	length_label		
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	15	0	0	0	0
0	0	360	384	0	0	0	0	0	1823		

```
## {r}
library(tidyrr)
data2 = data2 %>% drop_na()
```

FIGURE 3.6 – rcode

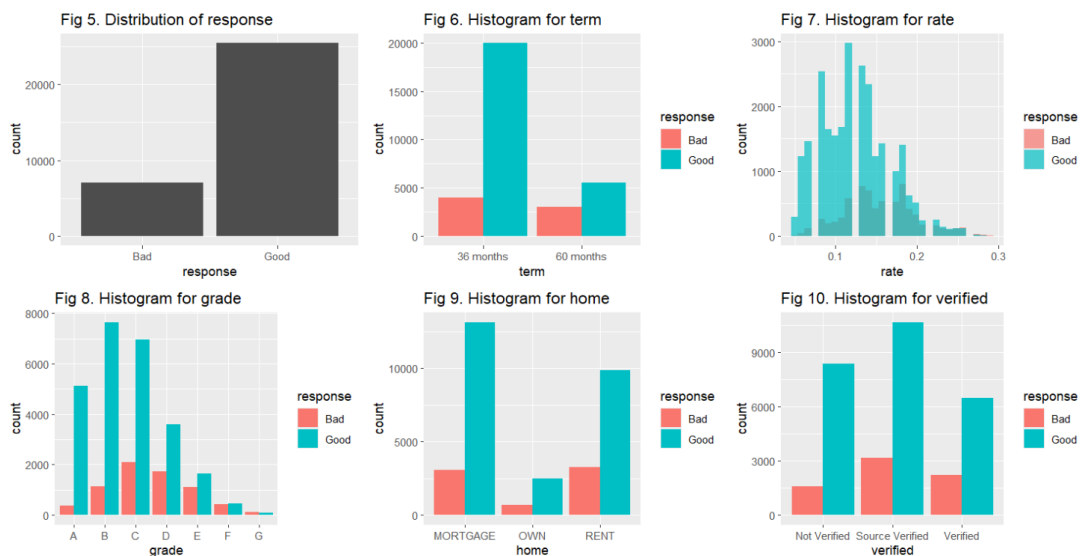


FIGURE 3.7 – histogrammes

## 4 Modèle de Régression Logistique

Dans cette section, nous allons construire un modèle de régression logistique pour prédire l'issue d'un prêt en tant que "Bon" ou "Mauvais". Nous utiliserons le jeu de données `loans`, que nous diviserons d'abord aléatoirement en un ensemble d'entraînement (80% des données) et un ensemble de test (20% des données).

### 4.1 Division du Jeu de Données

Nous divisons le jeu de données comme suit :

- `smp_size` représente la taille de l'ensemble d'entraînement, calculée comme 80% du nombre total de lignes dans le jeu de données.
- `train_ind` est un échantillon aléatoire des indices correspondant à l'ensemble d'entraînement.
- Le jeu de données est divisé en `train` (pour l'entraînement) et `test` (pour les tests).

Le code R permettant de diviser les données est le suivant :

```
smp_size = floor(0.8 * nrow(loans)) ## 80% de la taille de l'échantillon
set.seed(10)
train_ind = sample(seq_len(nrow(loans)), size = smp_size)
train = data2[train_ind, ]
test = data2[-train_ind, ]
train = train %>% select(-c(totalPaid,status,loanID))
```

### 4.2 Construction du Modèle de Régression Logistique

Nous construisons ensuite le modèle de régression logistique à l'aide de la fonction `glm`. La variable dépendante est `response`, qui indique si un prêt est `Bon` ou `Mauvais`. Le modèle est construit en utilisant tous les autres prédicteurs du jeu de données.

Le code R pour construire le modèle est :

```
##The Logistic Model

```{r}
smp_size = floor(0.8 * nrow(data2))
set.seed(10)
train_ind = sample(seq_len(nrow(data2)), size = smp_size)
train = data2[train_ind, ]
test = data2[-train_ind, ]
train = train %>% select(-c(totalPaid,status, loanID))
```

```{r}
full <- glm(response~.,data=train, family = "binomial")
```
```

FIGURE 4.1 – model

### 4.3 Évaluation du modèle de régression logistique

Après l'entraînement du modèle de régression logistique sur l'ensemble d'apprentissage, nous avons utilisé les données de test pour évaluer ses performances. Les probabilités prédites pour chaque observation ont été obtenues à l'aide de la fonction `predict` :

```
probs = predict(full, newdata = test, type = "response")
```

Ensuite, ces probabilités ont été comparées à un seuil de classification fixé à 0,5 pour obtenir les prédictions finales :

```
predictions = predictions %>%
  mutate(predicted = case_when((prob > threshold) ~ 'Good', TRUE ~ 'Bad'))
```

Nous avons ensuite généré une matrice de confusion pour comparer les valeurs réelles (`actual`) et les valeurs prédites (`predicted`) :

| Réel / Prédit | Bad | Good | Total |
|---------------|-----|------|-------|
| Bad           | 172 | 1281 | 1453  |
| Good          | 131 | 4911 | 5042  |
| Total         | 303 | 6192 | 6495  |

TABLE 4.1 – Matrice de confusion obtenue sur l'ensemble de test

À partir de cette matrice, nous calculons les métriques suivantes :



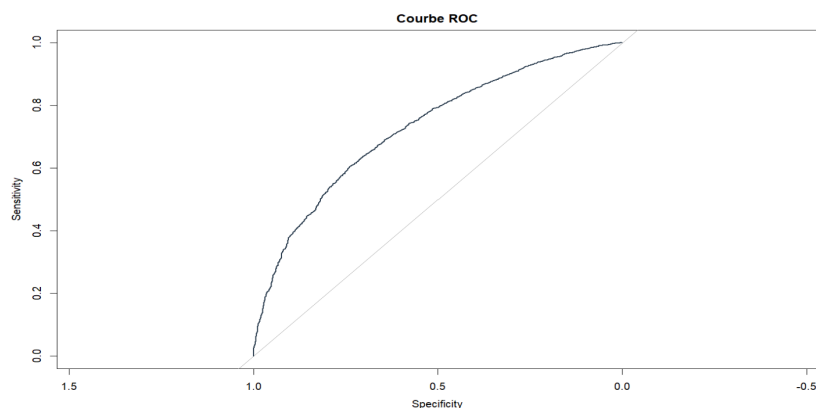
- **Accuracy** :  $\frac{172+4911}{6495} \approx 78,25 \%$
- **Precision** :  $\frac{4911}{4911+1281} \approx 79,32 \%$
- **Recall** :  $\frac{4911}{4911+131} \approx 97,39 \%$
- **F1-score** :  $2 \cdot \frac{0,7932 \cdot 0,9739}{0,7932+0,9739} \approx 87,56 \%$

Ces résultats montrent que le modèle est particulièrement bon pour identifier les bons emprunteurs (haut rappel), mais qu'il a tendance à mal classer certains mauvais emprunteurs (faux positifs élevés), ce qui peut poser un risque en contexte de prêt.

## 4.4 Évaluation du modèle par la courbe ROC

Pour évaluer la performance du modèle de régression logistique, nous avons utilisé la courbe ROC (Receiver Operating Characteristic), qui représente le compromis entre la sensibilité (taux de vrais positifs) et la spécificité (1 - taux de faux positifs).

La courbe ROC a été générée à partir des probabilités prédites par le modèle appliqué aux données de test. L'aire sous la courbe (AUC) obtenue est de 0,7286, ce qui indique que le modèle possède une bonne capacité de discrimination entre les prêts considérés comme *Good* et ceux considérés comme *Bad*.



**FIGURE 4.2** – Courbe ROC du modèle de régression logistique. Les axes débutent à zéro et l'AUC est indiquée sur le graphique.

Une AUC de 0,7286 signifie que dans environ 72,86% des cas, le modèle attribue une probabilité plus élevée à un prêt *Good* qu'à un prêt *Bad*. Ce résultat montre une performance correcte du modèle, bien qu'il soit encore possible de l'améliorer.

Nous avons veillé à ce que les axes de la courbe commencent à zéro afin de respecter

les échelles naturelles de la sensibilité et de la spécificité. Une ligne diagonale (en gris) représentant un modèle aléatoire a également été tracée pour comparaison.

## 4.5 Optimisation du seuil de classification

Dans cette section, nous faisons varier le seuil de classification du modèle logistique pour identifier la valeur qui maximise la précision (accuracy). Le seuil par défaut est 0.5, mais ce choix peut ne pas être optimal. En faisant varier le seuil entre la plus petite et la plus grande probabilité prédite, nous analysons son effet sur la précision, la sensibilité, la spécificité et le profit.

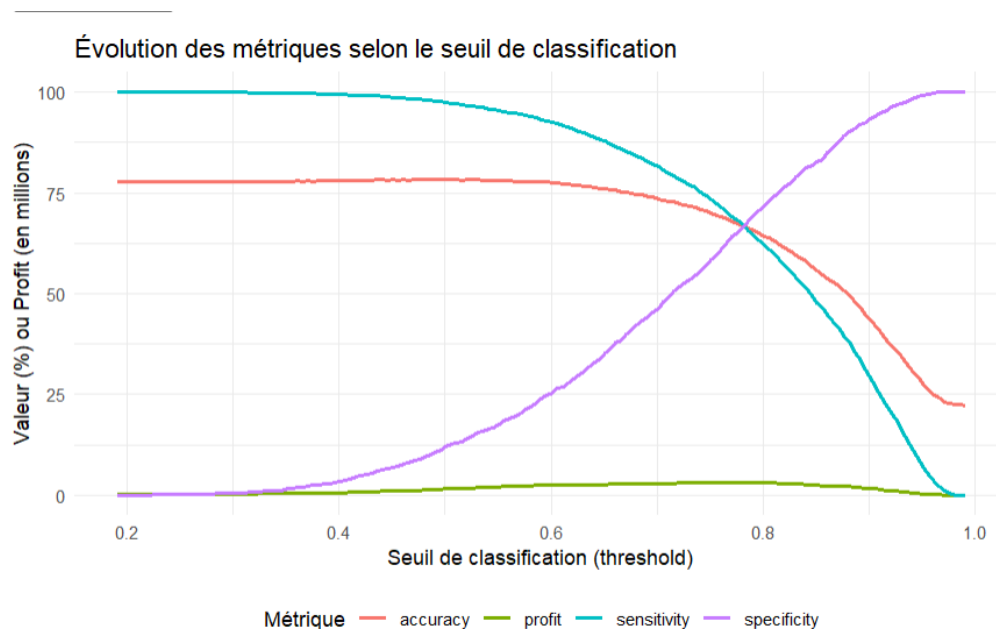


FIGURE 4.3 – Optimisation du seuil de classification

**Interprétation :** Le graphique montre que la précision et les autres métriques dépendent fortement du seuil choisi. En identifiant le seuil qui maximise la précision, on améliore la performance globale du modèle. Toutefois, ce seuil peut varier selon que l'on préfère minimiser les faux positifs (spécificité) ou maximiser les vrais positifs (sensibilité). Un compromis optimal est souvent nécessaire selon les objectifs de l'analyse (rentabilité, risque, etc.).

## 5 Conclusion

Ce projet démontre l'utilité concrète d'un modèle prédictif pour aider une banque à prendre de meilleures décisions d'octroi de prêts. L'analyse a permis d'identifier un seuil de décision optimal à **0.6504**, pour lequel le profit est maximisé.

### 0..1 Résultats clés :

- **Précision globale** : 76.29 %
- **Sensibilité (bons prêts bien prédits)** : 87.25 %
- **Spécificité (mauvais prêts bien rejetés)** : 37.12 %
- **Profit augmenté de 270.7 %** par rapport à la situation actuelle

Grâce à ce modèle, la banque pourrait réaliser un profit **2.7 fois supérieur** à celui obtenu actuellement sans modèle prédictif.

Cependant, il reste une marge d'amélioration. Un modèle parfait (rejetant tous les mauvais prêts) aurait permis un profit **9.3 fois supérieur**. Ainsi, le modèle proposé atteint environ **29 % du profit maximum théorique**.