*PRESENTATION*

# « Building Claim Prediction »

Par Generali

Master 1 ingénierie économique et financière – Data science
POUPONNEAU Guillaume – AGRAFFEL Nicolas

Université de Rennes

# Sommaire :



**Challenge :** Challenge data (ens.fr)

# Objectif

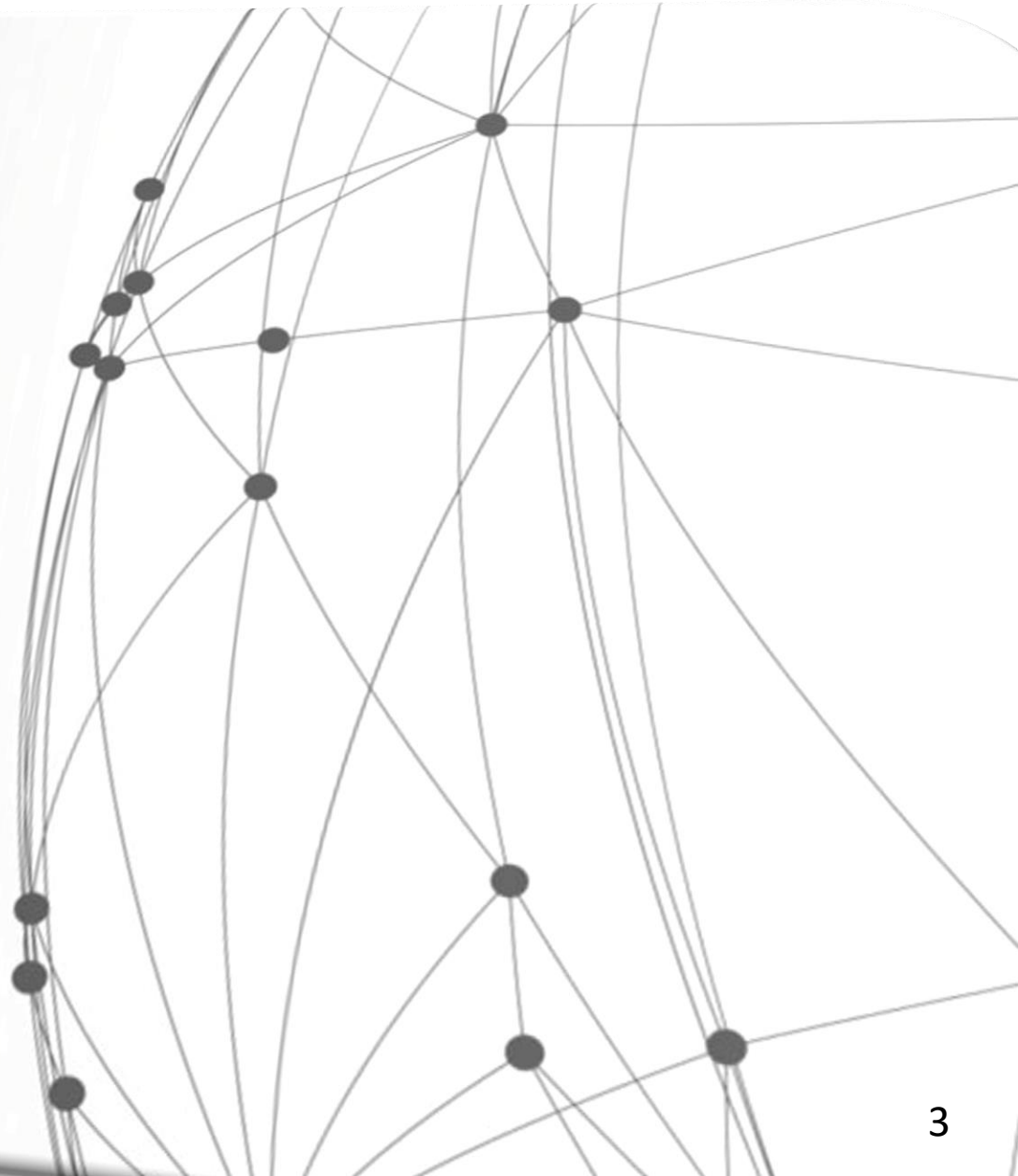
Le but du défi est de prédire si un bâtiment fera l'objet d'une réclamation d'assurance pendant une certaine période. On devra prévoir une probabilité d'avoir **au moins un sinistre** sur la période assurée d'un immeuble.

Le modèle sera basé sur les caractéristiques du bâtiment. La variable cible est un :

- **1** si l'immeuble présente **au moins un sinistre** sur la période assurée.
- **0** si le bâtiment ne présente **aucun sinistre** sur la période assurée. Lors de ce défi, on est encouragés à utiliser des données externes. Par exemple : numéro de commerces par code INSEE (code géographique), taux de chômage par code INSEE, météo…

# Les variables

- Description
- Nettoyage
- Analyse



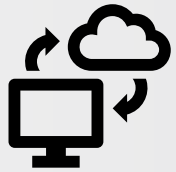
# La base de données :

La base de donnée contient beaucoup de variable, il est donc nécessaire de les **étudier** dans un premier temps

**Identifiant** : Variable pour identifier le client
**Ft_2_categ** : année d'observation
**Expo** : Temps assuré par Generali au cours de l'année (1 = 1an, 0,5 = 6 mois)
**Superficief** : superficie assuré en mètre carré
**Insee** : Code insee, pour localiser le bâtiment
**Target** : variable cible (binaire) : 0 = pas de sinistre, 1 = au moins un sinistre au cours de l'année
**Ft_i_categ** : variable catégoriel anonyme, qui sont des caractéristiques du bâtiment

Le sujet nous indique de lui-même que les variables « **superficief, ft_22_categ, EXPO** » sont les plus importantes.

# Une meilleur base de donnée :



En faisant des recherches sur des données externe sur internet→ base de donnée plus complète trouvée* , on fera donc une première étude dessus

+ De variables
- Criminalité
- La pluie
- Le taux de chômage
- Le revenu médian
- Les fuites
- Le nombre d'entreprises, de logements sociaux,
- Catastrophe naturelle par année

**Nous rajouterons cependant, grâce au code INSEE, des variables externe plus tard*

# Description de la base

```
> str(X_train)
'data.frame':   10110
 $ X           : int
 $ Identifiant: int
 $ ft_2_categ : int
 $ EXPO        : chr
 $ ft_4_categ : int
 $ ft_5_categ : chr
 $ ft_6_categ : chr
 $ ft_7_categ : chr
 $ ft_8_categ : chr
 $ ft_9_categ : chr
 $ ft_10_categ: chr
 $ ft_11_categ: chr
 $ ft_12_categ: chr
 $ ft_13_categ: chr
 $ ft_14_categ: chr
 $ ft_15_categ: chr
 $ ft_16_categ: chr
 $ ft_17_categ: chr
 $ ft_18_categ: chr
 $ ft_19_categ: int
 $ superficief: num
 $ ft_21_categ: int
 $ ft_22_categ: num
 $ ft_23_categ: num
 $ ft_24_categ: chr
 $ Insee       : chr
 $ pluie       : int
 $ catnat      : num
 $ criminalite: num
 $ pprn        : int
 $ chomage     : num
 $ rev_med     : num
 $ log_sociaux: int
 $ fuites      : num
 $ entreprises: num
>
```

-9 variables int
-17 variables en character
-9 variables numériques

Les variables **character** étant trop importantes pour être omises, il faut les **transformer en numérique** pour pouvoir les intégrer aux futurs modèles.

**One-hot-encoding**

# Description de la base

1- **Regroupement des bases de données** Train et Test pour Faciliter les futurs changements
2- **Etude basique** : Dim, Head, summary, str
3- **Transformation de la variable « EXPO » en numérique**, car utilisé dans le modèle plus tard

```
> summary(X_train_test) X Identifiant ft_2_categ EXPO ft_4_categ ft_19_categ Min.   :    0 Min.   :    0
Min.   :2012 Min.   :0.0000 Min.   :0.000 Min.   :1.000 1st Qu.:  1690 1st Qu.:  5198 1st Qu.:2012 1st
Qu.:1.0000 1st Qu.:0.000 1st Qu.:2.000 Median :  3380 Median :  9736 Median :2013 Median :1.0000
Median :0.000 Median :2.000 Mean   :  4209 Mean   :  9927 Mean   :2014 Mean   :0.9135 Mean   :0.278 Mean
:1.939 3rd Qu.:  6729 3rd Qu.:14843 3rd Qu.:2015 3rd Qu.:1.0000 3rd Qu.:1.000 3rd Qu.:2.000
Max.   :10109 Max.   :19995 Max.   :2016 Max.   :1.0000 Max.   :1.000 Max.   :2.000 superficief
ft_21_categ ft_22_categ Insee pluie catnat Min.   :  1 Min.   :1.000 Min.   : -1 Length:13522 Min.   :
0.000 Min.   : 0.000 1st Qu.:  500 1st Qu.:2.000 1st Qu.:1950 Class :character 1st Qu.:  1.000 1st
Qu.:  0.000 Median :  1002 Median :2.000 Median :1960 Mode :character Median :  4.000 Median :
0.000 Mean   :  1810 Mean   :2.243 Mean   :1727 Mean   :  5.291 Mean   :  3.544 3rd Qu.:  2192 3rd Qu.:3.000
3rd Qu.:1980 3rd Qu.:  9.000 3rd Qu.:  0.000 Max.   :34644 Max.   :4.000 Max.   :2016 Max.   :29.000
Max.   :182.000 criminalite pprn chomage rev_med log_sociaux fuites Min.   :0.01954 Min.   :  0.00
Min.   :  5.950 Min.   :11672 Min.   :  240.0 Min.   :  0.825 1st Qu.:0.04604 1st Qu.:  0.00 1st Qu.:
8.275 1st Qu.:18300 1st Qu.:  514.0 1st Qu.:  4.064 Median :0.06005 Median :  0.00 Median :  9.925
Median :19862 Median :  692.0 Median :  5.824 Mean   :0.06249 Mean   :  3.89 Mean   :10.009 Mean   :20809
Mean   :  733.3 Mean   :  6.555 3rd Qu.:0.07696 3rd Qu.:  3.00 3rd Qu.:11.675 3rd Qu.:22153 3rd Qu.:
836.0 3rd Qu.:  8.896 Max.   :0.12482 Max.   :354.00 Max.   :15.550 Max.   :46251 Max.   :1360.0 Max.
:19.092 entreprises Min.   :  1630 1st Qu.:  12596 Median :  19220 Mean   :  26403 3rd Qu.:  32124 Max.
:100837
```

*Summary : fournit un résumé statistique des données*

```
> str(X_train_test) 'data.frame': 13522 obs. of 19 variables: $ X : int 0 1 2 3 4 5 6 7 8 9 ...
$ Identifiant: int 18702 3877 4942 13428 17137 7580 11501 9102 11999 9772 ... $ ft_2_categ : int
2014 2014 2013 2013 2015 2016 2015 2016 2016 2013 ... $ EXPO : num 1 1 1 0.247 1 ... $
ft_4_categ : int 0 0 1 0 0 1 0 1 0 0 ... $ ft_19_categ: int 2 2 2 2 2 2 2 2 2 2 ... $
superficief: num 1351 1972 1630 532 1050 ... $ ft_21_categ: int 4 2 4 3 2 2 1 3 1 2 ... $
ft_22_categ: num 2012 1980 -1 -1 1972 ... $ Insee : chr "65440" "14341" "75109" "92004" ... $
pluie : int 7 1 0 0 1 4 1 2 3 9 ... $ catnat : num 1 0 0 0 0 0 0 0 39 0 ... $ criminalite: num
0.0441 0.0393 0.1093 0.0636 0.0624 ... $ pprn : int 1 0 0 0 67 4 0 24 13 0 ... $ chomage : num
11.35 9.75 8.28 7.75 12.95 ... $ rev_med : num 17098 19768 32167 23321 20914 ... $ log_sociaux:
num 556 826 1054 1240 1025 ... $ fuites : num 7.59 2.64 15.67 4.06 6.07 ... $ entreprises: num
4756 12025 100837 31493 38651 ...
```

*Str : fournit un aperçu concis de la structure interne*

# Cas des NA's

Le traitement des NA est délicat

```
colSums(is.na(Train)) #119 dans superficief, 1236 avec ft_22_categ
```

```
Identifiant           X  ft_2_categ        EXPO  ft_4_categ  ft_5_categ  ft_6_categ  ft_7_categ  ft_8_categ  ft_9_categ
          0           0           0           0           0           0           0           0           0           0
ft_10_categ ft_11_categ ft_12_categ ft_13_categ ft_14_categ ft_15_categ ft_16_categ ft_17_categ ft_18_categ ft_19_categ
          0           0           0           0           0           0           0           0           0           0
superficief ft_21_categ ft_22_categ ft_23_categ ft_24_categ       Insee         X.y      target
        119           0        1236           0           0           0           0           0
```

*colSums(is.na(Train))*

NA présent dans plusieurs variables :
-Certaines catégorielles
-Certaines numériques

*

```
# On remplace les NA de superficief et ft_22_categ par la médiane
Train[is.na(Train$superficief),]$superficief <- median(Train$superficief,na.rm = T)
Train[is.na(Train$ft_22_categ),]$ft_22_categ <- median(Train$ft_22_categ,na.rm = T)
X_test[is.na(X_test$ft_22_categ),]$ft_22_categ <- median(X_test$ft_22_categ,na.rm = T)
X_test[is.na(X_test$superficief),]$superficief <- median(X_test$superficief,na.rm = T)
```

**Dans la base de donné trouvée sur internet, les NA ont déjà été traités, mais voici comment on réglait le problème avant d'utiliser cette base.*
***Certain NA sont apparus après, lors de la fusion des 2 bases de données sur des variables catégorielles, possible erreur ?*

**Pour les numériques** : Superficief  (taille), ft_22_categ (année) :

**Possibilités :**

- Moyenne/médiane → **Notre choix**, le plus logique
- Mettre la valeur 0 → Incohérent, valeurs étant une taille, une année
- Supprimer → Impossible pour utiliser le modèle ensuite
- Ne rien faire → Impossible pour utiliser le modèle ensuite

```
Train[is.na(Train$superficief),]$superficief <- median(Train$superficief,na.rm = T)
Train[is.na(Train$ft_22_categ),]$ft_22_categ <- median(Train$ft_22_categ,na.rm = T)
X_test[is.na(X_test$ft_22_categ),]$ft_22_categ <- median(X_test$ft_22_categ,na.rm = T)
X_test[is.na(X_test$superficief),]$superficief <- median(X_test$superficief,na.rm = T)
```

**Pour les catégorielles** :

Nous avons fait le choix de remplacer les NAs des variables catégorielles par le mode de la variable (la valeur la plus fréquente pour plusieurs raisons:

- On conserve les données
- Le remplacement par le mode maintient la distribution de la valeur catégorielle
- On maintient les relations entre les variables

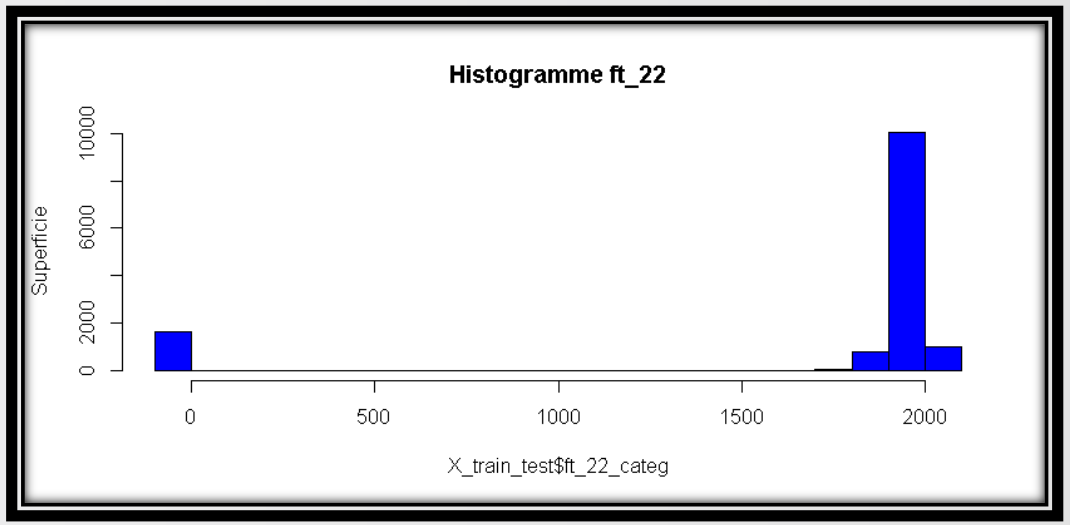
```r
df <- lapply(Test, function(x) {
  if(is.numeric(x)) {
    # Pour les variables numériques, remplace NA par la moyenne
    x[is.na(x)] <- mean(x, na.rm = TRUE)
  } else if(is.character(x)) {
    # Pour les variables de type caractère, remplace NA par le mode
    modeValue <- getMode(x[!is.na(x)]) # Calcule le mode des valeurs non-NA
    x[is.na(x)] <- modeValue
  }
  return(x)
})
```

## Choix médiane ou moyenne ?

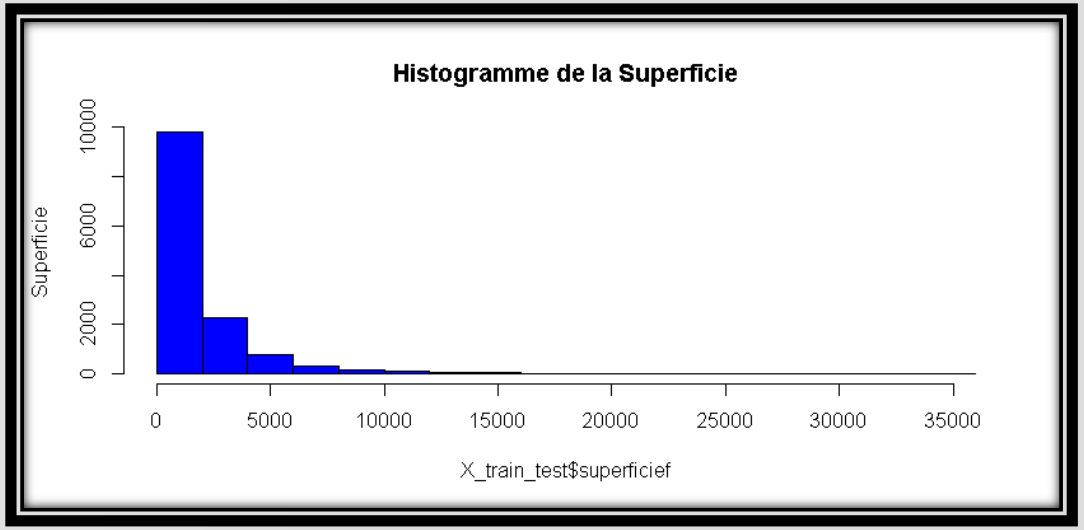
Moyenne : si distribution symétrique
→ Problème si valeurs aberrantes
Médiane : Si distribution asymétrique

→

Distribution pas vraiment symétrique :
→ Choix = **Médiane**



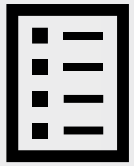

*hist(X_train_test$ft_22_categ, main="Histogramme ft_22", ylab="Superficie", col="blue")*




*hist(X_train_test$superficief, main="Histogramme de la Superficie", ylab="Superficie", col="blue")*

# One-hot encoding

**Définition** : Le One-Hot Encoding convertit les variables catégorielles en une forme qui peut être fournie aux algorithmes d'apprentissage automatique. Voici comment cela fonctionne :



Exemple : **ft_6_categ** a trois choix possibles : 1 0 ou v

On va faire trois variables avec 1 ou 0 : **ft_6_categ.1**, **ft_6_categ.0** et **ft_6_categ.v**

**Avantages** :

- Transformation en variable numérique
→ Utilisable dans un modèle
- Permet de les utiliser malgré leurs anonymats

**Inconvénients :**

- Augmentation dimensionnalité
→ Ici pas un problème
- Perte d'information sur les relations
→ Variables anonymes, donc pas de problème

# Autres ajustement nécessaire

-Renommer certaines données → Remplacer par des « _ » (enlever les espaces, les plus etc…)

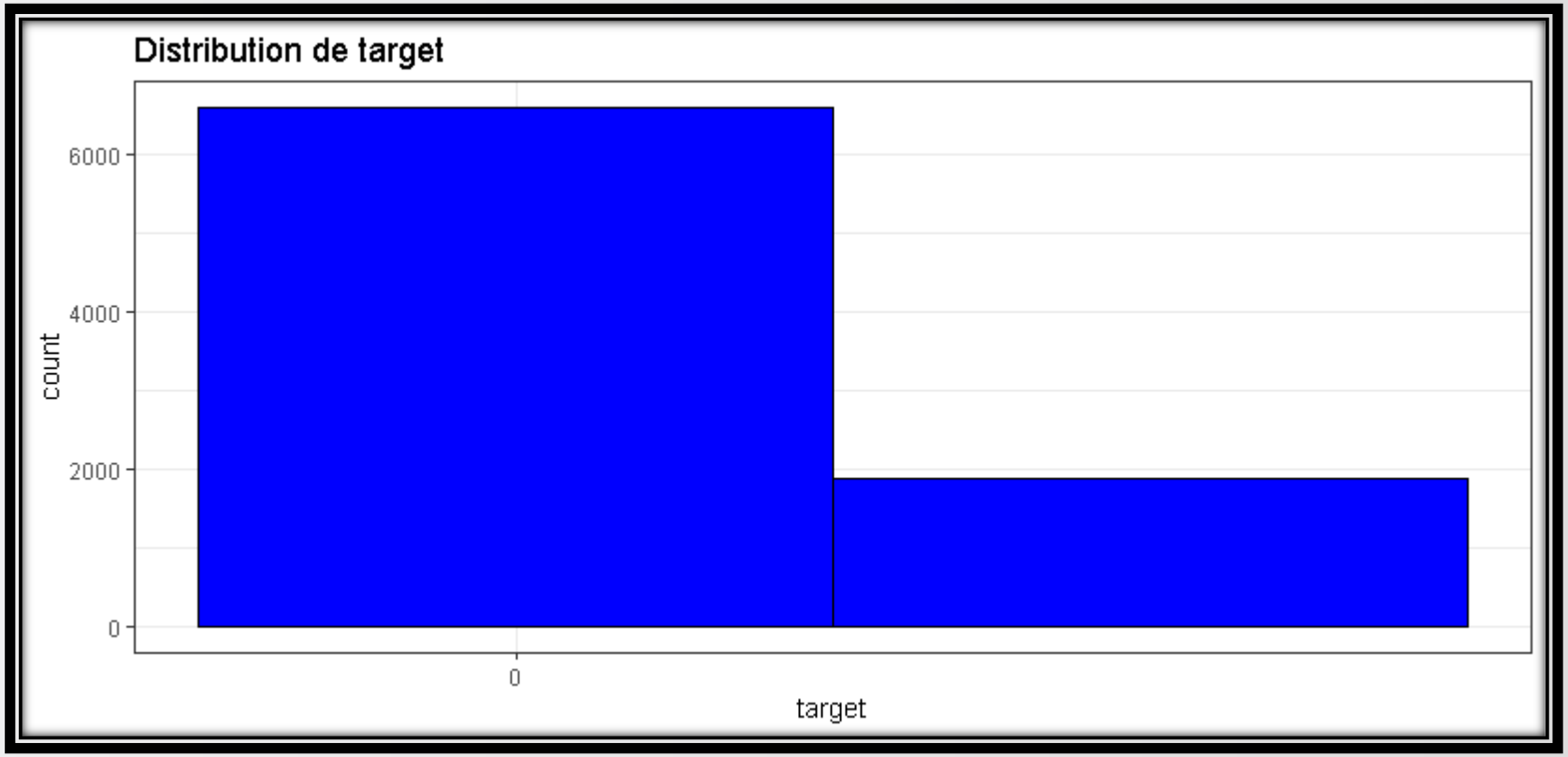
-Problème des -1 : variable « EXPO » contient des -1
Possible erreur lors de nos manipulations ?
→ Remplacer par la moyenne (faute de solution)
Impossible de les laisser, incohérence

- Re-séparation de X-train et X-test
→ Les modifications nécessaires aux deux ont été faites

- Création de la base d'entrainement « Train »
→ Merge X_train et Y_train

# Analyse Univariée

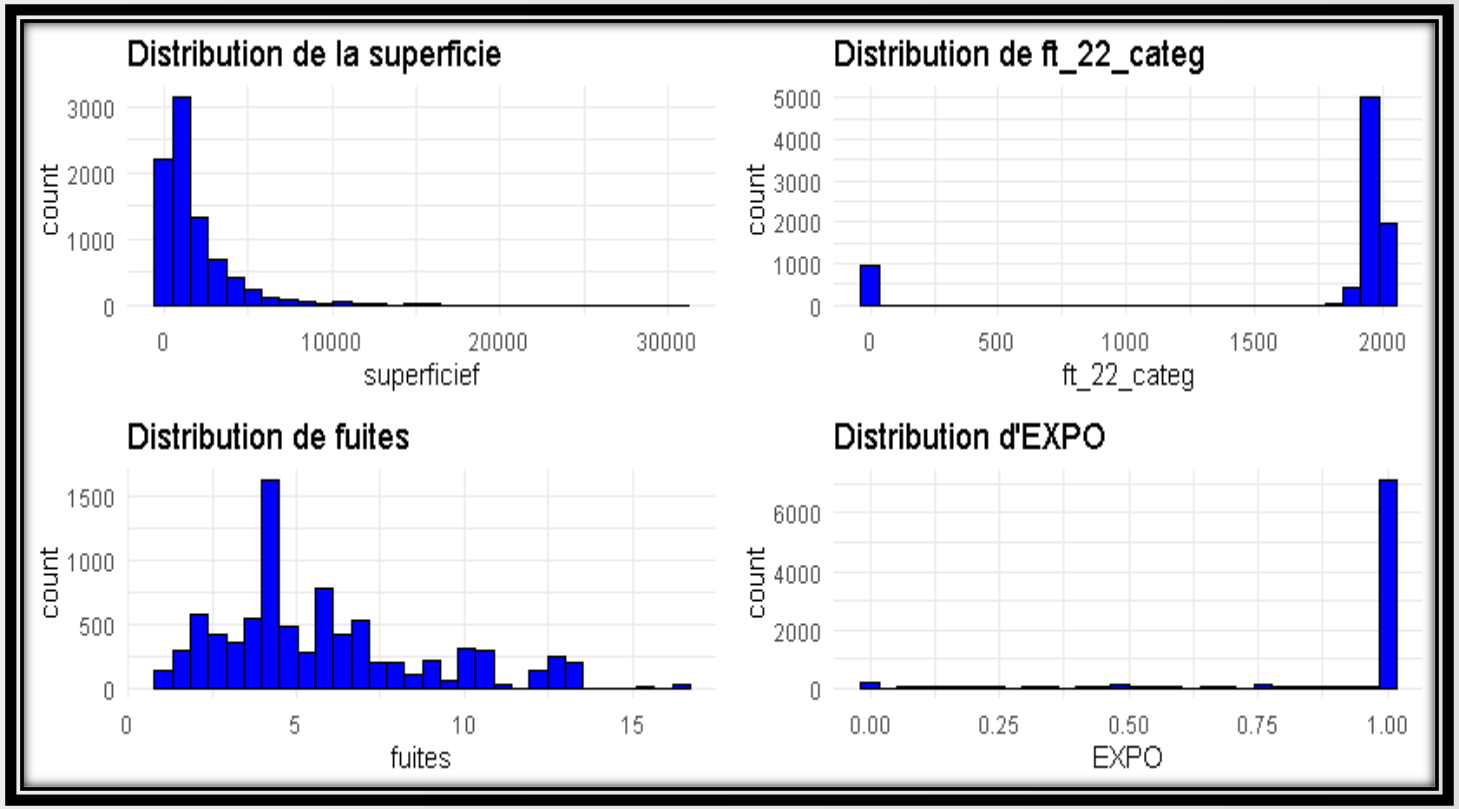
L'analyse univariée est cruciale pour comprendre les caractéristiques individuelles de chaque variable (valeurs aberrantes, asymétrie…)




Analyse variable cible :
Beaucoup plus de 0 que de 1
Modèle pourrait être biaisé en faveur de 0
→ Possible rééchantillonnage nécessaire

```
Train %>%  ggplot(aes(x = target))
+ geom_histogram(binwidth = 1, fill = "blue", color =
"black")
+  scale_x_continuous(breaks = seq(0, 90, 5))
 +  labs(title = "Distribution de target")
```

Distribution de certaine variables jugées importantes :



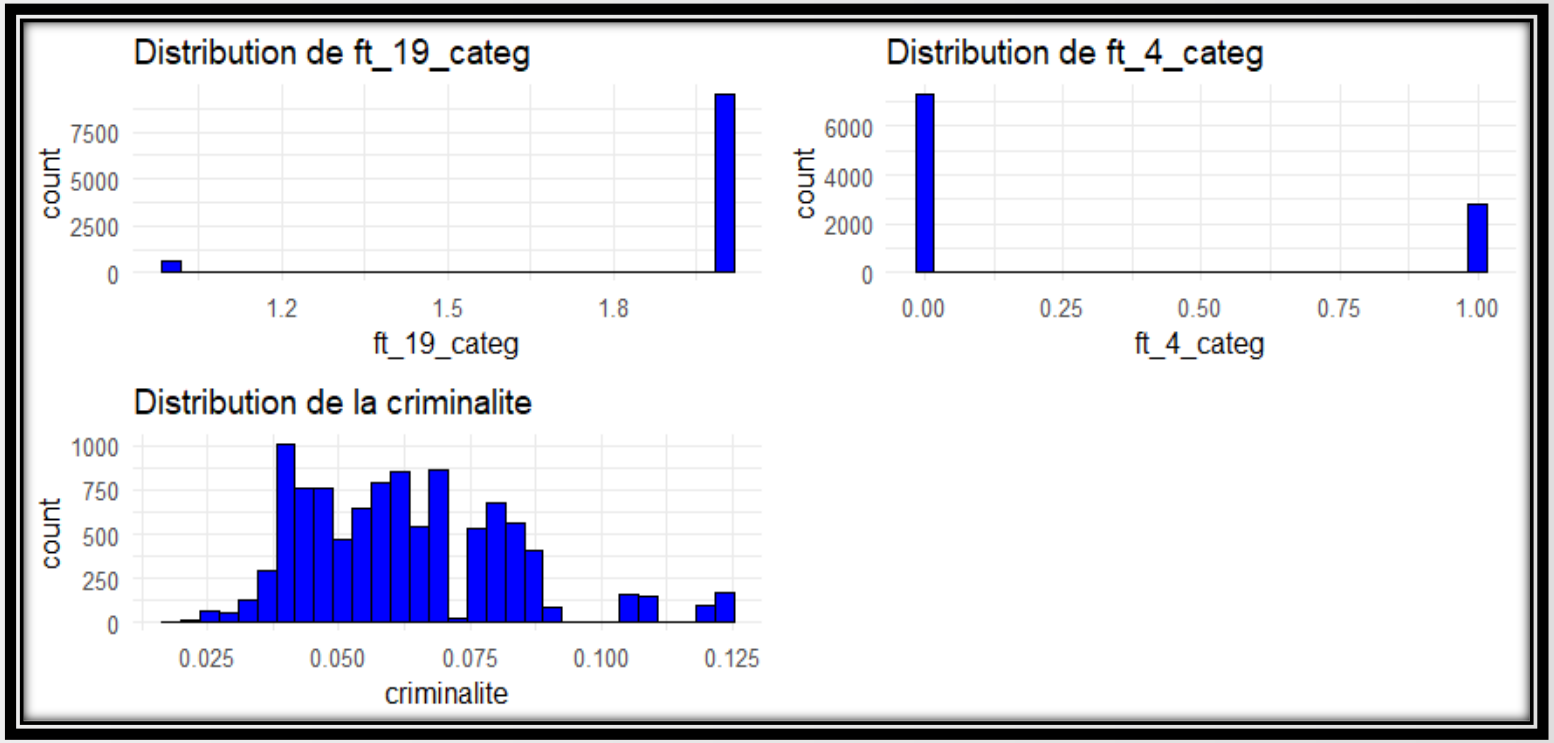

**Superficie** : Répartition biaisé →Possible transformation logarithmique nécessaire

**Ft_22_categ** : Présence valeur aberrante (des 0 ne font pas sens)

**Fuites** : Potentielles valeurs aberrantes ?

```
histogram1 <- ggplot(Train, aes(x = superficief)) +
 geom_histogram(bins = 30, fill = "blue", color = "black") +
 theme_minimal() +  labs(title = "Distribution de la superficie")
Idem pour les 3 autres
grid.arrange(histogram1, histogram2, histogram3, histogram4,ncol = 2)
```

Distribution de certaine variables jugées importantes :




**Ft_19_categ** :

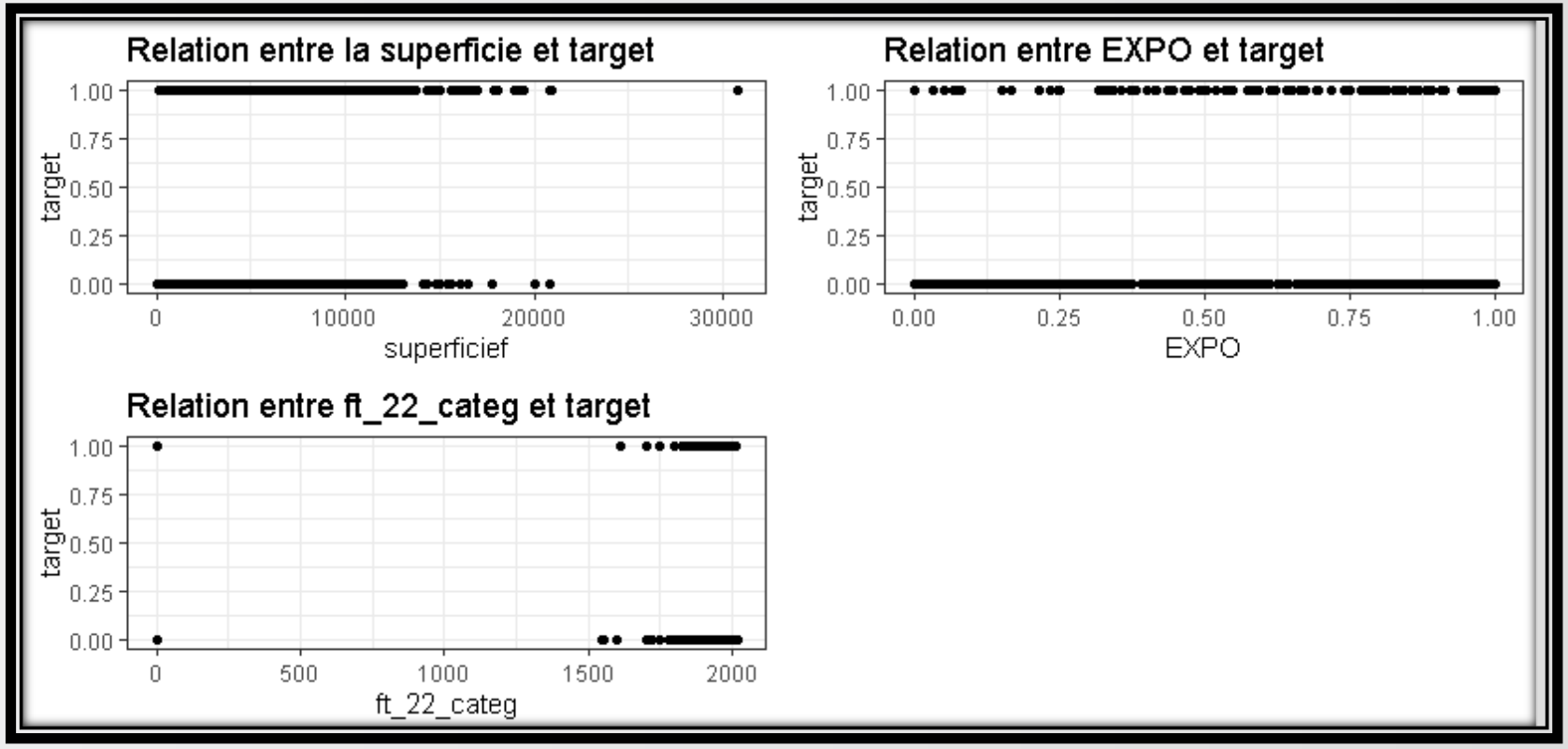
**Ft_4_categ** :

} Fort déséquilibre, → Potentielle biais du modèle

**Criminalité** : asymétrique, mais distribution variée → potentiellement intéressante

# Analyse Bivariée

L'analyse bivariée est cruciale car elle examine la relation entre deux variables à la fois, ce qui permet de comprendre comment elles interagissent entre elles(corrélation, tendance…).



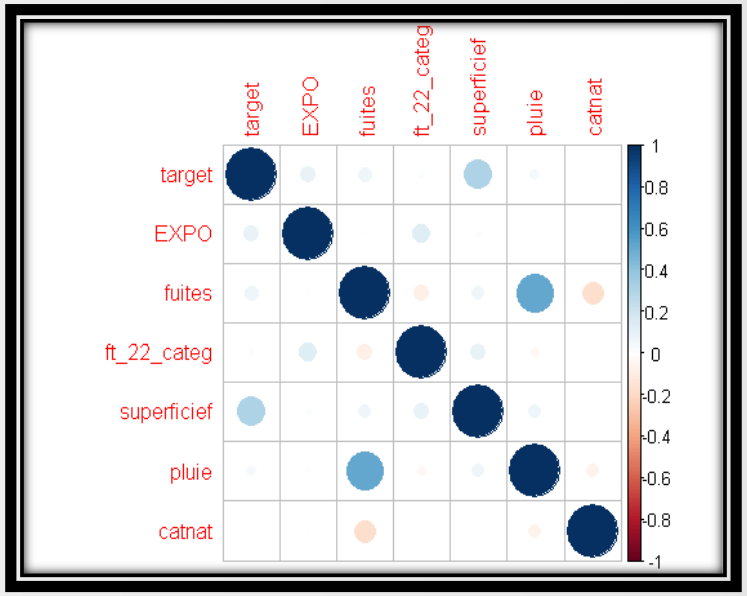

Relation entre la variable cible et les 3 variables jugées importante données dans le sujet :

→ Pas de tendance claire

```
ggplot1 <- ggplot(Train, aes(x = superficief, y = target)) +
geom_point() +  labs(title = "Relation entre la superficie et target")
ggplot2 <- ggplot(Train, aes(x = EXPO, y = target)) +  geom_point()
+  labs(title = "Relation entre EXPO et target")
ggplot3 <- ggplot(Train, aes(x = ft_22_categ, y = target)) +
geom_point() +  labs(title = "Relation entre ft_22_categ et target")

grid.arrange(ggplot1, ggplot2, ggplot3,ncol = 2)
```

# Matrice de corrélation

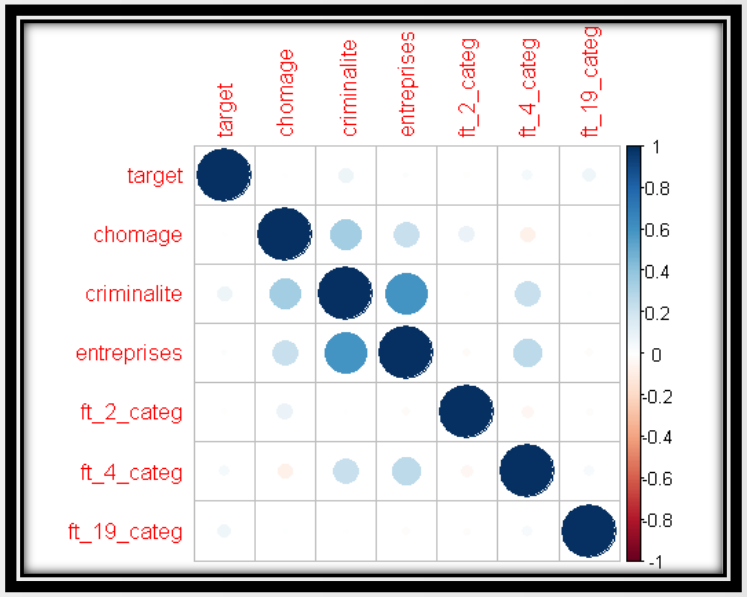



Montre la corrélation entre deux variables
(Ici, on cherche les corrélations avec target)

Faite avec les variables que l'on a jugé importante à première vu

Corrélation entre target et :
- Superficie
- Expo
- Fuites

+ corrélation forte entre pluie et fuites

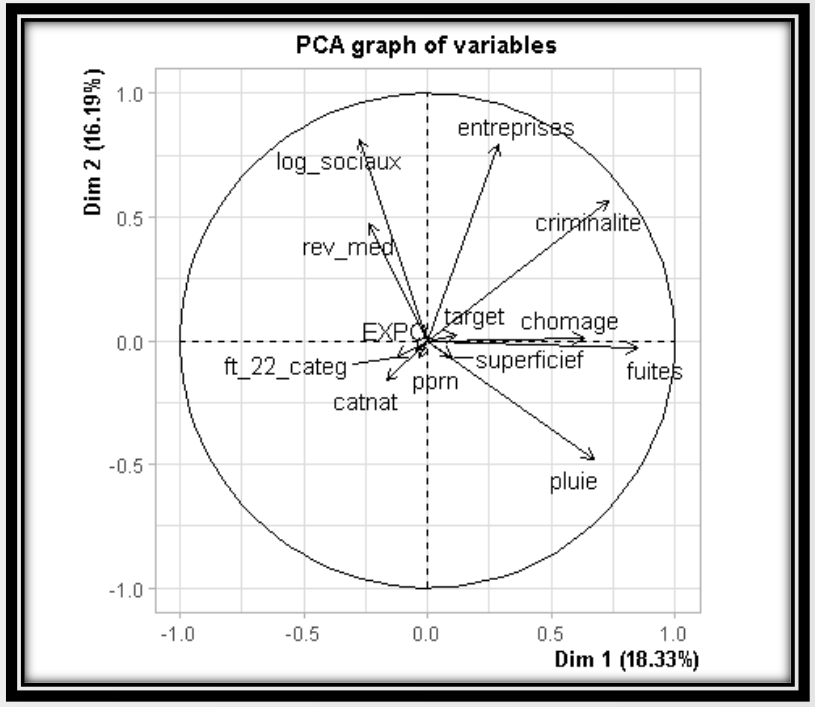



Ici, faites avec des variables jugées moins importante

- Légère corrélation entre target et criminalité
- Forte correlation entre criminalité et entreprises

```
correlation_matrix <- cor(Train[, c("target", "EXPO", "fuites","ft_22_categ", "superficief", "pluie", "catnat")])
corrplot(correlation_matrix, method = "circle")
```

# Analyse en composante principale




Dim 1 : explique 18.33% de la variance totale des données
Dim 2 : Explique 16,19%

L'ACP peut nous aider quant à la sélection des variables pour notre modèle :
Ici, semble importante les variables :

- Criminalité
- Entreprises
- Chômage
- Fuites

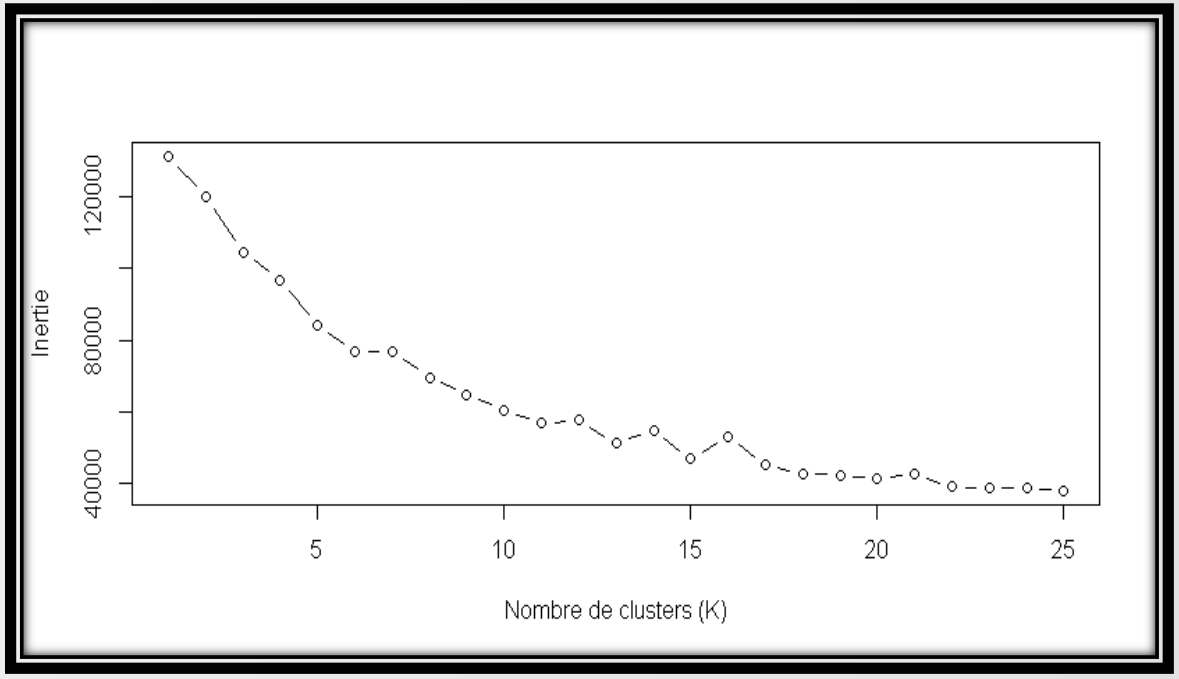
*clu = Train %>% select(target, EXPO , catnat , pluie, fuites, chomage, ft_22_categ, superficief, criminalite, pprn, rev_med, log_sociaux, entreprises)*

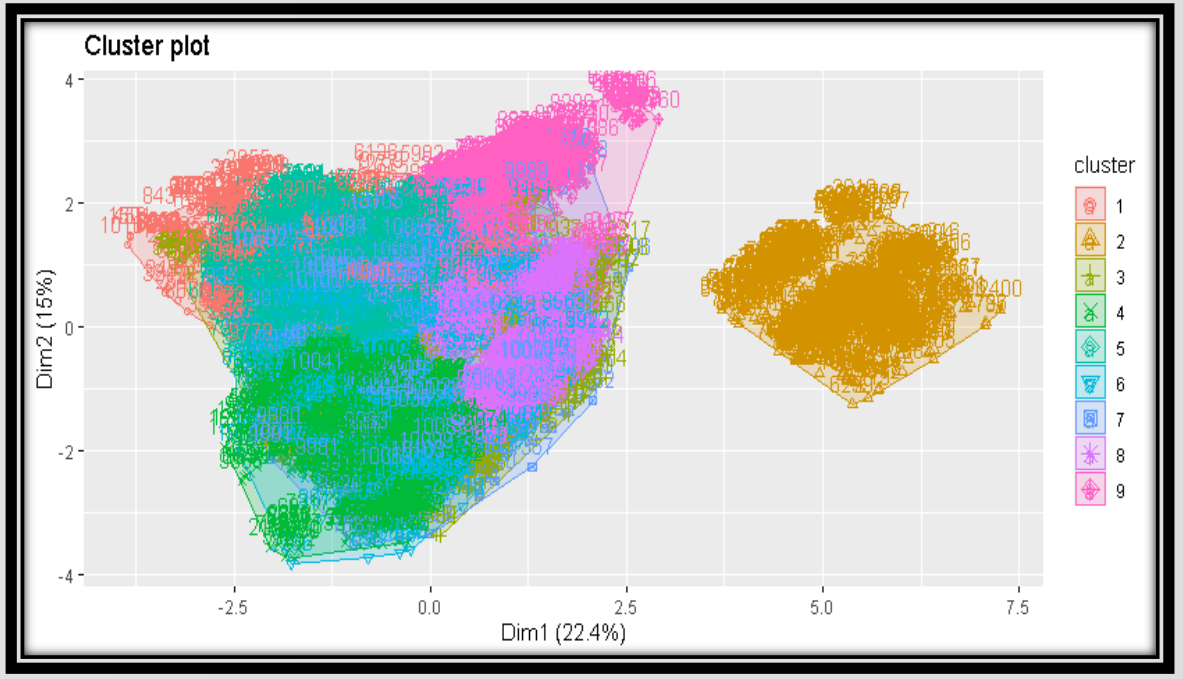
*cluscale <- scale(clu)*
*pca_result2 <- PCA(cluscale, graph = TRUE)*

# Clustering

Méthode du coude pour choisir le nombre de cluster :
→ 9 ou 10 cluster (on choisira 9 ici)

- Difficile d'utiliser cette méthode : base de donnée trop importante
- Cluster 2 isolé
- Cluster plutôt dense dans l'ensemble



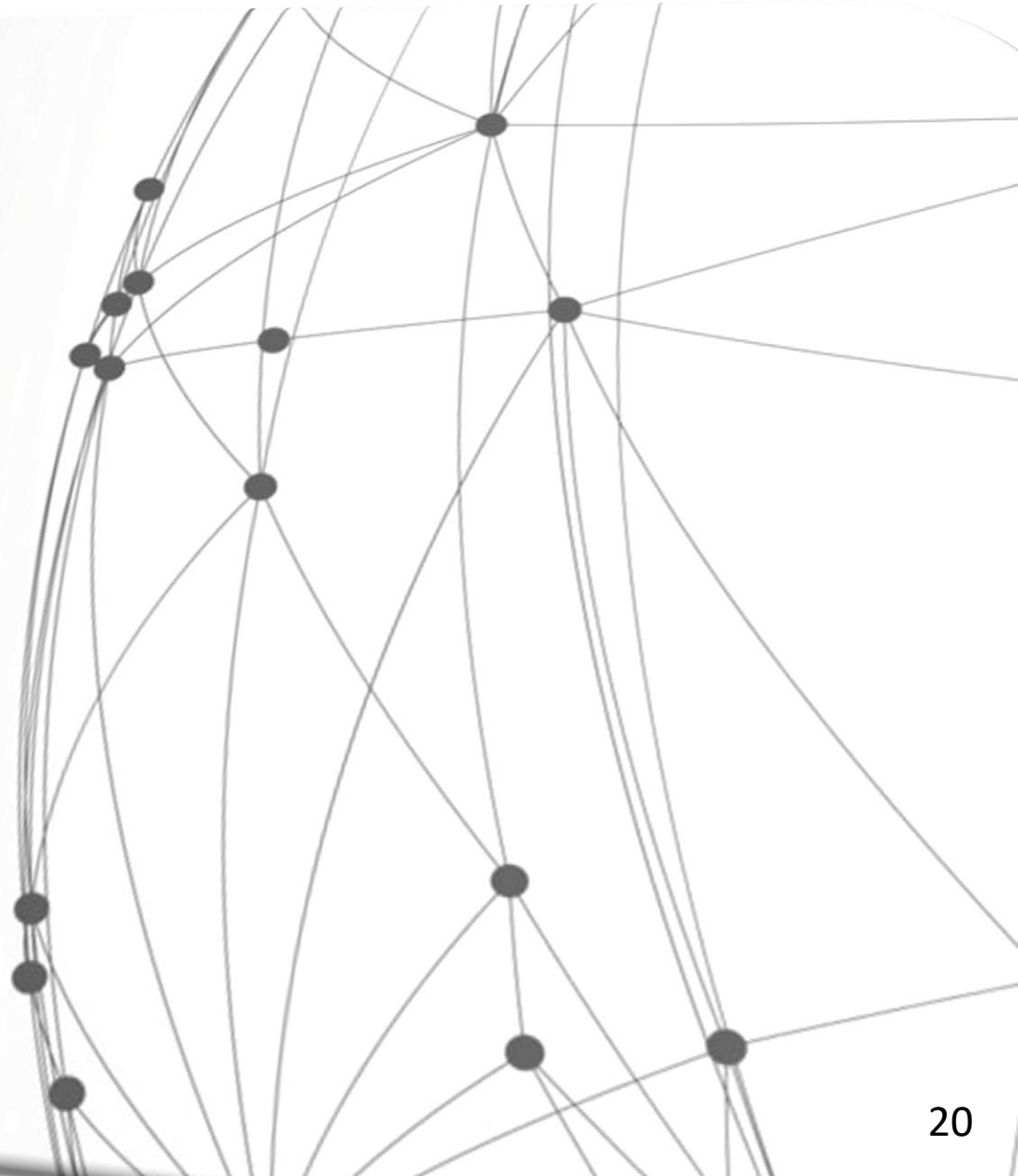




*wss <- numeric(10)*
*for (i in 1:25) { kmeans_result <- kmeans(cluscale, centers = i)*
*wss[i] <- sum(kmeans_result$tot.withinss)}*
*plot (1:25, wss, type = "b", xlab = "Nombre de clusters (K)", ylab = "Inertie")*

*K <- 9kmeans_result <- kmeans(cluscale, centers = K)*
*fviz_cluster(kmeans_result, data = cluscale)*

# Modélisation

- Régression logistique
- XGBOOST
- Ajout données externes



# Régression Logistique

calcule la probabilité qu'une observation appartienne à l'une des deux catégories d'une variable binaire

**Avantages :**

- Point de départ robuste
- Coefficients facile à interpréter

**Inconvénients :**

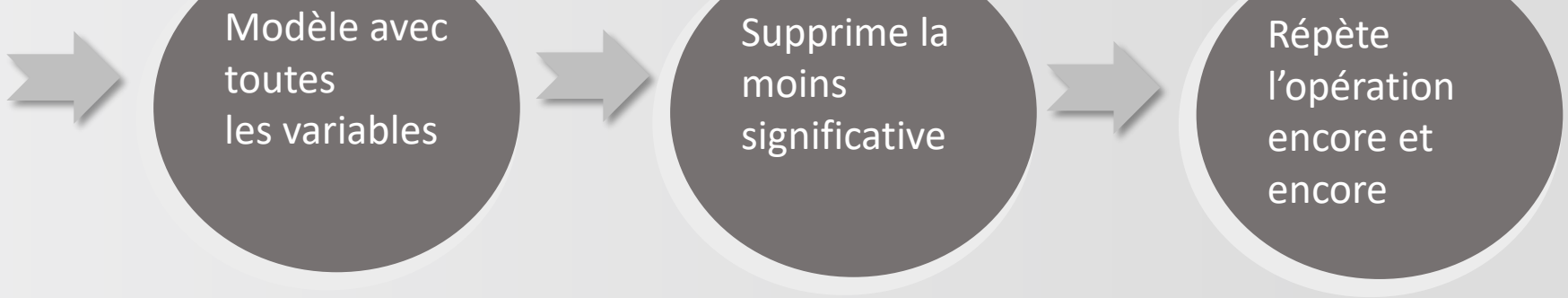
- Suppose une relation logarithmique
- Problème si déséquilibre de classe

Dans notre cas : Pas de relation linéaire claire, problème de déséquilibre (plus de 0 que de 1)
Cependant, c'est un bon point de départ car:

→ Rapide : Nous aide à choisir nos variables
→ Simple : Permet de voir si des modèles plus complexes sont utile, en comparant les résultats

# Création du modèle :

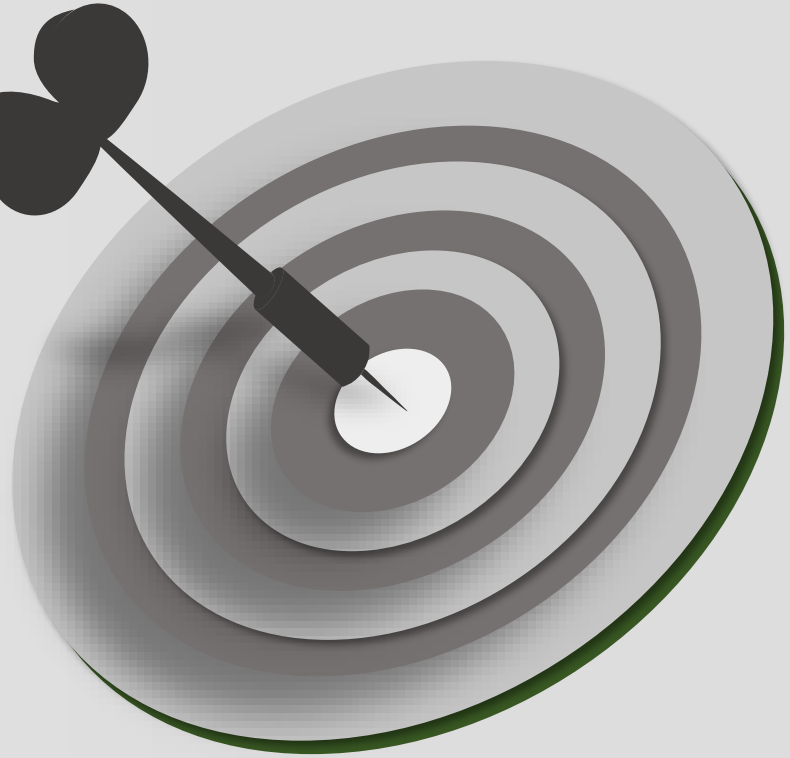
**Méthode Backward**




On finit par n'avoir que des variables significatives

```
model1<-
train(target~EXPO+ft_19_categ+superficief+ft_21_categ+criminalite+rev_med+ft_7_categ.2, data=Train_tout, method="glm", family="binomial", trControl = control)
summary(model1)
```

# Résultat



```
> summary(model1)

Call:
NULL

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.234e+00  3.169e-01 -16.517  < 2e-16 ***
EXPO         1.370e+00  1.423e-01   9.630  < 2e-16 ***
ft_19_categ  5.094e-01  1.317e-01   3.869 0.000109 ***
superficief  2.747e-04  1.216e-05  22.580  < 2e-16 ***
ft_21_categ  1.825e-01  3.246e-02   5.622 1.89e-08 ***
criminalite  6.172e+00  1.575e+00   3.918 8.91e-05 ***
rev_med      1.615e-05  5.733e-06   2.816 0.004857 **
ft_7_categ.2 2.603e-01  8.100e-02   3.214 0.001309 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 10842.7  on 10109  degrees of freedom
Residual deviance:  9795.8  on 10102  degrees of freedom
AIC: 9811.8

Number of Fisher Scoring iterations: 4
```

**Score sur le site challenge: 0,4245**

Sur le site, il est précisé que ft_22_categ a une grande importance dans la précision des modèles, on ne le retrouve pas dans notre modèle de régression: il n’est pas significatif et n’augmente pas notre score.

Les variables significatives semblent logique :
→ superficie, criminalité (destruction des biens),
le revenu médian (maintenance)…
Intuitivement, se sont des variables qui on un lien avec le nombre de sinistre.

# XGBOOST

Définition : optimisation de l'algorithme de boosting de gradient

**Avantages :**

- Performances supérieurs, notamment si relations non linéaires
- Moins enclin au surajustement

**Inconvénients :**

- Coefficients dur à interpréter
- Paramètre difficile à régler

C'est donc un modèle idéal pour nous :

- Relations qui ont l'air non linéaires
- Possible problème de surajustement (→Pas besoin de rééchantillonner)

- Utilisation du package R « caret » pour optimiser les paramètres du modèle
- On réutilise une méthode backward, comme expliquée précédement

# Le package caret sur R

Nous avons fait le choix d'utiliser le package caret sur R pour plusieurs raison:

-Simplification du processus de modélisation
-Caret supporte une variété de techniques de resampling, notamment la validation croisée k-fold
-Tuning des hyperparamètres

## Hyperparamètre XGBOOST

Afin de trouver les meilleurs paramètres pour notre modèle XGB, nous avons utilisé la ligne de code suivante pour la grille de paramètre:

```
xgbGrid <- expand.grid(nrounds = c(100, 200, 300),
                       max_depth = c(3, 4, 6),
                       colsample_bytree = c(0.8, 0.9),
                       eta = c(0.01, 0.05, 0.1),
                       gamma = c(0, 0.1, 0.2),
                       min_child_weight = c(1, 3, 5),
                       subsample = c(0.8, 1))
```

Valeur finale:

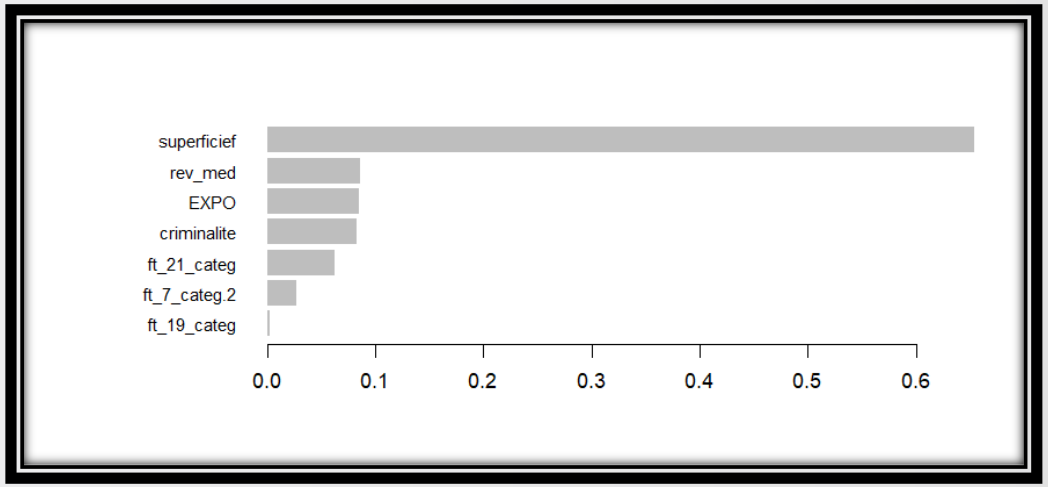
100
3
0,9
0,1
0
1
1

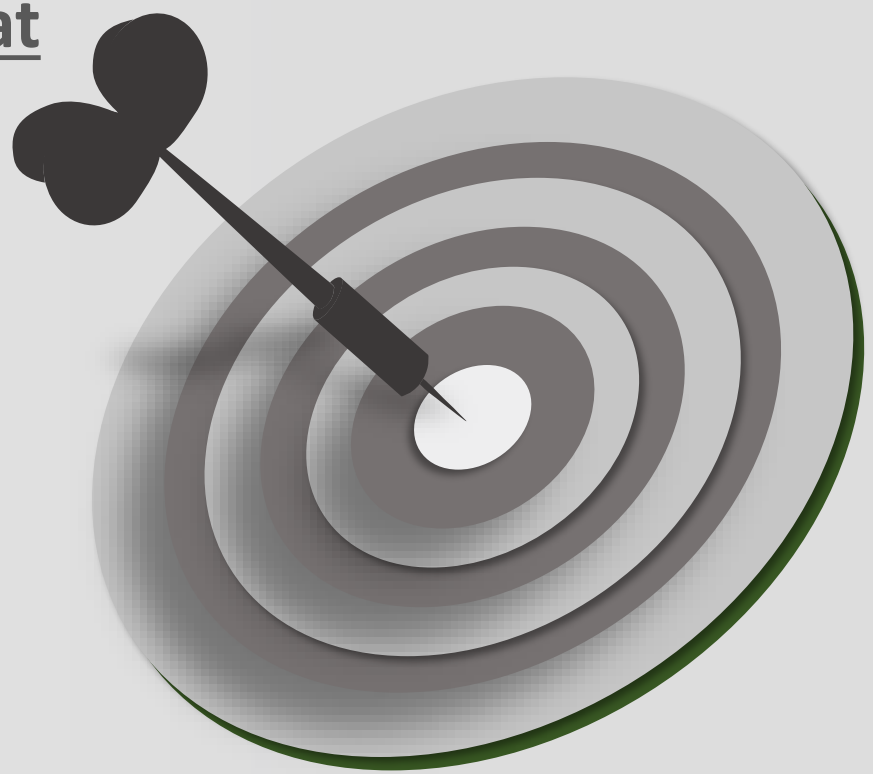
On peut par la suite, sortir les meilleurs paramètres.
**Inconvénient → prends du temps**

**Nrounds**: nombres d'arbre à construire
**Max_depth**: profondeur max de chaque arbre
**Colsample_bytree**: fraction des variables à échantillonner pour chaque arbre, la diminuer peut aider à prevenir le surapprentissage.
**Eta**: taux d'apprentissage, réduit l'impact de chaque arbre.
**Gamma**: seuil de réduction de la perte nécessaire pour effectuer une nouvelle division sur un nœud
**Min_child_weight**: contrôle sur la création de nouveau nœud.
**Subsample**: fraction des échantillons à utiliser pour l'entrainement sur chaque arbre.

# Résultat

En appliquant le modèle XGB à tout notre variables explicatives, on se rends compte que la plupart n'apportent aucune importance au modèle, voici les plus importantes:



On obtient un score de **0,4055**

La encore : Variables utilisées semblent logique (criminalité, superficie…)

# Commentaires sur ces résultats

Résultats illogiques :

Avec nos analyses, le XGBOOST devrait avoir de meilleur résultat , comme expliqué précédemment.

De plus, le nombre de variable inutiles semblent très importante.

**Potentielles erreurs ?**

*(Nous en parlerons plus tard dans le diapo)*

# Données externes

Données proviennent de data.gouv.fr
→ lien

→ Merge grâce au code INSEE, et jointure gauche car certaines données insee sont manquantes

Comme précédemment→ Traitement des données :

- Enlève les espaces dans les noms
- Gestion des NA (moyenne ou mode)

Enormément de données du type:

| CODGEO | Nb Pharmacies et parfumerie | Dynamique Entrepreneuriale | Dynamique Entrepreneuriale Service et Commerce | Synergie Médicale COMMUNE | Orientation Economique | Indice Fiscal Partiel | Score Fiscal | Indice Evasion Client | Score Evasion Client |
|---|---|---|---|---|---|---|---|---|---|

# Création des nouveaux modèles

Nous utiliserons les mêmes modèles que précédemment, à savoir régression logistique et XGBOOST

Au vu du grand nombre de variables, il est nécessaire dans un premier temps de faire un tri de nous-même, quelles variables sont les plus utiles? Ont-elles un lien avec notre variable cible?
Quelles variables sont utilisables, ou non.

On se concentrera plutôt sur des variables type infrastructure, population
Par exemple: score_croissance_population, dynamique demographique…

# Régression logistique

```
> summary(model3)

Call:
NULL

Coefficients:
                                              Estimate Std. Error z value Pr(>|z|)
(Intercept)                                 -4.531e+03  1.907e+04  -0.238 0.812224
superficief                                  2.822e-04  1.333e-05  21.167  < 2e-16 ***
EXPO                                         1.569e+00  1.691e-01   9.281  < 2e-16 ***
ft_21_categ                                  1.397e-01  3.917e-02   3.566 0.000362 ***
Nb_Atifs                                    -7.770e-06  4.257e-06  -1.825 0.067992 .
`FidélitéPop Sédentaire`                    -1.143e-01  1.246e-01  -0.918 0.358816
Moyenne_Revenus_Fiscaux_Départementaux      -1.420e-05  2.083e-05  -0.682 0.495499
Moyenne_Revenus_Fiscaux_Régionaux           -3.786e-05  4.698e-05  -0.806 0.420324
Dep_Moyenne_Salaires_Employé_Horaires        5.914e-01  2.570e-01   2.301 0.021393 *
Reg_Moyenne_Salaires_Horaires               -2.377e-01  2.759e-01  -0.861 0.388984
Reg_Moyenne_Salaires_Prof_Intermédiaire_Horaires  1.500e-01  3.061e-01   0.490 0.624238
Reg_Moyenne_Salaires_Employé_Horaires        1.016e+00  7.458e-01   1.363 0.172938
Dynamique_Entrepreneuriale_Service_et_Commerce  5.597e-05  4.523e-05   1.238 0.215870
Valeur_ajoutée_régionale                    -5.576e-05  2.242e-05  -2.487 0.012889 *
PIB_Régionnal                                7.100e-01  3.000e+00   0.237 0.812950
Score_PIB                                   -3.511e+03  1.484e+04  -0.237 0.812954
Dynamique_Entrepreneuriale                  -2.720e-05  6.418e-05  -0.424 0.671735
ft_19_categ                                  4.806e-01  1.378e-01   3.487 0.000488 ***
ft_7_categ.2                                 2.349e-01  8.659e-02   2.713 0.006676 **
Nb_de_Commerce                               2.230e-04  1.293e-04   1.725 0.084464 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

**Score**: 0,4271

Ce qui est étonnant, c’est que nous avons beaucoup de variables non significatives et pourtant c’est le meilleur score que nous avons obtenus.

Certaines de ces variables n’ont pas l’air d’avoir de lien logique avec le nombre de sinistre.
Il y a un potentiel surajustement du modèle, d’où le meilleur résultat obtenu.

# XGBOOST (et autre)

Le modèle XGBOOST ne nous a donné aucun résultat satisfaisant, de ce fait, nous avons décidé de ne pas l'utiliser pour les données externe.

Nous avons également utilisé d'autres modèles qui n'ont pas surpassé la régression logistique:
random-forest par exemple.

```
# Entraînement du modèle Random Forest avec des poids de classe
rf_model <- randomForest(target ~
superficief+EXPO+criminalite+ft_21_categ+Nb_Atifs+Fidélité+Moyenne_Revenus_Fiscaux_Départementaux+Moyenne_
Revenus_Fiscaux_Régionaux+Dep_Moyenne_Salaires_Employé_Horaires+Reg_Moyenne_Salaires_Horaires+Reg_Moyenne_
Salaires_Prof_Intermédiaire_Horaires+Reg_Moyenne_Salaires_Employé_Horaires+Dynamique_Entrepreneuriale_Serv
ice_et_Commerce+Valeur_ajoutée_régionale+
PIB_Régionnal+Score_PIB+Dynamique_Entrepreneuriale+ft_19_categ+ft_7_categ.2+Nb_de_Commerce,
                        data = Train,
                        trControl = trainControl(method = 'cv', number = 5,))

rf_predict <- predict(rf_model, newdata = X_test)
X_test$target <- rf_predict
Y_testrf <- X_test[, c("X","Identifiant", "target")]
```

# Erreurs potentielles – Problèmes rencontrés

Parce que nous nous sommes lancés dans le projet un peu trop rapidement, nous n'avons pas pris le temps de le cerner complétement.
De ce fait, notre code R peut sembler brouillon, et il n'a pas été facile de se repérer dedans.
Nous avons également perdu du temps sur la modification de nos données Train sans modifier Test. Nous avons bloqué une bonne partie du temps sur ce point.
Nous avons bloqués une bonne partie du temps sur les variables catégorielles.
Nous nous demandons également si nous n'avons pas effectué un mauvais nettoyage des données et/ou un mauvais encodage des variables catégorielles, ou une mauvaise analyse des variables.
Des colonnes se sont dupliqués, du bruit dans les données est apparus, et lorsqu'un problème était réglé, un autre apparaissait. Le tout a pu affecter les performances de nos modèles.

Tout cela peut expliquer pourquoi le XGBOOST ne nous donne pas un meilleur résultat, alors qu'il le devrait.

# Conclusion

Nous sommes satisfait de nos résultats qui nous ont permis de battre le benchmark, et de nous placer relativement haut dans le classement (nous avons été 1$^{er}$ du classement académique et 9$^{ème}$ public pendant la quasi-totalité du projet, mais surpassé les derniers jours).
Cependant, nous sommes déçu du fait que nos résultats ne soient pas totalement cohérent avec les attentes des modèles.

Ce challenge et ce projet étaient vraiment intéressant, nous avons pu utiliser des méthodes nouvelles, nettoyer nos propres données, et vivre une vraie expérience de compétition.
De plus réaliser un challenge nous a permis d'utiliser les connaissances acquises dans les différents cours de façon concrète, et non pas seulement d'apprendre de la théorie sans application.