



# Détection de Fraude à la Création de Compte Bancaire

Yannis RACLOT - Arthur VORANGET - Vicnesh VENEDITTAN

Projet de Statistiques en Grande Dimension

Master 2 Mathématiques et Applications ( IMSD )

Université de Lorraine

Décembre 2023

# 1 Présentation du Projet

## 1.1 Contexte

La sécurité et la prévoyance revêtent une importance cruciale dans le secteur bancaire et de l'assurance. Afin de minimiser les risques, il est impératif pour les institutions financières de mettre en place des méthodes permettant de détecter d'éventuelles fraudes. Avec le développement de l'intelligence les tentatives de fraudes seront de plus en plus élaboré il est donc de plus en plus nécessaire de savoir encore mieux les détecter et ce demanière précise.

Dans ce contexte, notre mission consiste à développer des modèles capables de prédire, à partir de jeux de données portant sur des demandes de création de compte bancaire, si ces demandes sont susceptibles d'être frauduleuses ou si elles sont normales. L'objectif principal de notre démarche est d'exploiter nos connaissances acquises pour comprendre les données qui nous sont fournies, les nettoyer de manière approfondie, puis entreprendre la création du meilleur modèle possible pour détecter efficacement les potentielles fraudes liées aux demandes de compte bancaire.
Nous utiliserons un jeu de donnée sous licence libre trouvé sur la plateforme communautaire Kaggle.

Nous nous poserons les questions suivantes tout au long du projet :

## 1.2 Problématique

Quelles sont les variables les plus influentes dans la détection des fraudes ? Quelles méthodes s'avèrent les plus efficaces pour la création de notre modèle ? Enfin, dans quelle mesure notre modèle final est-il réellement efficace ?

# Table des matières

**1 Présentation du Projet** **1**
1.1 Contexte . . . . . 1
1.2 Problématique . . . . . 1

**2 Exploration & Premières Analyses de Données** **3**
2.1 Jeu de Données . . . . . 3
2.2 Visualistion . . . . . 4
2.3 Variables manquantes . . . . . 6

**3 Créations de Modèles** **6**
3.1 Modèle Linéaire . . . . . 7
3.2 Modèle GLM . . . . . 9
3.3 Modèle SVM . . . . . 10
3.4 Modèle Fôret Aléatoires . . . . . 11

**4 Conclusion** **13**
4.1 Comparaison des Résultats . . . . . 13
4.2 Ouverture . . . . . 14

**5 Annexes** **15**

# 2 Exploration & Premières Analyses de Données

## 2.1 Jeu de Données

Nous nous sommes munis d'un jeu de donnée "Base" sous format .csv qui représente les caractéristiques des souscripteurs ayant demandé la création d'un compte bancaire, le jeu de données possèdent 1000000 observations (souscripteur) et chacune possédant 32 variables différentes. Les variables qui le composent sont telles que :

1. **fraud_bool** (binaire) : Est-ce que la demande est frauduleuse ou non
2. **foreign_request** (binaire) : Est-ce que le pays du souscripteur est différent du pays d'origine de la banque
3. **income** (numérique) : Revenue Annuel du souscripteur (forme décimale) marge comprise entre [0.1, 0.9]
4. **customer_age** (numérique) : Âge du souscripteur arrondi à la dizaine [10,90] ans
5. **device_os** (catégorielle) : Le système d'exploitation de l'appareil qui a effectué la demande. Les valeurs possibles sont : Windows, macOS, Linux, X11, ou autre.
6. **velocity_24h** (numérique) : Nombre moyen de demandes par heure au cours des dernières 24 heures. Varie entre [1297, 9586].

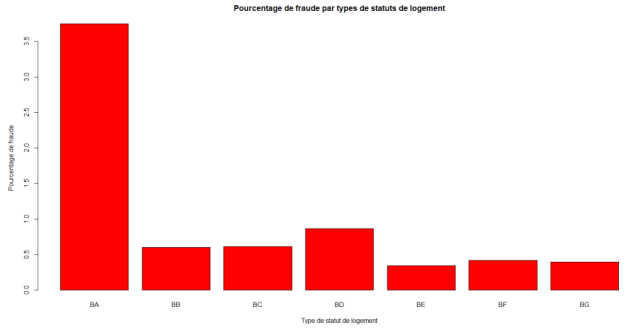
Au total il y a 20 variables quantitatives, 7 variables binaires, 5 variables catégorielles, l'ensemble des descriptions des 32 variables se trouvent dans l'Annexe. (Certaines des variables catégorielles possèdent des modalités au nom anonymisé, c'est le cas des variables **payment_type**, **employment_status** et **housing_status**. )

Exceptés les variables **velocity_6h** & **credit_risk_score** toutes les valeurs négatives du jeu de données sont en réalité des valeurs manquantes. Nous les avons donc directement transformé en *NAN*.
Seulement 6 variables possèdent au moins une variables manquantes : **prev_address_months_count ; current_address_months_count ; intended_balcon_amount ; bank_months_count ; session_length_in_minutes ; device_distinct_emails_8w**
Le pourcentage des valeurs manquantes de ces variables vas de 0.0359% à 74.2523%.
Nous avons supprimé la variable device_fraud_count, étant donné qu'elle ne possède qu'une seule modalité, elle n'a aucun intérêt.

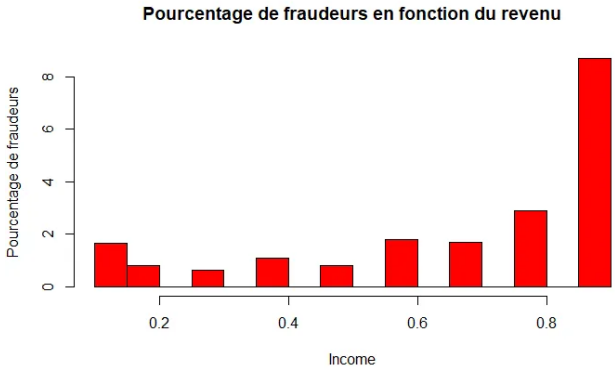
Nous voulons prédire si les souscripteurs sont des fraudeurs ou non, alors la variable binaire que nous voulons prédire est donc la variable : **fraud_bool**, il y'a au total 1.1% de fraudeurs ( **fraud_bool**=1) dans tout le jeu de données.

## 2.2 Visualistion

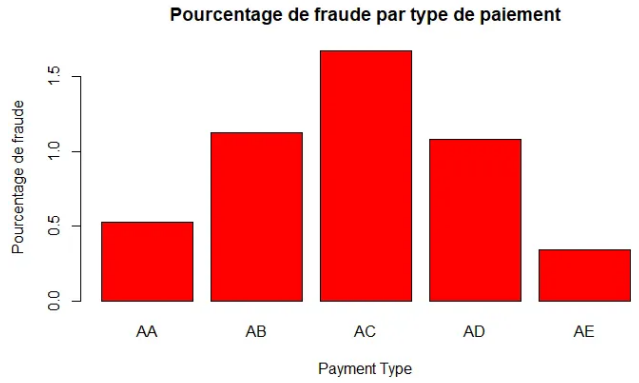
Commençons d'abord par des analyses simples de l'évolution de notre variable cible par rapports à certaines autres variables, nous avons sélectionné 4 variables qui nous semblaient pertinentes (**housing_status** ; **payment_type** ; **income** ; **name_email_similarity**), car nous les imaginions plutôt corrélés avec notre variable cible :



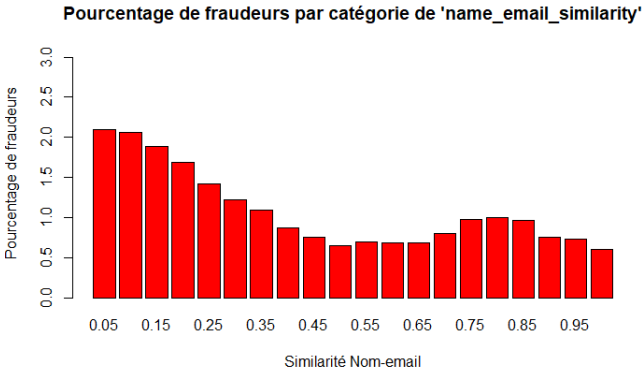

BA est largement le type de logement qui en proportion possède le plus de fraudeurs, les autres types de logement sont indiqués en proportion assez similairement.




Les fraudeurs ont tendance à fournir lors de leur inscription un revenu élevé.




C'est le type de paiement AC en proportion qui possèdent plus de fraudeurs, AE le moins et AB ET AD respecte à peu près la moyenne.




Parmi les souscriptions considérés comme frauduleuse on remarque que les emails sont plus éloignés de leurs identités.

Nous avons aussi fait le tableau de corrélation de toutes les variables ainsi qu'un focus sur la première ligne de ce tableau concernant **fraud_bool1** avec toutes les autres variables :

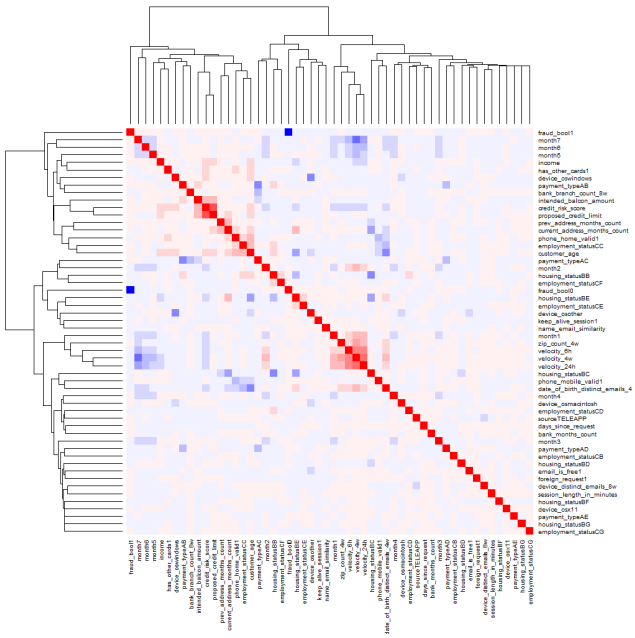


FIGURE 2 – Matrice de corrélation avec toutes les variables

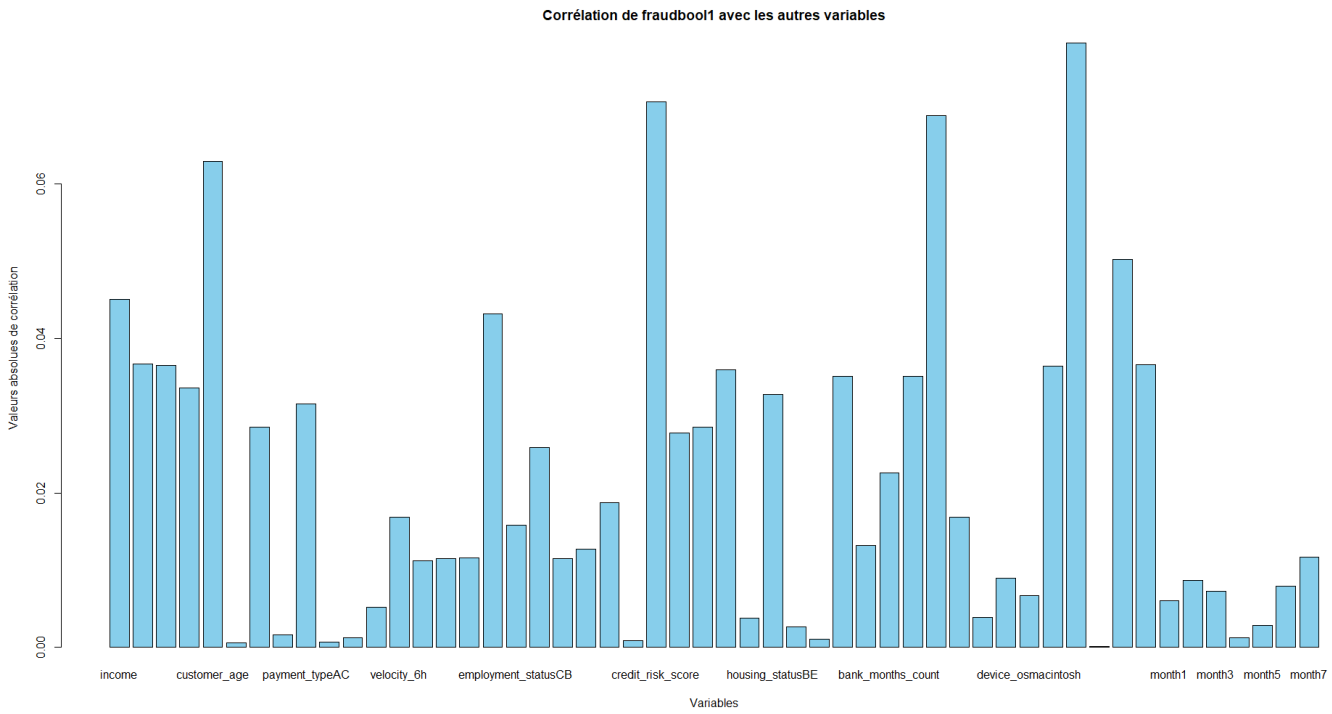



FIGURE 3 – Corrélation la variable **fraud_bool1** ayant pour valeur "1"

Nous obserons des corrélations jusqu'à 0.10 avec **fraud_bool1** selon les variables. Et certaines qui sont totalement décorrélés.

### 2.3 Variables manquantes

Certains modèles ne sont pas très compatibles avec dataset comprenant des données manquantes, afin de ne pas avoir à supprimer les observations possédant une donnée manquante, et afin de ne pas supprimer les variables possédant beaucoup de données manquantes, nous avons décidé de remplacer ces données non pas à l'aide de la moyenne de chaque variable, ce qui pourrait créer des biais dans l'estimation de notre variable cible, mais plutot par la méthode Predictive Mean Matching (PMM). L'intérêt de cette méthode est de prendre en compte la distribution des données en imputant des valeurs qui reflètent mieux la variabilité des données observées, afin d'imputer des valeurs pour remplacer celles-manquantes, cette méthode se base sur les relations entre les variables pour prédire ou estimer les valeurs manquantes d'une variable spécifique, elle utilise les valeurs observées des autres variables pour trouver des observations similaires à celle dont la donnée est manquante.

Afin de vérifier si remplacer les données manquantes des deux variables possédants plus de 70% de données manquantes est bien utile pour la prédiction de notre variable cible, nous avons comparé les résultats obtenus sur nos modèles avec les même modèles utilisant le dataset sans les deux variables en question, et les modèles étaient moins bons sans les variables, nous avons donc gardé cette méthode pour la suite.

## 3 Créations de Modèles

Afin de pouvoir limiter la durée d'exécution des différents modèles, il était impératif de réduire le nombre d'observations que l'on va utiliser pour créer les modèles, sur les exemples suivant nous avons séléctionnés aléatoirement 50 000 observations.

Suite à cela nous avons séparés les données aléatoirement en un ensemble train et test, possédants respectivement 70% et 30% des observations.

La variable cible **fraud_bool** possède un nombre de valeur très déséquilibré dans ses modalités, nous avons donc suréchantillonné la classe minoritaire de l'ensemble de train de sorte à ajouter la moitié de la différence du nombre d'observation de chaque modalité, à la modalité minoritaire, cela nous permet d'ajuster le poids de chaque modalité de la variable cible.

## 3.1 Modèle Linéaire

Nous avons tout d'abord voulu commencer par un modèle simple : la régression linéaire. Nous savions que le modèle final ne serait pas parfait, car dans cette étude nous cherchons à prédire une variable binaire, et la régression linéaire n'est pas adaptée à cela.

Nous avons d'abord tenté avec toutes les variables, mais pour effectuer la régression linéaire il fallait séparer les variables catégorielles en plusieurs variables binaires. Avec ceci, nous obtenions le résultat suivant :

```
> summary(modellin)

Call:
lm(formula = fraud_bool ~ ., data = data_train3)

Residuals:
     Min       1Q   Median       3Q      Max
-0.10822 -0.02129 -0.00729  0.00371  1.01035

Coefficients: (1 not defined because of singularities)
                                 Estimate Std. Error t value Pr(>|t|)
(Intercept)                    -1.358e-02  3.334e-02  -0.407 0.683767
income                          6.367e-03  1.990e-03   3.200 0.001377 **
name_email_similarity          -1.432e-02  1.912e-03  -7.490 7.03e-14 ***
prev_address_months_count       3.503e-05  7.515e-06   4.661 3.16e-06 ***
current_address_months_count    1.294e-05  7.120e-06   1.817 0.069256 .
customer_age                    3.705e-04  5.678e-05   6.526 6.86e-11 ***
days_since_request              1.016e-04  9.553e-05   1.064 0.287513
intended_balcon_amount          2.080e-05  2.190e-05   0.950 0.342198
zip_count_4w                    2.007e-06  5.758e-07   3.486 0.000491 ***
velocity_6h                     3.331e-08  2.104e-07   0.158 0.874189
velocity_24h                   -7.280e-08  4.728e-07  -0.154 0.877626
velocity_4w                    -1.414e-06  1.205e-06  -1.174 0.240547
bank_branch_count_8w           -2.308e-06  1.211e-06  -1.906 0.056721 .
date_of_birth_distinct_emails_4w -1.196e-04  1.272e-04  -0.940 0.347080
credit_risk_score               3.574e-05  1.077e-05   3.320 0.000902 ***
bank_months_count               9.617e-05  4.804e-05   2.002 0.045304 *
proposed_credit_limit           1.901e-06  1.484e-06   1.281 0.200171
session_length_in_minutes       1.319e-04  7.000e-05   1.884 0.059546 .
device_distinct_emails_8w       2.432e-02  3.108e-03   7.823 5.29e-15 ***
payment_typeAA                  2.995e-03  3.220e-02   0.093 0.925875
payment_typeAB                  3.184e-03  3.219e-02   0.099 0.921213
payment_typeAC                  1.066e-02  3.220e-02   0.331 0.740621
payment_typeAD                  4.387e-03  3.222e-02   0.136 0.891699
payment_typeAE                         NA         NA      NA       NA
employment_statusCB            -4.727e-03  1.612e-03  -2.932 0.003374 **
employment_statusCC             6.840e-03  3.036e-03   2.253 0.024277 *
employment_statusCD            -2.036e-03  3.419e-03  -0.596 0.551446
employment_statusCE            -3.754e-03  3.777e-03  -0.994 0.320190
employment_statusCF            -2.993e-03  2.771e-03  -1.080 0.280124
employment_statusCG            -1.430e-02  2.823e-02  -0.506 0.612509
email_is_free1                  7.216e-03  1.116e-03   6.464 1.03e-10 ***
housing_statusBB               -2.492e-02  1.844e-03 -13.515  < 2e-16 ***
housing_statusBC               -2.381e-02  1.784e-03 -13.349  < 2e-16 ***
housing_statusBD               -2.478e-02  3.661e-03  -6.768 1.33e-11 ***
housing_statusBE               -2.394e-02  2.110e-03 -11.346  < 2e-16 ***
housing_statusBF               -2.819e-02  1.301e-02  -2.167 0.030251 *
housing_statusBG               -2.275e-02  3.849e-02  -0.591 0.554529
phone_home_valid1              -7.372e-03  1.194e-03  -6.174 6.73e-10 ***
phone_mobile_valid1             2.711e-03  1.843e-03   1.471 0.141407
has_other_cards1               -9.191e-03  1.376e-03  -6.678 2.47e-11 ***
foreign_request1                5.304e-03  3.476e-03   1.526 0.127082
source_TELEAPP                  3.387e-02  6.633e-03   5.105 3.32e-07 ***
device_osmacintosh              8.833e-03  2.580e-03   3.424 0.000619 ***
device_osother                  2.011e-03  1.370e-03   1.468 0.142013
device_oswindows                1.541e-02  1.453e-03  10.610  < 2e-16 ***
device_osx11                    7.608e-03  6.366e-03   1.195 0.232105
keep_alive_session1            -7.965e-03  1.134e-03  -7.021 2.25e-12 ***
month1                         -4.043e-03  2.400e-03  -1.684 0.092159 .
month2                         -4.172e-03  2.380e-03  -1.753 0.079684 .
month3                         -8.536e-03  2.605e-03  -3.276 0.001053 **
month4                         -4.143e-03  2.872e-03  -1.443 0.149152
month5                         -1.141e-03  3.203e-03  -0.356 0.721607
month6                         -3.877e-03  3.406e-03  -1.138 0.254948
month7                          2.480e-04  4.130e-03   0.060 0.952114
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1017 on 34947 degrees of freedom
Multiple R-squared:  0.03642,   Adjusted R-squared:  0.03498
F-statistic:  25.4 on 52 and 34947 DF,  p-value: < 2.2e-16
```

FIGURE 4 – Modèle Linéaire avec toutes les variables

Il y a plusieurs variables non-significatives dans ce modèle, le $R^2 = 0.034$ ce qui est ridicule, notre modèle n'explique que 3.4% de notre jeu de données. Si nous tentons de prédire les fraudes à l'aide de ce modèle, nous obtenons :

```
> print(conf_matrixlin)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 10922    29
         1  3905   144

               Accuracy : 0.7377
                 95% CI : (0.7306, 0.7448)
    No Information Rate : 0.9885
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0471

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.73663
            Specificity : 0.83237
         Pos Pred Value : 0.99735
         Neg Pred Value : 0.03556
             Prevalence : 0.98847
         Detection Rate : 0.72813
   Detection Prevalence : 0.73007
      Balanced Accuracy : 0.78450

       'Positive' Class : 0
```

FIGURE 5 – Prédiction avec modèle Linéaire avec toutes les variables

Ce qui est une précision correcte, mais nous verrons plus tard que la précision n'est pas une bonne métrique pour estimer de la qualité de notre modèle.

Afin d'améliorer notre modèle, nous avons aussi pratiqué la méthode "backward" afin de sortir les variables qui plombaient le modèle et qui n'étaient pas significatives. Ainsi, nous obtenons :

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.10993 -0.02124 -0.00725  0.00359  1.01001

Coefficients:
                              Estimate Std. Error t value Pr(>|t|)
(Intercept)                 -1.048e-02  6.026e-03  -1.740 0.081855 .
income                       6.932e-03  1.971e-03   3.517 0.000437 ***
name_email_similarity       -1.428e-02  1.908e-03  -7.485 7.35e-14 ***
prev_address_months_count    3.510e-05  7.500e-06   4.680 2.87e-06 ***
current_address_months_count 1.330e-05  7.058e-06   1.885 0.059494 .
customer_age                 3.802e-04  5.307e-05   7.164 8.01e-13 ***
zip_count_4w                 1.932e-06  5.712e-07   3.382 0.000720 ***
velocity_4w                 -1.469e-06  6.944e-07  -2.116 0.034388 *
bank_branch_count_8w        -2.348e-06  1.207e-06  -1.945 0.051768 .
credit_risk_score            3.771e-05  1.063e-05   3.546 0.000391 ***
bank_months_count            9.708e-05  4.781e-05   2.031 0.042295 *
proposed_credit_limit        2.079e-06  1.467e-06   1.417 0.156607
session_length_in_minutes    1.332e-04  6.968e-05   1.912 0.055939 .
device_distinct_emails_8w    2.440e-02  3.106e-03   7.858 4.02e-15 ***
payment_typeAC               7.232e-03  1.308e-03   5.528 3.26e-08 ***
employment_statusCB         -4.395e-03  1.591e-03  -2.763 0.005736 **
employment_statusCC          7.506e-03  3.000e-03   2.502 0.012342 *
email_is_free1               7.029e-03  1.105e-03   6.359 2.05e-10 ***
housing_statusBB            -2.535e-02  1.822e-03 -13.909  < 2e-16 ***
housing_statusBC            -2.402e-02  1.774e-03 -13.536  < 2e-16 ***
housing_statusBD            -2.499e-02  3.658e-03  -6.832 8.50e-12 ***
housing_statusBE            -2.399e-02  2.097e-03 -11.441  < 2e-16 ***
housing_statusBF            -2.848e-02  1.300e-02  -2.191 0.028494 *
phone_home_valid1           -7.485e-03  1.189e-03  -6.293 3.15e-10 ***
phone_mobile_valid1          2.653e-03  1.839e-03   1.443 0.149149
has_other_cards1            -9.474e-03  1.359e-03  -6.974 3.14e-12 ***
foreign_request1             5.100e-03  3.469e-03   1.470 0.141515
sourceTELEAPP                3.479e-02  6.612e-03   5.261 1.44e-07 ***
device_osmacintosh           7.789e-03  2.470e-03   3.153 0.001618 **
device_oswindows             1.445e-02  1.271e-03  11.362  < 2e-16 ***
keep_alive_session1         -7.911e-03  1.132e-03  -6.988 2.84e-12 ***
month1                      -3.926e-03  1.871e-03  -2.098 0.035928 *
month2                      -3.913e-03  1.807e-03  -2.165 0.030375 *
month3                      -8.318e-03  1.710e-03  -4.863 1.16e-06 ***
month4                      -3.812e-03  1.794e-03  -2.125 0.033570 *
month6                      -3.367e-03  1.933e-03  -1.742 0.081533 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1017 on 34964 degrees of freedom
Multiple R-squared:  0.03614,   Adjusted R-squared:  0.03517
F-statistic: 37.45 on 35 and 34964 DF,  p-value: < 2.2e-16
```

```
> print(conf_matrixbacklin)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 10949    27
         1  3878   146

               Accuracy : 0.7397
                 95% CI : (0.7326, 0.7467)
    No Information Rate : 0.9885
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0485

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.73845
            Specificity : 0.84393
         Pos Pred Value : 0.99754
         Neg Pred Value : 0.03628
             Prevalence : 0.98847
         Detection Rate : 0.72993
   Detection Prevalence : 0.73173
      Balanced Accuracy : 0.79119

       'Positive' Class : 0
```

FIGURE 6 – Modèle Linéaire après Backward & Prédictions

Le modèle après avoir incanté avec la méthode "Backward" est légèrement meilleur, le $R^2$ augmente de 0.002 et la matrice de confusion est aussi meilleure ( plus de "1" bien prédit et moins de "0" prédit en "1") ( Nous n'analysons pas la sensibilité et la spécificité car celles-ci correspondent à la prédiction de "0" or c'est celle de "1" que nous souhaitons obtenir.

## 3.2 Modèle GLM

Nous avons ensuite voulu utiliser un supposé meilleur modèle que la régression linéaire basique, nous avons donc utilisés un modèle linéaire généralisé (GLM), ce modèle possède plusieurs avantages qui nous sont bénéfiques dans notre cas, contrairement aux modèles linéaires classiques qui supposent une distribution normale des résidus, les modèles linéaires généralisés permettent de modéliser des variables de réponse qui suivent d'autres distributions.
De plus, ce genre de modèle fonctionne bien avec tout types de variables, notre modèle possédant des variables quantitatives, catégorielles et binaires, nous n'avons donc pas besoin de séparer chaque variable catégorielle en plusieurs variables binaires.
Voici les résultats obtenus lors de l'application de ce modèle :

```
> summary(model)

Call:
glm(formula = fraud_bool ~ ., family = "binomial", data = data_balanced)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0703  -0.5656  -0.2651   0.4955   2.7368

Coefficients:
                                  Estimate Std. Error z value Pr(>|z|)
(Intercept)                     -3.695e+00  2.155e-01 -17.148  < 2e-16 ***
income                           8.820e-01  4.858e-02  18.156  < 2e-16 ***
name_email_similarity           -1.394e+00  4.459e-02 -31.272  < 2e-16 ***
prev_address_months_count        3.056e-03  1.557e-04  19.632  < 2e-16 ***
current_address_months_count     2.085e-03  1.612e-04  12.937  < 2e-16 ***
customer_age                     2.813e-02  1.264e-03  22.255  < 2e-16 ***
days_since_request               1.271e-02  1.875e-03   6.783 1.18e-11 ***
intended_balcon_amount           1.173e-03  4.623e-04   2.538  0.01114 *
payment_typeAB                   4.207e-01  3.870e-02  10.871  < 2e-16 ***
payment_typeAC                   7.674e-01  3.963e-02  19.365  < 2e-16 ***
payment_typeAD                   3.647e-01  4.899e-02   7.444 9.75e-14 ***
payment_typeAE                  -1.200e+01  2.489e+02  -0.048  0.96153
zip_count_4w                     1.715e-04  1.350e-05  12.706  < 2e-16 ***
velocity_6h                     -1.491e-05  5.249e-06  -2.840  0.00452 **
velocity_24h                     2.790e-05  1.113e-05   2.507  0.01218 *
velocity_4w                     -5.604e-05  2.875e-05  -1.949  0.05124 .
bank_branch_count_8w            -3.512e-04  3.296e-05 -10.655  < 2e-16 ***
date_of_birth_distinct_emails_4w -2.025e-02  3.152e-03  -6.424 1.33e-10 ***
employment_statusCB             -3.963e-01  4.050e-02  -9.785  < 2e-16 ***
employment_statusCC             -1.666e-01  5.693e-02  -2.926  0.00343 **
employment_statusCD             -6.761e-02  8.713e-02  -0.776  0.43782
employment_statusCE             -1.102e+00  1.566e-01  -7.039 1.93e-12 ***
employment_statusCF             -1.676e+00  1.278e-01 -13.107  < 2e-16 ***
employment_statusCG             -1.381e+01  2.055e+02  -0.067  0.94642
credit_risk_score                2.647e-03  2.328e-04  11.371  < 2e-16 ***
email_is_free1                   7.894e-01  2.748e-02  28.722  < 2e-16 ***
housing_statusBB                -1.375e+00  3.763e-02 -36.536  < 2e-16 ***
housing_statusBC                -1.322e+00  3.543e-02 -37.312  < 2e-16 ***
housing_statusBD                -1.208e+00  8.208e-02 -14.723  < 2e-16 ***
housing_statusBE                -1.959e+00  5.526e-02 -35.450  < 2e-16 ***
housing_statusBF                -1.402e+01  9.802e+01  -0.143  0.88629
housing_statusBG                -1.375e+01  2.938e+02  -0.047  0.96269
phone_home_valid1               -6.664e-01  2.946e-02 -22.618  < 2e-16 ***
phone_mobile_valid1              1.107e-01  4.216e-02   2.626  0.00864 **
bank_months_count                8.907e-03  1.114e-03   7.994 1.31e-15 ***
has_other_cards1                -9.931e-01  3.912e-02 -25.385  < 2e-16 ***
proposed_credit_limit            3.304e-05  3.102e-05   1.065  0.28684
foreign_request1                 3.899e-01  6.795e-02   5.738 9.57e-09 ***
source_TELEAPP                   1.837e+00  1.119e-01  16.427  < 2e-16 ***
session_length_in_minutes        9.163e-03  1.403e-03   6.533 6.47e-11 ***
device_osmacintosh               1.187e+00  5.575e-02  21.299  < 2e-16 ***
device_osother                   2.299e-01  3.814e-02   6.027 1.67e-09 ***
device_oswindows                 1.426e+00  3.397e-02  41.966  < 2e-16 ***
device_osx11                     1.209e-01  1.408e-01   0.859  0.39041
keep_alive_session1             -7.662e-01  2.694e-02 -28.447  < 2e-16 ***
device_distinct_emails_8w        1.160e+00  5.186e-02  22.366  < 2e-16 ***
month1                          -3.806e-01  5.822e-02  -6.538 6.24e-11 ***
month2                          -5.135e-01  5.728e-02  -8.965  < 2e-16 ***
month3                          -9.519e-01  6.248e-02 -15.236  < 2e-16 ***
month4                          -4.793e-01  6.702e-02  -7.153 8.52e-13 ***
month5                           2.966e-02  7.431e-02   0.399  0.68980
month6                          -4.137e-01  7.929e-02  -5.218 1.81e-07 ***
month7                           1.887e-03  9.697e-02   0.019  0.98448
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 65693  on 51741  degrees of freedom
Residual deviance: 39456  on 51689  degrees of freedom
AIC: 39562

Number of Fisher Scoring iterations: 13
```

FIGURE 7 – Modèle Linéaire généralisé

```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 13364    57
         1  1463   116

               Accuracy : 0.8987
                 95% CI : (0.8937, 0.9035)
    No Information Rate : 0.9885
    P-Value [Acc > NIR] : 1

                  Kappa : 0.114

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.90133
            Specificity : 0.67052
         Pos Pred Value : 0.99575
         Neg Pred Value : 0.07346
             Prevalence : 0.98847
         Detection Rate : 0.89093
   Detection Prevalence : 0.89473
      Balanced Accuracy : 0.78592

       'Positive' Class : 0
```

FIGURE 8 – Prédiction avec modèle Linéaire généralisé

Ce modèle étant lui aussi une régression, pour obtenir une prédiction de notre variable cible binaire, il est nécessaire de transformer le résultat de la prédiction en variable binaire en définissant une probabilité seuil, sous laquelle on définit la prédiction comme étant 0, et 1 dans le cas contraire. Les résultats de notre matrice de confusion dépendent beaucoup de ce seuil, nous avons utilisé ici 0.5.

## 3.3 Modèle SVM

Afin de ne pas avoir à être dépendant d'un seuil lors de la transformation d'une prédiction résultante d'une régression, nous avons ensuite testé un modèle permettant de faire une classification binaire, le modèle SVM avec un noyau radial pour la classification binaire de 'fraud_bool' en fonction des autres variables.
Avec ceci, nous obtenons les résultats suivants :

```
> print(conf_matrixSVM)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 14084    98
         1   743    75

               Accuracy : 0.9439
                 95% CI : (0.9401, 0.9476)
    No Information Rate : 0.9885
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1349

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.94989
            Specificity : 0.43353
         Pos Pred Value : 0.99309
         Neg Pred Value : 0.09169
             Prevalence : 0.98847
         Detection Rate : 0.93893
   Detection Prevalence : 0.94547
      Balanced Accuracy : 0.69171

       'Positive' Class : 0
```

FIGURE 9 – Prédiction avec modèle SVM

Suite à cela, nous avons voulu vérifier les performances de ce modèle en utilisant une technique de validation croisée avec la méthode 5-Fold, on obtient ceci :

```
> print(svm_model_cv)
Support Vector Machines with Radial Basis Function Kernel

51742 samples
   30 predictor
    2 classes: '0', '1'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 41393, 41394, 41394, 41393, 41394
Resampling results across tuning parameters:

  C     Accuracy   Kappa
  0.25  0.9259789  0.8342114
  0.50  0.9498859  0.8884654
  1.00  0.9705074  0.9345976

Tuning parameter 'sigma' was held constant at a value of 0.01220467
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.01220467 and C = 1.
```

FIGURE 10 – Résultats de la validation croisée

On observe que la meilleure valeur du paramètre C utilisé dans la méthode SVM est 1, la valeur d'Accuracy et de Kappa étant la meilleure pour ce choix-là.
Suite à cela on fait un nouveau modèle SVM en utilisant le résultat de la validation croisée et on obtient ceci :

```
> print(conf_matrixSVM_cv)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 14299   111
         1   528    62

               Accuracy : 0.9574
                 95% CI : (0.954, 0.9606)
    No Information Rate : 0.9885
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1473

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.9644
            Specificity : 0.3584
         Pos Pred Value : 0.9923
         Neg Pred Value : 0.1051
             Prevalence : 0.9885
         Detection Rate : 0.9533
   Detection Prevalence : 0.9607
      Balanced Accuracy : 0.6614

       'Positive' Class : 0
```

FIGURE 11 – Prédiction avec modèle SVM séléctionné par 5-Fold

## 3.4 Modèle Fôret Aléatoires

Enfin, nous utilisons la forêt aléatoire car c'est le modèle qui nous semble être le plus adapté à la détection de souscription frauduleuse en raison de sa capacité à bien classifier les variables catégorielles non équilibrer. Toutefois, un déséquilibre trop important peut entraîner un biais de notre modèle envers la classe majoritaire.

Ici, le suréchantillonnage de la classe "1" est bénéfique, car en augmentant le nombre de cas de fraude dans l'ensemble d'entraînement, le modèle a une meilleure chance d'apprendre les caractéristiques spécifiques associées aux fraudes. Cela peut améliorer la sensibilité du modèle. Après moult essais avec un échantillonage d'environ 1.1% de classe "1" qui ne nous apportaient que des modèle de prédisant que la classe "0", nous avons donc décidé de sur-échantilloner le jeu de donnée d'entraînements en classe "1", nous avons passé la proportion de classe "1" à 11% au lieu de 1.1%, et ainsi nous obtenons comme résultats :

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 51110     6
         1     0   606

               Accuracy : 0.9999
                 95% CI : (0.9997, 1)
    No Information Rate : 0.9882
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.995

 Mcnemar's Test P-Value : 0.04123

            Sensitivity : 1.0000
            Specificity : 0.9902
         Pos Pred Value : 0.9999
         Neg Pred Value : 1.0000
             Prevalence : 0.9882
         Detection Rate : 0.9882
   Detection Prevalence : 0.9883
      Balanced Accuracy : 0.9951

       'Positive' Class : 0
```

FIGURE 12 – Matrice de confusion pour le modèle de forêt aléatoire

Avec 50 arbres nous obtenons un excellent modèle avec très peu d'erreurs, avec une compilation plus longue mais toute de même abordable de 100 arbres le modèle devient parfait. Mais toute fois il faut garder en tête le sur-échantillonage de la classe minoritaire "1" qui forcément biaise légèrement les résultats.

Concernant l'importance des variables :

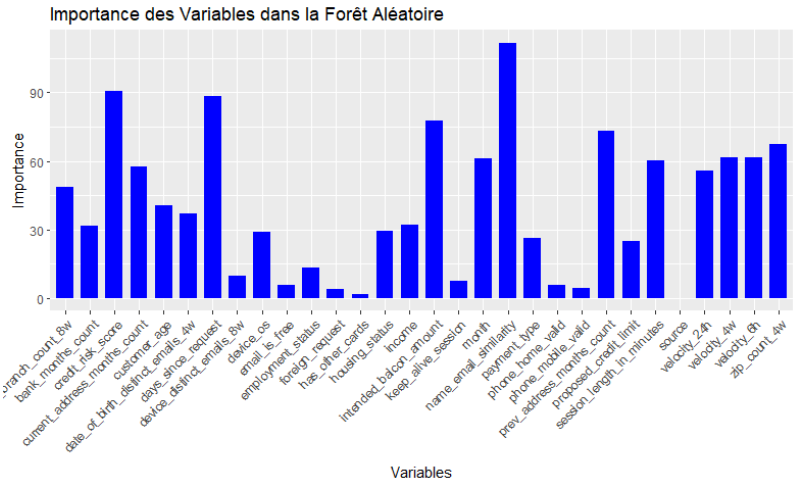



FIGURE 13 – Importance des variables pour la forêt aléatoire

On remarque que différentes observations faites au préalable sur nos données semblent être juste, en effet les variables **name_email_similarity**, **days_since_request**, **credit_risk_score** sont les plus importants dans notre forêt aléatoire c'est-à-dire celle qui discrimine le plus nos données.

# 4 Conclusion

## 4.1 Comparaison des Résultats

1. La métrique "Vitesse de Calcul" est importante afin de pouvoir créer nos modèles rapidement et de pouvoir les optimiser rapidement aussi, mais elle peut être facilement améliorer avec du meilleur matérie.
2. La "Sensibilité" ici nous est très utile, elle nous permet d'obtenir le F1-score.
3. La "Spécificité" ici est importante mais pas nécessaire, elle nous donne le taux de bonnes prédictions de la classe "1".
4. La "Précision de la classe "1" est très importante, c'est cette métrique qui nous permet de déterminer si on accuse les souscripteur en tord ou pas. Et aussi de déterminer le F1-score.
5. $R^2$/AIC sont des métriques relatives à certains modèles en particulier.
6. Enfin la métrique F1-Score est une métrique qui est un assemblement des métriques Sensibilité et Précision.

| Méthode | Vitesse Calcul | Sensibilité | Spécificité | Précision(Classe 1) | $R^2$/AIC | F1Score |
|---|---|---|---|---|---|---|
| Linéaire | rapide | 0.7366 | 0.8323 | 0.0356 | 0.035 | 0.068 |
| Linéaire(backward) | rapide | 0.7385 | **0.8439** | 0.03628 | 0.0351 | 0.069 |
| GLM | très rapide | 0.901 | 0.671 | 0.073 | 39562 | 0.136 |
| SVM | très long | 0.950 | 0.434 | 0.092 | - | 0.167 |
| SVM(5-fold) | très long | **0.964** | 0.358 | **0.105** | - | **0.189** |
| R-F | moyen | **1.000** | **0.99** | **1.00** | - | **1** |

En résumé, nos 5 premiers modèles sont des modèles imparfaits, la métrique de Sensibilité est plûtôt saitisfaisainte pour chaque, mais le F1-Score et la Précision qui sont des métriques très importantes pour nous ( afin de ne pas accuser à tord des souscripteurs de fraude ) ne sont pas satisfaisantes. Ces problèmes viennent de la distribution de la classe "1" qui est très minoritaire par rapport à la classe "0" entraînant évidemment une difficulté pour la prédire.

Soit on trouve une majorité des fraudeurs ( Haute Spécificté ) mais parmis ceux qu'on accusent beaucoup le seront à tord ( Précision basse )
Soit on diminue le pourcentage de fraudeurs que l'ont trouvent ( Basse Spécificité ) mais on limite

les fausses accusations à nos clients.

Nous pensons que limiter les fausses accusations est plus important que de trouver une majorité de fraudeurs nous mettons donc plus en avant la métrique "Précision" par rapport à la métrique "Spécificité".

Alors quels modèles chosir ?

Si on en croit nos résultats les 2 meilleurs modèles sur presques toutes les métrique d'évalutations sont les modèles SVM (avec une séparation des donnes Train/test par la méthode 5-Fold ) et la méthode Random Forest, les deux avec suréchantillonage de la classe "1" dans les données d'apprentissage.

La question de laquelle choisir est donc évidente. Il nous semblait naturel que la méthode Random Forest soit la meilleur méthode pour créer un modèle de classification, étant fait pour ça. Et il s'agit en effet du meilleur modèle peut importe la métrique. Par contre la méthode Random Forest sans suréchantillonage de la classe "1" ne trouvent quasiment aucune fraude, il est donc impératif de suréchantillonner pour que cette méthode fonctionne.

## 4.2 Ouverture

Nous avons vu que pour la prédiction d'une variable Binaire/catégorielle la question du suréchantillonnage joue un rôle prépondérant pour la création de modèle, en effet avec un certains équilibre des différentes classes la création d'un modèle efficace est beaucoup plus facile. Mais en faisant ce suréchantillonnage nous biaisons volontairement le jeu de donnée. Il y'a donc une question très intéressante sur le dosage de l'échantillonage à étudier, savoir si il faut mieux des modèles efficaces mais avec des proportions fictives ou des modèles moins efficaces mais avec des proportions de classes réelles.
Plusieurs ensembles de données relatifs à cette problématique étaient disponibles avec notre jeu de donnée initiale, tels que "Variant I" et "Variant II", qui présentent des caractéristiques différentes, notamment une disparité plus marquée entre les classes.

Lors de l'évaluation de nos modèles, nous avons opté pour la méthode k-fold pour la séparation de données en données d'apprentissage et données de Tests .Comparer cette approche à une méthode bootstrap pour la même méthode d'approximation aurait été intéressant, en particulier pour des modèles tels que la régression linéaire ou la forêt aléatoire.

# 5 Annexes

Le code utilisé est disponible après la description des variables ainsi qu'en pièces jointes. Voici la description de l'ensemble des variables de notre jeu de données :

**revenu** (numérique) : Revenu annuel du demandeur (sous forme décimale). Varie entre [0.1, 0.9].

**similarité_nom_email** (numérique) : Mesure de la similarité entre l'e-mail et le nom du demandeur. Des valeurs plus élevées représentent une similarité plus élevée. Varie entre [0, 1].

**mois_précédents_à_l'adresse** (numérique) : Nombre de mois à la précédente adresse enregistrée du demandeur, c'est-à-dire la résidence précédente du demandeur, le cas échéant. Varie entre [-1, 380] mois (-1 est une valeur manquante).

**mois_actuels_à_l'adresse** (numérique) : Mois à l'adresse actuellement enregistrée du demandeur. Varie entre [-1, 429] mois (-1 est une valeur manquante).

**âge_client** (numérique) : Âge du demandeur en années, arrondi à la décennie. Varie entre [10, 90] ans.

**jours_depuis_demande** (numérique) : Nombre de jours écoulés depuis la demande. Varie entre [0, 79] jours.

**montant_balcon_prévu** (numérique) : Montant initial transféré pour la demande. Varie entre [-16, 114] (les valeurs négatives sont des valeurs manquantes).

**type_paiement** (catégorique) : Type de plan de paiement à crédit. 5 valeurs possibles (anonymisées).

**nombre_applications_code_postal_4s** (numérique) : Nombre de demandes dans le même code postal au cours des 4 dernières semaines. Varie entre [1, 6830].

**vélocité_6h** (numérique) : Vélocité total des demandes effectuées au cours des 6 dernières heures, c'est-à-dire le nombre moyen de demandes par heure au cours des 6 dernières heures. Varie entre [-175, 16818].

**vélocité_24h** (numérique) : Vélocité total des demandes effectuées au cours des 24 dernières heures, c'est-à-dire le nombre moyen de demandes par heure au cours des 24 dernières heures. Varie entre [1297, 9586].

**vélocité_4s** (numérique) : Vélocité total des demandes effectuées au cours des 4 dernières semaines, c'est-à-dire le nombre moyen de demandes par heure au cours des 4 dernières semaines. Varie entre [2825, 7020].

**nombre_agences_bancaires_8s** (numérique) : Nombre total de demandes dans la succursale bancaire sélectionnée au cours des 8 dernières semaines. Varie entre [0, 2404].

**nombre_emails_distincts_naiss_4s** (numérique) : Nombre d'e-mails pour les demandeurs ayant la même date de naissance au cours des 4 dernières semaines. Varie entre [0, 39].

**statut_emploi** (catégorique) : Statut d'emploi du demandeur. 7 valeurs possibles (anonymisées).

**score_risque_crédit** (numérique) : Score interne du risque d'application. Varie entre [-191, 389].

**email_gratuit** (binaire) : Domaine de l'e-mail de la demande (gratuit ou payant).

**statut_logement** (catégorique) : Statut résidentiel actuel du demandeur. 7 valeurs possibles (anonymisées).

**téléphone_fixe_valide** (binaire) : Validité du téléphone fixe fourni.

**téléphone_mobile_valide** (binaire) : Validité du téléphone mobile fourni.

**mois_comptes_bancaires** (numérique) : Ancienneté du compte précédent (le cas échéant) en mois. Varie entre [-1, 32] mois (-1 est une valeur manquante).

**a_autres_cartes** (binaire) : Si le demandeur a d'autres cartes de la même compagnie bancaire.

**limite_crédit_proposée** (numérique) : Limite de crédit proposée par le demandeur. Varie entre [200, 2000].

**demande_étrangère** (binaire) : Si le pays d'origine de la demande est différent du pays de la banque.

**source** (catégorique) : Source en ligne de la demande. Soit navigateur (INTERNET) soit application (TELEAPP).

**durée_session_en_minutes** (numérique) : Durée de la session utilisateur sur le site Web bancaire en minutes. Varie entre [-1, 107] minutes (-1 est une valeur manquante).

**système_exploitation_appareil** (catégorique) : Système d'exploitation de l'appareil ayant effectué la demande. Valeurs possibles : Windows, macOS, Linux, X11 ou autre.

**garder_session_active** (binaire) : Option utilisateur sur la déconnexion de la session.

**nombre_emails_distincts_appareil** (numérique) : Nombre d'e-mails distincts sur le site Web bancaire à partir de l'appareil utilisé au cours des 8 dernières semaines. Varie entre [-1, 2] e-mails (-1 est une valeur manquante).

**nombre_fraudes_appareil** (numérique) : Nombre de demandes frauduleuses avec l'appareil utilisé. Varie entre [0, 1].

**mois** (numérique) : Mois où la demande a été faite. Varie entre [0, 7].

**fraude_bool** (binaire) : Si la demande est frauduleuse ou non.

```
install.packages("caret")
install.packages("pROC")
install.packages("Hmisc")

library("mice")
library(dplyr)
library("elasticnet")
library(caret)
library(pROC)
library("e1071")

library(Hmisc)
library(corrplot)

datatest = Base



#On créer une liste contenant les indices des variables qualitatives, et on va définir
#les variabels associées à ses indices comme étant as.factor
indices <- c(1,9,16,18:21,23,25,26,28,29,31,32)

for (i in indices) {
  datatest[[i]] <- as.factor(datatest[[i]])
}

#La variable device_fraud_count n'a qu'une seule modalité, on la supprime
datatest <- subset(datatest, select = -device_fraud_count)

#Dans ce dataset les données manquantes ne sont pas écrites comme NA
#On cherche donc à trouver et redéfinir toute les données manquantes
datatest$prev_address_months_count[datatest$prev_address_months_count == -1] <- NA
datatest$current_address_months_count[datatest$current_address_months_count == -1] <- NA
datatest$intended_balcon_amount[datatest$intended_balcon_amount < 0] <- NA
#Valeur négative velocity_6h pourquoi ?
datatest$bank_months_count[datatest$bank_months_count == -1] <- NA
datatest$session_length_in_minutes[datatest$session_length_in_minutes == -1] <- NA
datatest$device_distinct_emails_8w[datatest$device_distinct_emails_8w == -1] <- NA









#Analyse simple

#Pourcentage fraudeurs par type de paiements

percentage_data <- aggregate(fraud_bool ~ payment_type, data = datatest, FUN = function(x) mean(x ==
 1) * 100)
```

```
barplot(percentage_data$fraud_bool, names.arg = percentage_data$payment_type, col = "red",
        main = "Pourcentage de fraude par type de paiement",
        xlab = "Payment Type",
        ylab = "Pourcentage de fraude")
#Nous remarquons qu'en fonction du type de paiement nous n'avons pas le même taux de fraudeurs, le mode de paiement AC est bien plus utilisé par les fraudeurs
#A l'inverse les modes AA et AE sont beaucoup moins utilisé, on peut donc avoir + confiance en moyenne aux clients utilisants ces types de paiements là


#En fonction du revenu

hist(datatest$income[datatest$fraud_bool == 1], col = "red", main = "Pourcentage de fraudeurs en fonction du revenu",
     xlab = "Income", ylab = "Pourcentage de fraudeurs", xlim = range(datatest$income), freq = FALSE)

#Nous remarquons ici que ce sont les individus qui mettent un revenu très élevé( entre 0.6 et 0.9) qui ont tendance à frauder le plus en moyenne
#Nous pouvons y voir là une nécessité pour les fraudeur sd'exagérer les revenus afin de paraître plus "crédible" pour la banque ou pour obtenir des avantages
#Que les gens avec un faible revenus ne pourraient ne pas avoir


#En fonction de l'age

hist_data_age <- hist(datatest$customer_age[datatest$fraud_bool == 1], col = "red",
                      main = "Pourcentage de fraudeurs en fonction de l'âge",
                      xlab = "Âge du client", ylab = "Pourcentage de fraudeurs",
                      freq = FALSE)

# Ajouter des étiquettes aux barres
text(hist_data_age$mids, hist_data_age$density,
     labels = paste0(round(hist_data_age$density * 100, 2), "%"),
     pos = 3, col = "red")


#De la même manière les âges priviligiés par les fraudeurs sont compris entre 20 et 60ans, avec un pic à 40
#Ce qui peut de même paraître "plus naturels" pour un fraudeur



# En fonction de si l'utilisateur à un nom de domaine d'adresse mail payant ou gratuit
cross_table <- table(datatest$email_is_free, datatest$fraud_bool)
fraud_percentage <- (cross_table[,2] / rowSums(cross_table))*100

# Créer un histogramme
barplot(fraud_percentage, names.arg = c("Non Free", "Free"), col = "red",
        main = "Pourcentage de fraudeurs en fonction de email_is_free",
        xlab = "Email is Free or Not", ylab = "Pourcentage de fraudeurs",
        ylim = c(0, max(fraud_percentage) + 2))

#Nous observons quasiment 2x plus de fraudeurs en moyenne quand le domaine de l'adresse email est gratuite
#Ce sont des adresses plus faciles à créer et qui peuvent être créer en masse, donc utile pour tenter de fr
```

```
auder


# En fonction de la similarité nom/email

hist_data <- hist(datatest$name_email_similarity, breaks = 20, plot = FALSE)

fraud_percentages <- sapply(1:(length(hist_data$breaks)-1), function(i) {
  subset_data <- datatest$name_email_similarity >= hist_data$breaks[i] &
    datatest$name_email_similarity < hist_data$breaks[i+1]
  fraud_percentage <- mean(datatest$fraud_bool[subset_data] == 1) * 100
  return(fraud_percentage)
})

barplot(height = fraud_percentages, names.arg = hist_data$breaks[-1], col = "red",
        main = "Pourcentage de fraudeurs par catégorie de 'name_email_similarity'",
        xlab = "name_email_similarity", ylab = "Pourcentage de fraudeurs",
        ylim = c(0, max(fraud_percentages) + 1))

#Enfin nous observons que plus l'adresse mail n'a aucun lien avec l nom et prénom du client alors plus ell
e a de chances d'être détenu par un fraudeur




#Pourcentage de donnée manquante dans chaque variable
colMeans(is.na(datatest))*100

#Calcul du nombre d'éléments différents pour chaque variable
num_elements <- sapply(datatest, function(x) n_distinct(x, na.rm = TRUE))
#Affichage du nombre d'éléments différents pour chaque variable
print(num_elements)


summary(datatest)
str(datatest)


#On génère des données pour remplacer les valeurs manquantes avec le
#package mice et la méthode pmm
#Ceci étant long à faire, il a été exécuté une fois et le résultat a
#été téléchargé afin d'être réutilisé sans attendre de nouveau
#imputed_data <- mice(datatest, m = 15, method = "pmm", seed = 123)
#completed_data=complete(imputed_data)

#Dans completed_data il faut redéfinir les variables comme étant as.factor
indices <- c(1,9,16,18:21,23,25,26,28,29,31)
for (i in indices) {
  completed_data[[i]] <- as.factor(completed_data[[i]])
}
```

```
#proportion_train <- 0.7
# Générer des indices aléatoires pour l'apprentissage et le test
#set.seed(123)  # Pour reproduire les résultats
#indices_train <- sample(nrow(completed_data), proportion_train * nrow(completed_data))
# Créer les ensembles d'apprentissage (train) et de test (test)
#data_train <- completed_data[indices_train, ]
#data_test <- completed_data[-indices_train, ]



#Si on veut faire des test avec un nombre de ligne choisies aléatoirement
# Déterminer le nombre total de lignes dans votre ensemble de données
nb_lignes_totales <- nrow(completed_data)
# Définir le nombre de lignes que vous voulez garder (100000 dans cet exemple)
nb_lignes_a_garder <- 50000
# Sélectionner aléatoirement les indices des lignes à garder
indices_aleatoires <- sample(1:nb_lignes_totales, nb_lignes_a_garder)
# Sélectionner les lignes correspondantes à ces indices
completed_data_reduced <- completed_data[indices_aleatoires, ]

proportion_train <- 0.7
# Générer des indices aléatoires pour l'apprentissage et le test
set.seed(123)  # Pour reproduire les résultats
indices_train <- sample(nrow(completed_data_reduced), proportion_train * nrow(completed_data_reduce
d))
# Créer les ensembles d'apprentissage (train) et de test (test)
data_train <- completed_data_reduced[indices_train, ]
data_test <- completed_data_reduced[-indices_train, ]


# Séparer les classes majoritaires et minoritaires
majority_class <- data_train[data_train$fraud_bool == 0, ]
minority_class <- data_train[data_train$fraud_bool == 1, ]

# Suréchantillonnage de la classe minoritaire
set.seed(123)  # Définir une graine aléatoire pour la reproductibilité
oversampled_minority <- minority_class[sample(nrow(minority_class), replace = TRUE, size = (nrow(majo
rity_class) - nrow(minority_class))/2), ]

# Combiner les classes majoritaire et suréchantillonnée de la classe minoritaire
data_balanced <- rbind(majority_class, oversampled_minority)




##########################################
#1ER MODEL AVEC GLM


model <- glm(fraud_bool ~ . , data = data_balanced, family = "binomial")
summary(model)
```

```
# Prédiction sur les données de test
predictions <- predict(model, newdata = data_test, type = "response")
binary_predictions <- as.factor(ifelse(predictions > 0.5, 1, 0))
# Afficher le pourcentage de réussite de la prédiction
sum (as.numeric(binary_predictions) == as.numeric(data_test$fraud_bool))/dim(data_test)[1]*100

#Calcul de la matrice de confusion
conf_matrix <- confusionMatrix(as.factor(binary_predictions), as.factor(data_test$fraud_bool))
# Affichage de la matrice de confusion
print(conf_matrix)



##########################################




##########################################
#SVM
svm_model <- svm(fraud_bool ~ ., data = data_balanced, kernel = "radial")
# Prédiction sur les données de test
predictionsSVM <- predict(svm_model, newdata = data_test)

# Affichage des résultats ou évaluation du modèle

sum (as.numeric(predictionsSVM) == as.numeric(data_test$fraud_bool))/dim(data_test)[1]*100
#Calcul de la matrice de confusion
conf_matrixSVM <- confusionMatrix(as.factor(predictionsSVM), as.factor(data_test$fraud_bool))
# Affichage de la matrice de confusion
print(conf_matrixSVM)



#ON APPLIQUE UNE METHODE 5 FOLD
control <- trainControl(method = "cv", number = 5)
svm_model_cv <- train(fraud_bool ~ .,
                      data = data_balanced,
                      method = "svmRadial",
                      trControl = control)
print(svm_model_cv)
predictionsSVM_cv <- predict(svm_model_cv, newdata = data_test)

sum (as.numeric(predictionsSVM_cv)-1 == as.numeric(data_test$fraud_bool)-1)/dim(data_test)[1]*100
conf_matrixSVM_cv <- confusionMatrix(predictionsSVM_cv, data_test$fraud_bool)
print(conf_matrixSVM_cv)
##############################################
```

```
#Le but de ce paragraphe est de transformer les variables quali en quanti


datatest_indicators <- model.matrix(~.-1, data = completed_data)
# Création d'un ensemble de données à partir de la matrice d'indicateurs
dataset_from_indicators <- data.frame(datatest_indicators)
# Attribuer les noms de colonnes appropriés
colnames(dataset_from_indicators) <- colnames(datatest_indicators)
#Calcul du nombre d'éléments différents pour chaque variable
num_elements2 <- sapply(dataset_from_indicators, function(x) n_distinct(x, na.rm = TRUE))
#Affichage du nombre d'éléments différents pour chaque variable
print(num_elements2)



#Créer une matrice de corrélation

correlation_matrix=cor(dataset_from_indicators)
heatmap(correlation_matrix,
        col = colorRampPalette(c("blue","white","red"))(20),
        symm = TRUE,
        margins=c(10,10))




# Calcul de la corrélation de "fraud_bool1" avec toutes les autres variables
correlation_specific_variable <- cor(dataset_from_indicators["fraud_bool1"], dataset_from_indicators[, -c(1
, 2)])
# Affichage des corrélations
print(correlation_specific_variable)
barplot(abs(correlation_specific_variable),
        main = "Corrélation de fraudbool1 avec les autres variables",
        col = "skyblue",
        xlab = "Variables",
        ylab = "Valeurs absolues de corrélation")


#Toute la partie ci dessous sert tout d'abord à transormer les variables
#quali en quanti, puis de suréchantilloner la classe minoritaire

#on transforme la variable cible en numeric pour le modèle linéraire
completed_data_reduced[[1]] <- as.numeric(completed_data_reduced[[1]])-1
variables_qualitatives <- completed_data_reduced[, sapply(completed_data_reduced, is.factor) | sapply(c
ompleted_data_reduced, is.character)]

#Appliquer l'encodage one-hot pour toutes les variables qualitatives
encoded_data <- model.matrix(~ . - 1, data = variables_qualitatives)

#Combiner les variables encodées avec les autres variables du DataFrame
data_encoded <- cbind(completed_data_reduced[, !names(completed_data_reduced) %in% names(varia
bles_qualitatives)], encoded_data)


# Créer les ensembles d'apprentissage (train) et de test (test) en réutilisant
```

```
#les mêmes indices que pour la création des train test précédent
data_train3 <- data_encoded[indices_train, ]
data_test3 <- data_encoded[-indices_train, ]


# Séparer les classes majoritaires et minoritaires
majority_class <- data_train3[data_train3$fraud_bool == 0, ]
minority_class <- data_train3[data_train3$fraud_bool == 1, ]
# Suréchantillonnage de la classe minoritaire
set.seed(123)  # Définir une graine aléatoire pour la reproductibilité
oversampled_minority <- minority_class[sample(nrow(minority_class), replace = TRUE, size = (nrow(majo
rity_class) - nrow(minority_class))/2), ]

# Combiner les classes majoritaire et suréchantillonnée de la classe minoritaire
data_balanced3 <- rbind(majority_class, oversampled_minority)




#Exemple de régression linéaire avec les nouvelles variables encodées
#########################################################

modellin <- lm(fraud_bool ~ ., data = data_train3)
summary(modellin)


predictionslin <- predict(modellin, newdata = data_test3,type = "response")
predictionslin <- as.factor(ifelse(predictionslin > 0.02, 1, 0))
# Affichage des résultats ou évaluation du modèle
sum (as.numeric(predictionslin)-1 == as.numeric(data_test3$fraud_bool))/dim(data_test3)[1]*100
#Calcul de la matrice de confusion
conf_matrixlin <- confusionMatrix(as.factor(predictionslin), as.factor(data_test3$fraud_bool))
# Affichage de la matrice de confusion
print(conf_matrixlin)

#################################################




#Modèle précédent après avoir utilisé la méthode backward pour
#réduire la dimension du modèle
#################################################
#On applique la méthode backward au modèle
modelbacklin <- step(modellin, direction = "backward")
summary(modelbacklin)  # Afficher les résultats du nouveau modèle
#après sélection des variables

#Faire des prédictions sur l'ensemble de test
predictionsbacklin <- predict(modelbacklin, newdata = data_test3)
#Ajuster la proba pour varier de beaucoup les résultats
predictionsbacklin <- as.factor(ifelse(predictionsbacklin > 0.02, 1, 0))
# Affichage des résultats ou évaluation du modèle
```

```
sum (as.numeric(predictionsbacklin) == as.numeric(data_test$fraud_bool))/dim(data_test)[1]*100
#Calcul de la matrice de confusion
conf_matrixbacklin <- confusionMatrix(as.factor(predictionsbacklin), as.factor(data_test3$fraud_bool))
# Affichage de la matrice de confusion
print(conf_matrixbacklin)
##############################################################################



##############################################################################################
#########
########################################Foret aléatoire#################################
######
##############################################################################################
#########
X_test <- subset(data_balanced, select = -fraud_bool)
y_test <- data_balanced$fraud_bool

formula <- fraud_bool ~ .

# Modèle foret aléatoire
rf_model <- randomForest(formula, data = data_balanced, ntree = 50, importance = TRUE)

predictionsforet <- predict(rf_model, newdata = X_test)

# Matrice de confusion
conf_matrixforet <- table(Prédiction = predictionsforet, Observation = y_test)
print("Matrice de Confusion :")
print(conf_matrixforet)

var_importance <- importance(rf_model)
print("Importance des Variables :")
print(var_importance)

confusionMatrix(as.factor(predictions), as.factor(y_test))

var_importance <- importance(rf_model)

importance_df <- as.data.frame(var_importance)

importance_df$Variables <- row.names(var_importance)

importance_df <- importance_df[order(-importance_df$MeanDecreaseGini), ]

ggplot(importance_df, aes(x = Variables, y = MeanDecreaseGini)) +
  geom_bar(stat = "identity", fill = "blue", width = 0.7) +
  labs(title = "Importance des Variables dans la Forêt Aléatoire",
       x = "Variables",
       y = "Importance") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```