PROJET DATA SCIENCE

✧✧✧

Compétition de Tarification Auto

GROUPE 11

NGUYEN Dinh Viet

NGUYEN Anh Quang

LAI Phuong Hoa

Contenu

INTRODUCTION

Dans ce projet, notre objectif est de construire un modèle de prédiction de la prime pure pour une assurance automobile. Les données historiques de 60000 polices observées pendant au moins 3 mois de l'année dernière seront utilisées comme données de train (*dataTrain*) pour construire chaque modèle. Ensuite, le modèle de tarification le plus précis sera utilisé pour proposer la prime adéquate pour 20000 nouveaux assurés dans nos données de test (*test*).

Notre prime pure sera structurée comme

$$P_{pure} = P[Y > 0]E[Y|Y > 0, X]$$

où Y est la valeur du sinistre (*claimValue*) pour chaque observation, et X est un vecteur de variable dénotant les autres informations de cette observation. Donc, $P[Y > 0]$ est la probabilité de claim, en autres termes, la probabilité qu'un observation ait 1 sinistre ou plus l'année précédente; et $E[Y|Y > 0, X]$ la valeur attendue du sinistre en sachant qu'elle est positive pour cette observation.

Dans les sections suivantes, nous allons d'abord effectuer une analyse descriptive des données, puis nous créerons des modèles séparés pour la probabilité de sinistre et la valeur attendue du sinistre. Ensuite, ces modèles seront utilisés pour prédire la prime pure pour chaque observation.

TRAITEMENT DE DONNEES

1.1 Description des Variables

Notre projet utilise les packages suivants:

- remotes
- xgboost
- forecast
- caret
- tidyverse
- data.table
- e1071
- statmod
- tweedie
- randomForest
- ranger
- boot
- tree
- glmnet
- ggplot2
- dplyr
- patchwork
- rpart plotly

- ggcorrplot
- pdp
- Metrics
- plyr
- h2o

Tout d'abord, on importe 2 fichiers de données "training" et "testing" selon le code ci-dessous:

```
folder <- "D:/R/"
training <- read.csv(paste(folder,"dataTrain_11.csv",sep=""))
testing <- read.csv(paste(folder,"test.csv",sep=""))
```

Pour la variable âge du conducteur ou conductrice dans la donnée "training", l'interquartile de cette variable est 22. Alos, nous supprimons toutes les observations dont l'âge est supérieur à $52 + 22 \times 1.5 = 85$ ans.

Ensuite, nous créons une nouvelle variable AgeGroup basée sur les quartiles:

```
training <- training %>% filter(age <= 85)

# Creating AgeGroup variable based on quartiles
training <- training %>%
  mutate(AgeGroup = case_when(
    age >= 18 & age < 30 ~ 1,
    age >= 30 & age < 40 ~ 2,
    age >= 40 & age < 52 ~ 3,
    age >= 52 ~ 4
  ))
```

Maintenant, il y a $58,036$ observations dans notre ensemble "training".

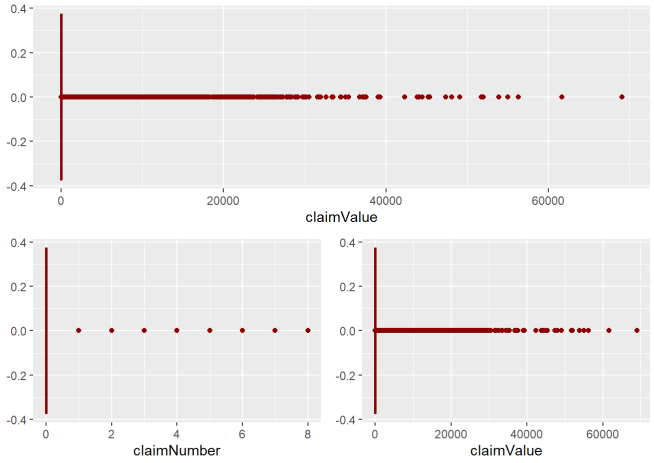
Nous créons une nouvelle variable binaire claimBin, qui vaut 0 si aucune réclamation n'était présente pour l'individu ($claimValue = 0$) et vaut 1 si des réclamations ont été faites ($claimValue > 0$).

En utilisant la fonction "head" pour lire la donnée "training" après avoir supprimé et ajouté des variables, on obtient les résultats suivants :

```
##   id gender carType carCategory    occupation age carGroup bonus carValue
## 1  1   Male       C       Large      Employed  31       11   -30     3595
## 2  2   Male       A       Large      Employed  27       14   -50    17695
## 3  3   Male       A      Medium       Retired  71       13   -30     9820
## 4  4   Male       C       Large     Housewife  26        7   -50    11165
## 5  5   Male       D       Large Self-employed  45       13   -10    25485
## 6  6 Female       D      Medium      Employed  34       10   100    26685
##   material subRegion region cityDensity exposure claimNumber claimValue
## 1        0       Q34      Q   212.42471      365           0          0
## 2        1       L43      L    73.26207      264           0          0
## 3        0       R27      R   250.84133      365           0          0
## 4        1        R6      R   236.39938      365           0          0
## 5        0       P14      P    30.62119      365           0          0
## 6        0      L134      L    49.53391      365           0          0
##   AgeGroup claimBin
## 1        2        0
## 2        1        0
## 3        4        0
## 4        1        0
## 5        3        0
## 6        2        0
```

1.2 Analyses graphiques

1.2.1 Le Graphique Boxplot



D'après le graphique boxplot ci-dessus, on peut voir que, sur toutes les observations, le nombre de polices dont la valeur et le nombre de sinistres sont égaux à 0 est très élevé.

Sur la base de la fonction "quantile" pour la variable claimValue et de la fonction "table" pour la variable claimNumber, nous obtenons les résultats suivants :

```
quantile(training$claimValue)
```

```
##       0%      25%      50%      75%     100%
##     0.00     0.00     0.00     0.00 69068.03
```

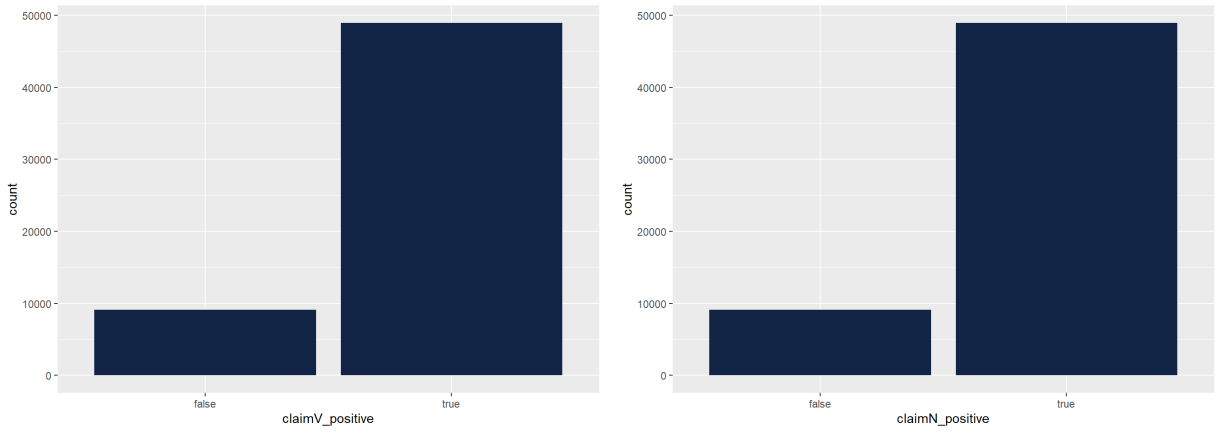
```
table(training$claimNumber)
```

```
##
##     0     1     2     3     4     5     6     7     8
## 48923  7593  1201   247    44    15     9     3     1
```

Jusqu'à 75% des policies ont reçu une valeur de réclamation de 0 et il y avait 48, 923 polices sur un total de 58, 036 polices observées qui n'ont reçu aucune réclamation.

1.2.2 L'histogramme

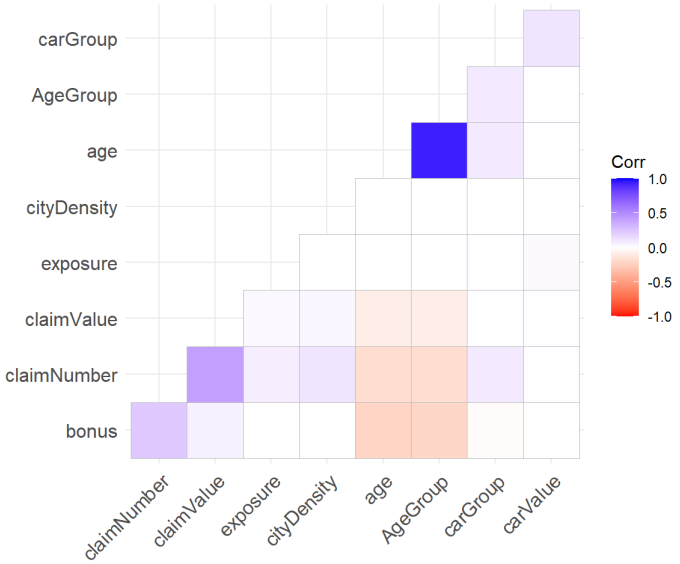
L'histogramme ci-dessous montre que la valeur et le nombre de polices qui doivent être réclamées ne sont pas beaucoup par rapport au nombre qui n'a pas à être réclamé.



1.3 Choix des Variables

1.3.1 Corrélation

Le graphique ci-dessous montre la corrélation entre les variables quantitatives dans les données



1.3.2 Anova

On utilise Anova pour estimer comment la variable dépendante quantitative change avec le degré d'une ou plusieurs variables indépendantes catégorielles. Ici, claimValue et claimBin sont deux variables que nous considérons.

Pour la Variable claimValue:

```
##                Df    Sum Sq   Mean Sq   F value   Pr(>F)
## id              1 1.326e+07 1.326e+07     4.474  0.03441 *
## gender          1 1.253e+07 1.253e+07     4.228  0.03976 *
## carType         5 5.973e+07 1.195e+07     4.030  0.00117 **
## carCategory     2 2.420e+08 1.210e+08    40.827  < 2e-16 ***
## occupation      4 5.179e+08 1.295e+08    43.678  < 2e-16 ***
## age             1 9.855e+08 9.855e+08   332.454  < 2e-16 ***
## carGroup        1 9.126e+07 9.126e+07    30.786 2.89e-08 ***
## bonus           1 3.987e+08 3.987e+08   134.487  < 2e-16 ***
## carValue        1 5.875e+06 5.875e+06     1.982  0.15918
## material        1 9.227e+07 9.227e+07    31.129 2.43e-08 ***
## subRegion     470 2.364e+09 5.031e+06     1.697  < 2e-16 ***
## exposure        1 1.650e+08 1.650e+08    55.658 8.74e-14 ***
## claimNumber     1 3.259e+10 3.259e+10 10995.080  < 2e-16 ***
## AgeGroup        1 2.737e+03 2.737e+03     0.001  0.97576
## claimBin        1 5.836e+08 5.836e+08   196.872  < 2e-16 ***
## Residuals   57543 1.706e+11 2.964e+06
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

À partir des résultats ci-dessus, nous procédons à la suppression de la variable carValue qui est la variable avec une P-value supérieure à 0,05.

Pour la Variable claimBin:

De même, on supprime "carValue".

Bien que AgeGroup montre un manque de significatif ici, la variable age le fait ; nous devons donc le conserver dans notre modèle, car en réalité, l'âge a un impact significatif sur le montant de la réclamation.

```
##                Df Sum Sq Mean Sq   F value Pr(>F)
## id              1      0       0 6.314e+00  0.012 *
## gender          1     10      10 3.716e+02 <2e-16 ***
## carType         5      6       1 4.438e+01 <2e-16 ***
## carCategory     2     31      16 5.973e+02 <2e-16 ***
## occupation      4    108      27 1.033e+03 <2e-16 ***
## age             1    125     125 4.789e+03 <2e-16 ***
## carGroup        1     75      75 2.872e+03 <2e-16 ***
## bonus           1    257     257 9.841e+03 <2e-16 ***
## carValue        1      0       0 4.350e-01  0.510
## material        1      6       6 2.264e+02 <2e-16 ***
## subRegion     470    214       0 1.743e+01 <2e-16 ***
## exposure        1     33      33 1.280e+03 <2e-16 ***
## claimNumber     1   5304    5304 2.033e+05 <2e-16 ***
## claimValue      1      5       5 1.969e+02 <2e-16 ***
## AgeGroup        1      0       0 3.000e-03  0.960
## Residuals   57543   1501       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1.3.3 Régression au Lasso avec Variable claimBin pour la Sélection des Fonctionnalités

Considérons les variables de réponse (claimBin) et variables prédictives selon le code ci-dessous:

```
# Response Variables: claimBin
claimBin <- as.numeric(training$claimBin) - 1

# Predictor Variables:
predVar <- data.matrix(training[, c("gender", "carType", "carCategory", "occupation", "carGroup", "bonus", "material", "regi
on", "cityDensity", "AgeGroup")])
```

On effectue une validation croisée k-fold pour trouver la valeur λ optimale avec $\alpha = 1$ signifie régression au Lasso.

```
# Perform k-fold cross-validation to find optimal lambda value:
cv_claim <- cv.glmnet(predVar, claimBin, alpha = 1)
  # alpha = 1 signifies Lasso Regression
best_lambda_claim <- cv_claim$lambda.min
best_lambda_claim
```

On obtient $\lambda = 0.000224635$

Ensuite, on trouve les coefficients du meilleur modèle selon le code ci-dessous:

```
# Find coefficients of best model:
  # Claim Value:
best_model_claim <- glmnet(predVar, claimBin, alpha = 1, lambda = best_lambda_claim)
coef(best_model_claim)
```

Sur la base des résultats reçus, nous ne supprimons plus de variables.

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                          s0
## (Intercept)  0.0778629068
## gender       0.0359626228
## carType      0.0060699853
## carCategory  0.0026610532
## occupation   0.0015355984
## carGroup     0.0076430873
## bonus        0.0014093448
## material    -0.0190558076
## region      -0.0016190190
## cityDensity  0.0004724241
## AgeGroup    -0.0464280352
```

1.3.4 Les Variables carType, carCategory et carGroup

En utilisant le test du chi carré de Pearson, à partir des résultats obtenus ci-dessous, nous pouvons voir qu'il existe une relation significative entre le groupe de voitures et la catégorie, nous supprimons donc l'un d'entre eux pour éviter la multicolinéarité. Ici, nous avons choisi de supprimer carCategory.

```
chisq.test(training$carType,training$carGroup)
```

```
##
##  Pearson's Chi-squared test
##
## data:  training$carType and training$carGroup
## X-squared = 73.35, df = 95, p-value = 0.9515
```

```
chisq.test(training$carCategory,training$carGroup)
```

```
##
##  Pearson's Chi-squared test
##
## data:  training$carCategory and training$carGroup
## X-squared = 112.7, df = 38, p-value = 2.511e-09
```

```
chisq.test(training$carType,training$carCategory)
```

```
##
##  Pearson's Chi-squared test
##
## data:  training$carType and training$carCategory
## X-squared = 6.1969, df = 10, p-value = 0.7985
```

Nous supprimons également Bonus car il n'est généralement pas pertinent lorsque notre ensemble de données est gonflé à zéro. Nous convertissons également la variable "occupation" en "employ situation" ("Self-employed" et " Employed" seront "Working", l'autre sera "Jobless"). Parce que "housewife", "retired" and "unemployed" sont susceptibles d'utiliser moins la voiture que les personnes ayant un emploi. Par conséquent, ils ont moins d'impact sur le nombre de réclamations et la valeur des réclamations. Ainsi, en les regroupant en un seul groupe, nous pouvons diminuer la complexité du modèle En conclusion, nous avons choisi les variables suivantes pour le modèle : "gender", "carType", "Employ situation", "AgeGroup", "material", "region", "cityDensity"

1.3.5 Séparation Supplémentaire des Données Train

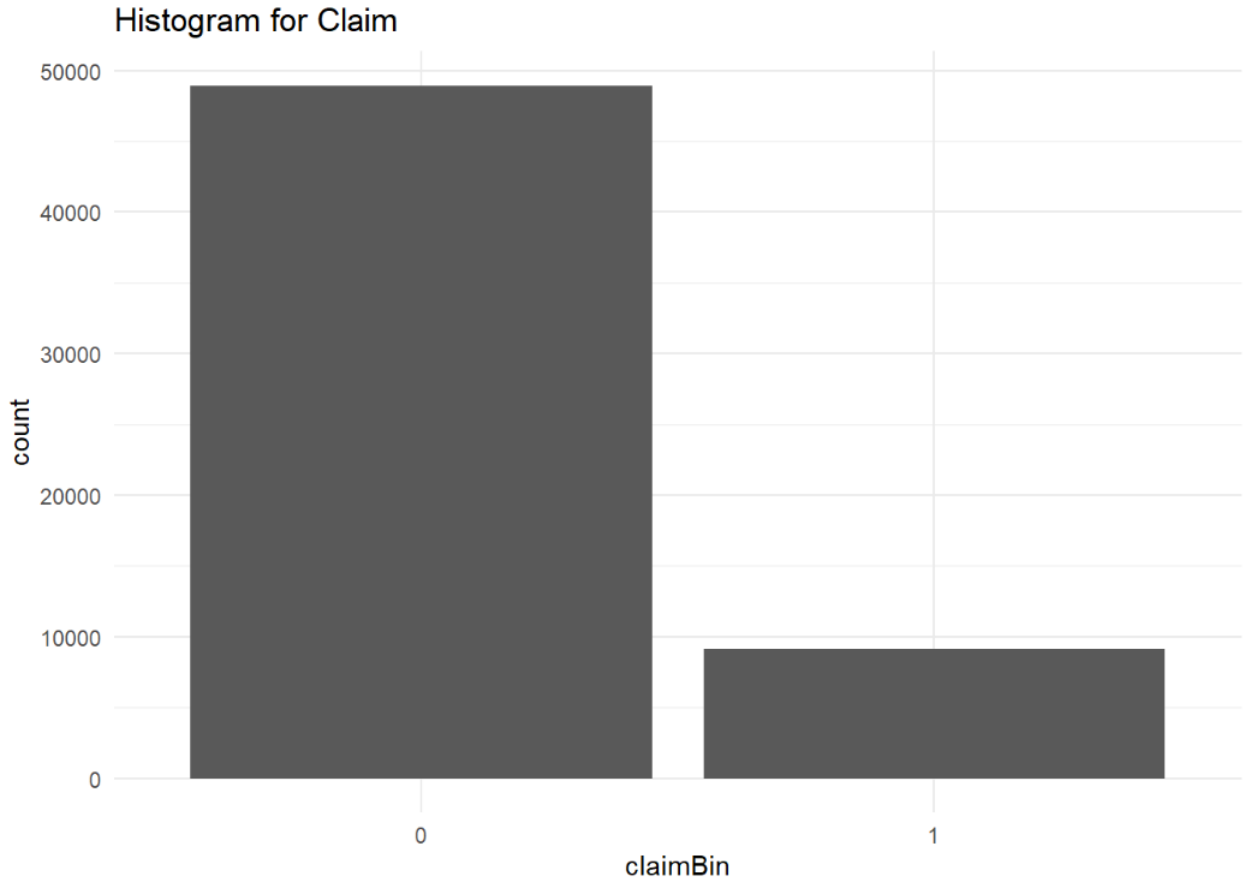
Nous diviserons ensuite les données Train en deux parties : 80% seront utilisées pour l'entraînement des modèles et 20% seront utilisées pour la validation.

MODELISATION

2.1 Probabilité de Sinistre

2.1.1 Analyse Descriptive du Nombre de Sinistre

Premièrement, nous allons faire une simple analyse descriptive de la variable *claimBin*. Rappelons qu'ici, 0 représente que l'observation n'a pas de sinistre ($claimNumber = 0$), et 1 représente que l'observation a un sinistre ($claimNumber > 0$). Nous allons montrer les données sous la forme d'un histogramme.



En observant le graphique, nous pouvons voir que le nombre de personnes sans sinistre est environ 7 fois plus élevé que celui des personnes avec sinistre, avec respectivement 48934 et 9102 observations.

Nous pouvons également calculer la proportion d'échantillon de sinistres comme ci-dessous.

$$p = \frac{\text{Nombre d'observations avec sinistre}}{\text{Nombre d'observations}} \tag{1}$$

Le résultat est

```
sampleProp <- sum(as.numeric(data_test_final$claimBin) - 1)/nrow(data_test_final)
sampleProp
```

```
## [1] 0.1550354
```

Avec cette brève introduction à la variable de réponse, nous pouvons commencer à entrer dans notre processus de modélisation.

Nous allons utiliser 5 modèles pour prédire la probabilité CART, Random Forest et XGBoost. Nous utiliserons ensuite le paramètre de précision des matrices de confusion et l'AUC ROC individuelle comme échelle de sélection des modèles. Le processus général consistera à construire chaque modèle à l'aide des données de train, puis à prédire la probabilité de sinistre sur les données de validation.

2.1.2 Modèle logistique (Logit)

Le modèle logistique (ou modèle logit) est un GLM classique permettant de prédire la probabilité qu'un événement se produise en faisant en sorte que sa probabilité soit une combinaison linéaire d'autres variables indépendantes.

Voici le résultat

```
logit.fit <- glm(claimFormula, data = data_train_final, family = binomial(link = "logit"))
summary(logit.fit)
```

```
##
## Call:
## glm(formula = claimFormula, family = binomial(link = "logit"),
##     data = data_train_final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1705  -0.6320  -0.4956  -0.3738   2.5633
##
```

```
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -1.3429145  0.0593667 -22.621  < 2e-16 ***
## genderMale               0.2867108  0.0278098  10.310  < 2e-16 ***
## carTypeB                 0.0838360  0.0377804   2.219 0.026485 *
## carTypeC                 0.0924192  0.0435626   2.122 0.033877 *
## carTypeD                 0.1135912  0.0390217   2.911 0.003603 **
## carTypeE                 0.1888923  0.0455617   4.146 3.39e-05 ***
## carTypeF                 0.2748423  0.0583068   4.714 2.43e-06 ***
## Employ_situationWorking -0.0984115  0.0267044  -3.685 0.000229 ***
## material                -0.1224025  0.0269336  -4.545 5.50e-06 ***
## regionM                  0.1098562  0.0653542   1.681 0.092776 .
## regionN                  0.1860055  0.0629230   2.956 0.003116 **
## regionO                  0.0408078  0.0722498   0.565 0.572199
## regionP                  0.0220216  0.0724373   0.304 0.761121
## regionQ                 -0.2166121  0.0572564  -3.783 0.000155 ***
## regionR                 -0.4489922  0.0925910  -4.849 1.24e-06 ***
## regionS                  0.0448938  0.0734725   0.611 0.541181
## regionT                  0.0995988  0.0712766   1.397 0.162306
## regionU                 -0.1375547  0.0676210  -2.034 0.041931 *
## cityDensity              0.0054124  0.0004416  12.257  < 2e-16 ***
## AgeGroup                -0.4641030  0.0129213 -35.918  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 40320  on 46328  degrees of freedom
## Residual deviance: 38133  on 46309  degrees of freedom
## AIC: 38173
##
## Number of Fisher Scoring iterations: 5
```

En examinant les coefficients, nous pouvons constater que la plupart de nos variables indépendantes sont significatives (p = 0,05), à l'exception de certaines des variables *region*. Ceci est logique puisque si ces régions ont une population plus faible que les autres, elles seront moins significatives dans le modèle final.

Ensuite, comme

$$\chi^2 \sim \text{Variance nulle} - \text{Variance résiduelle} \tag{2}$$

Nous pouvons effectuer le test d'adéquation du Khi-deux. La p-value qui en résulte est de $0,000 < 0,05$, ce qui signifie que le modèle est très utile pour prédire la probabilité de sinistre.

Après, le critère d'information d'Akaike (AIC) est également un mesure de la qualité d'un modèle, calculé comme

$$AIC = 2k - 2\log(L) \tag{3}$$

où k est le nombre de paramètres à estimer du modèle et L est le maximum de la fonction de vraisemblance. Le modèle avec l'AIC le plus bas offre le meilleur prédiction. Pour le modèle logit, $AIC = 38173$

Finalement, nous construisons la matrice de confusion pour notre modèle. Pour chaque modèle, nous utilisons la proportion de l'échantillon calculée dans la section ci-dessus comme seuil entre les deux événements "Avoir un sinistre" et "Pas de sinistre". En d'autres termes, si la probabilité de sinistre prédite par le modèle pour une observation est inférieure à la proportion de l'échantillon, cette observation est classée comme "Sans sinistre".

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5932  681
##          1 3960 1134
##
##                Accuracy : 0.6036
##                  95% CI : (0.5946, 0.6124)
##     No Information Rate : 0.845
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1292
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5997
##             Specificity : 0.6248
##          Pos Pred Value : 0.8970
##          Neg Pred Value : 0.2226
##              Prevalence : 0.8450
##          Detection Rate : 0.5067
##    Detection Prevalence : 0.5649
##       Balanced Accuracy : 0.6122
##
##        'Positive' Class : 0
##
```

On notons "Sans sinistre" comme 0 et "Avoir un sinistre" comme 1 dans la matrice. Dans ce modèle, la précision est 0,6036.

2.1.3 Modèle Probit

Similaire au modèle Logit, le modèle Probit est également un GLM mais avec une fonction de lien probit au lieu d'une fonction de lien logit.

```
probit.fit <- glm(claimFormula, data = data_train_final, family = binomial(link = "probit"))
summary(probit.fit)
```

```
##
## Call:
## glm(formula = claimFormula, family = binomial(link = "probit"),
##     data = data_train_final)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1207  -0.6371  -0.5012  -0.3698   2.6090
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -0.8190410  0.0328878 -24.904  < 2e-16 ***
## genderMale               0.1504772  0.0152278   9.882  < 2e-16 ***
## carTypeB                 0.0466778  0.0207370   2.251 0.024389 *
## carTypeC                 0.0486848  0.0239631   2.032 0.042188 *
## carTypeD                 0.0611087  0.0214747   2.846 0.004433 **
## carTypeE                 0.1035350  0.0252862   4.095 4.23e-05 ***
## carTypeF                 0.1491742  0.0326527   4.569 4.91e-06 ***
## Employ_situationWorking -0.0491766  0.0147676  -3.330 0.000868 ***
## material                -0.0670586  0.0148098  -4.528 5.95e-06 ***
## regionM                  0.0598652  0.0365195   1.639 0.101157
## regionN                  0.0993010  0.0349121   2.844 0.004451 **
## regionO                  0.0213320  0.0384673   0.555 0.579203
## regionP                  0.0151380  0.0385102   0.393 0.694252
## regionQ                 -0.1251275  0.0313686  -3.989 6.64e-05 ***
## regionR                 -0.2537274  0.0515014  -4.927 8.37e-07 ***
## regionS                  0.0247838  0.0390900   0.634 0.526067
## regionT                  0.0513314  0.0381526   1.345 0.178488
## regionU                 -0.0755241  0.0365118  -2.068 0.038594 *
## cityDensity              0.0030105  0.0002463  12.221  < 2e-16 ***
## AgeGroup                -0.2497143  0.0069943 -35.703  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 40320  on 46328  degrees of freedom
## Residual deviance: 38168  on 46309  degrees of freedom
## AIC: 38208
##
## Number of Fisher Scoring iterations: 4
```

Pour ce modèle, comme le modèle logit, la plupart de variables indépendantes sont significatives (p = 0,05).

Le p-value du test d'adéquation du Khi-deux est $0,000 < 0,05$, donc le modèle est utile pour prédire. Nous remarquons également que $AIC = 38208$.

Voici la matrice de confusion

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 5805  658
##          1 4087 1157
##
##                Accuracy : 0.5947
##                  95% CI : (0.5857, 0.6036)
##     No Information Rate : 0.845
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1266
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.5868
##             Specificity : 0.6375
##          Pos Pred Value : 0.8982
##          Neg Pred Value : 0.2206
##              Prevalence : 0.8450
##          Detection Rate : 0.4959
##    Detection Prevalence : 0.5521
##       Balanced Accuracy : 0.6122
##
##        'Positive' Class : 0
##
```

La précision est 0,5947 pour le modèle probit.

2.1.4 CART

Classification And Regression Tree (CART) est une variante de l'algorithme de l'arbre de décision. Il explique comment les valeurs de la variable cible peuvent être prédites en fonction d'autres éléments. Nous utilisons le package *rpart* sur R pour cette section. La matrice de confusion donne une précision de 0,7343.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7805 1023
##          1 2087  792
##
##                Accuracy : 0.7343
##                  95% CI : (0.7262, 0.7423)
##     No Information Rate : 0.845
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.1819
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.7890
##             Specificity : 0.4364
##          Pos Pred Value : 0.8841
##          Neg Pred Value : 0.2751
##              Prevalence : 0.8450
##          Detection Rate : 0.6667
##    Detection Prevalence : 0.7541
##       Balanced Accuracy : 0.6127
##
##        'Positive' Class : 0
##
```

2.1.5 Random Forest

Le Random Forest, comme son nom l'indique, se compose d'un grand nombre d'arbres de décisions individuels qui fonctionnent comme un ensemble. Chaque arbre individuel d'un Random Forest émet une prédiction de classe et la classe ayant reçu le plus de votes devient la prédiction de notre modèle. Le resultat en utilisant le package *randomForest* sur R est

```
##
## Call:
##  randomForest(formula = claimFormula, data = data_train_final,      importance = TRUE, mode = "classification")
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 15.71%
## Confusion matrix:
##       0  1  class.error
## 0 39038  4 0.0001024538
## 1  7272 15 0.9979415397
```

Nous pouvons encore une fois créer une matrice de confusion, qui donne une précision de 0.8403.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 9705 1683
##          1  187  132
##
##                Accuracy : 0.8403
##                  95% CI : (0.8335, 0.8469)
##     No Information Rate : 0.845
##     P-Value [Acc > NIR] : 0.9214
##
##                   Kappa : 0.0811
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.98110
##             Specificity : 0.07273
##          Pos Pred Value : 0.85221
##          Neg Pred Value : 0.41379
##              Prevalence : 0.84496
##          Detection Rate : 0.82899
##    Detection Prevalence : 0.97275
##       Balanced Accuracy : 0.52691
##
##        'Positive' Class : 0
##
```

2.1.6 XGBoost

XGBoost est un algorithme assez récent qui gagne en popularité ces dernières années en raison de sa précision en matière de prédiction. Cependant, dans le cadre de ce projet, nous ne détaillerons pas le fonctionnement de cet algorithme, ni le processus de tuning des hyper-paramètres pour cette section particulière. Cependant, à titre d'exemple de l'algorithme, nous allons tout de même montrer notre résultat en utilisant le package *xgBoost*, en utilisant $max.depth = 20$ et $nround = 2000$. La matrice de confusion résultante est ci-dessous, avec une précision de 0,7238.

Avec tous les modèles construits, nous pouvons maintenant passer à la phase de sélection. Nous avons gardé ce processus assez simple, car nous irons plus en profondeur avec la validation croisée k-fold et le tuning des hyper-paramètres pour la prochaine section du projet, les modèles pour la valeur espérée de la sinistre. Cependant, notez que le résultat pourrait potentiellement être plus précis si nous appliquons ces méthodes pour les modèles de probabilité de sinistre dans cette section.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 7954 1296
##          1 1938  519
##
##                Accuracy : 0.7238
##                  95% CI : (0.7156, 0.7318)
##     No Information Rate : 0.845
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0787
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.8041
##             Specificity : 0.2860
##          Pos Pred Value : 0.8599
##          Neg Pred Value : 0.2112
##              Prevalence : 0.8450
##          Detection Rate : 0.6794
##    Detection Prevalence : 0.7901
##       Balanced Accuracy : 0.5450
##
##        'Positive' Class : 0
##
```

2.1.7 Selection du Modèle

Avant notre processus de sélection, nous calculons également l'AUC ROC (La plaine sous le courbe ROC), en raison du déséquilibre de nos données (le nombre d'observations sans sinistre est beaucoup plus élevé que celui des observations avec sinistre). Ensuite, nous pouvons tabuler la précision et l'AUC ROC pour chaque modèle.

```
##     Model Accuracy ROCAUC
## 1   Logit   0.6036 0.6630
## 2  Probit   0.5947 0.6628
## 3    CART   0.7343 0.6186
## 4      RF   0.8403 0.6050
## 5 XGBoost   0.7238 0.5737
```

Ce résultat est logique puisque, même si les GLM (Logit et Probit) sont d'excellents modèles en théorie, ils ont des difficultés à surpasser les trois autres algorithmes complexes. Random Forest est composé de nombreux arbres de décision, son résultat devrait donc être meilleur que celui d'un simple modèle CART. Bien que le résultat de XGBoost soit médiocre, il est compréhensible car il s'agit simplement d'une démonstration de cet algorithme sans traitement

approprié pour obtenir un meilleur résultat.

Avec cela, nous pouvons voir que même si Random Forest produit un résultat ROC AUC légèrement inférieur, sa précision est bien meilleure que celle des autres modèles. Par conséquent, nous allons choisir Random Forest comme modèle de prédiction pour notre probabilité de sinistres.

2.2 La valeur attendue de sinistre en sachant qu'elle est positive

Avant d'aller construire le modèle, nous devons filtrer les données qui ont le claimValue = 0. Nous diviserons à nouveau les données Train en deux parties :80% seront utilisées pour l'entraînement des modèles et 20% seront utilisées pour la validation.

2.2.1 The Generalized Linear Model (Gamma)

Nous allons commencer par comparer le modèle Le Modèle Linéaire Généralisé (GLM) pour la distribution Gamma. Le GLM généralise la régression linéaire en permettant au modèle linéaire d'être relié à la variable réponse via une fonction lien. GLM est simple,commun en assurance et il est facile à interpréter. Cependent, il a des hypothèses strictes autour de la forme de la distribution et du caractère aléatoire des termes d'erreur et il a un faible pouvoir prédictif. Voici le résultat

```
# Model
#Premier modele lognormal
modele_gamma<-glm(formula = data_train_final2$claimValue ~ gender+
              carType+Employ_situation+AgeGroup+material+region+cityDensity
            ,data=data_train_final2,  family = Gamma(link="log"))
summary(modele_gamma)
```

```
##
## Call:
## glm(formula = data_train_final2$claimValue ~ gender + carType +
##     Employ_situation + AgeGroup + material + region + cityDensity,
##     family = Gamma(link = "log"), data = data_train_final2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -5.1695  -1.5014  -0.8690  -0.1164   7.4341
```

```
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                7.8235567  0.1129750  69.250  < 2e-16 ***
## genderMale                -0.0796038  0.0526400  -1.512  0.13052
## carTypeB                   0.1059311  0.0722931   1.465  0.14288
## carTypeC                   0.1203397  0.0818170   1.471  0.14138
## carTypeD                   0.1277359  0.0737956   1.731  0.08350 .
## carTypeE                   0.1647146  0.0857437   1.921  0.05477 .
## carTypeF                   0.0117467  0.1072885   0.109  0.91282
## Employ_situationWorking -0.0645130  0.0496785  -1.299  0.19412
## AgeGroup                  -0.0741224  0.0237840  -3.116  0.00184 **
## material1                 -0.1420362  0.0513949  -2.764  0.00573 **
## regionM                   -0.0147859  0.1238051  -0.119  0.90494
## regionN                    0.0891010  0.1195183   0.746  0.45599
## regionO                    0.0286005  0.1363754   0.210  0.83389
## regionP                   -0.1914566  0.1368395  -1.399  0.16182
## regionQ                   -0.1112315  0.1088269  -1.022  0.30677
## regionR                    0.0148145  0.1735129   0.085  0.93196
## regionS                    0.0750982  0.1427812   0.526  0.59893
## regionT                   -0.0357792  0.1382612  -0.259  0.79581
## regionU                   -0.1005755  0.1278818  -0.786  0.43162
## cityDensity                0.0001874  0.0008211   0.228  0.81944
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Gamma family taken to be 4.429914)
##
##     Null deviance: 16626  on 7301  degrees of freedom
## Residual deviance: 16439  on 7282  degrees of freedom
## AIC: 123765
##
## Number of Fisher Scoring iterations: 8
```

En examinant les coefficients, on constate que seuls "Groupe d'âge" et "Matériel" sont significatifs (p = 0,05). C'est logique puisque le comportement au volant est différent à chaque âge. Les jeunes ont tendance à conduire de façon imprudente, ce qui augmente souvent la valeur du sinistre. Et les voitures avec un matériau de couverture subissent moins de dommages en cas d'accident. On calcule aussi La racine de l'erreur quadratique moyenne (REQM) ou en anglais, root-mean-square error ou RMSE pour mesurer la puissance de prédiction.

```
predictions_gamma <- predict(modele_gamma, data_test_final2)
rmse_gamma <- sqrt(mean((predictions_gamma  - data_test_final2$claimValue)^2))
rmse_gamma
```

```
## [1] 4969.359
```

Le RMSE du modèle est élevé e qui signifie que la puissance predi du test n'est pas l'efficacité

2.2.2 Xgboost

XGBoost est une bibliothèque de renforcement de gradient distribuée optimisée conçue pour être très efficace, flexible et portable. Il implémente des algorithmes d'apprentissage automatique dans le cadre du Gradient Boosting. XGBoost a besoin que toutes les entrées soient des nombres, nous devrons convertir les données que nous utilisons en nombre. Nous utiliserons "one hot encoding" pour convertir.One Hot encoding est le processus de conversion d'une variable catégorielle avec plusieurs catégories en plusieurs variables, chacune avec une valeur de 1 ou 0. Si nous ultilisons Label Code, xgboost interprétera à tort la caractéristique comme ayant une relation numérique. Voici la result de l'One Hot encoding:

Filter

	genderFemale	genderMale	carTypeA	carTypeB	carTypeC	carTypeD	carTypeE	carTypeF	Employ_s
1	0	1	0	0	1	0	0	0	
2	1	0	0	0	1	0	0	0	
3	1	0	0	1	0	0	0	0	
4	0	1	1	0	0	0	0	0	
5	0	1	0	0	0	1	0	0	
6	1	0	0	0	0	0	0	1	
7	1	0	0	0	0	0	1	0	
8	0	1	0	0	0	1	0	0	
9	0	1	0	0	0	1	0	0	
10	0	1	0	1	0	0	0	0	
11	1	0	0	1	0	0	0	0	
12	1	0	1	0	0	0	0	0	
13	0	1	0	1	0	0	0	0	
14	1	0	0	1	0	0	0	0	
15	0	1	0	0	0	1	0	0	
16	1	0	1	0	0	0	0	0	
17	1	0	0	1	0	0	0	0	
18	0	1	1	0	0	0	0	0	
19	1	0	0	0	0	1	0	0	

Ensuite, nous exécutons un simple XGBoost

```
xgb.model1 <- xgb.train(data = dbTrain_xgb,
                        nround = 100,
                        max_depth = 6,
                        eta = 0.3,
                        gamma = 0,
                        min_child_weight = 1,
                        subsample = 1,
                        objective = "reg:squarederror"
)

xgb.pred1 <- predict(xgb.model1,dbTest_xgb)
RMSE(xgb.pred1,test_data_xgb$claimValue)
```

Nous obtenons la resultats:

```
## [1] 5035.723
```

Le RMSE est élevé. Nous essayons maintenant de faire une Grid-Search pour essayer de nombreuses combinaisons différentes d'hyper-paramètres pour voir quelle combinaison nous donne le meilleur ajustement (best fit)

```
grid_tune <- expand.grid(
  nrounds = c(100,200,500), #number of trees
  max_depth = c(2,4,6),
  eta = c(0.1,0.2,0.3), #c(0.025,0.05,0.1,0.3), #Learning rate
  gamma = 0,
  colsample_bytree = c(0.4),
  min_child_weight = c(1,2,3), # c(1,2,3) # the larger, the more conservative the model
  #is; can be used as a stop
  subsample = c(0.5, 0.75, 1.0)
)

train_control <- trainControl(method = "cv",
                              number=5,
                              verboseIter = TRUE,
                              allowParallel = TRUE)

xgb_tune <- train(x = train_data_xgb[,-ncol(train_data_xgb)],
                  y = train_data_xgb[,ncol(train_data_xgb)],
                  trControl = train_control,
                  tuneGrid = grid_tune,
                  method= "xgbTree",
                  objective = "reg:squarederror",
                  verbose = TRUE)
```

Nous avons constaté que cette combinaison d'hyper-paramètres nous donnera le meilleur ajustement (best fit): $max\,depth = 2$ $eta = 0.1$ $gamma = 0$ $min\,child\,weight = 3$ $colsample\,bytree = 0.4$ $subsample = 1$

Nous recommençons ensuite avec la meilleure combinaison de paramètres

```
xgb.model.best.tune <- xgb.train(data = dbTrain_xgb,
                      label = dbY,
                      nround = 100,
                      max_depth = 2,
                      eta = 0.1,
                      gamma = 0,
                      min_child_weight = 3,
                      colsample_bytree = 0.4,
                      subsample = 1,
                      objective = "reg:squarederror"
)
```

```
xgb.pred2 <- predict(xgb.model.best.tune,dbTest_xgb)
rmse_xgb_tuned = caret::RMSE(test_data_xgb$claimValue, xgb.pred2)
importance.matrix <- xgb.importance(feature_names = xgb.model.best.tune$feature_names, model =xgb.model.best.t
xgb.plot.importance(importance_matrix = importance.matrix)
```

Nous obitons la resultats:

```
rmse_xgb_tuned
```

```
## [1] 4484.422
```

Nous pouvons voir que l'efficacité du modèle a augmenté de manière significative.

2.2.3 RandomForest

Tout d'abord, nous allons exécuter un modèle Random Forest simple. Voici la resultal. Voici la résultat

```
set.seed(1234)
library(Metrics)

rf.default<-randomForest(formula = claimValue~.,
                         data=train.rf,
                         method = "rf"
)
rf.default
```

```
##
## Call:
##  randomForest(formula = claimValue ~ ., data = train.rf, method = "rf")
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##           Mean of squared residuals: 19870535
##                     % Var explained: -3.55
```

```
rf.default.pre1 <- predict(rf.default,test.rf)
rmse_rf_default<-rmse(rf.default.pre1,test.rf$claimValue)
rmse_rf_default
```

```
## [1] 4529.062
```

Le RMSE de ce modèle est encore élevé, nous allons donc améliorer le modèle en utilisant Grid Search.

```
y <- "claimValue"
x <- setdiff(names(train.rf), y)

# hyperparameter grid
hyper_grid.h2o <- list(
  ntrees      = seq(100, 300, by = 100),
  mtries      = seq(3, 6, by = 1),
  sample_rate = c(.55, .632, 0.75),
  max_depth = seq(10, 30, by = 10),
  min_rows = seq(1, 3, by = 1),
  nbins = seq(20, 30, by = 10)
)

# build grid search
grid <- h2o.grid(
  algorithm = "randomForest",
  grid_id = "rf_grid",
  x = x,
  y = y,
  seed = 1234,
  nfolds =5,
  training_frame = train.h2o,
  hyper_params = hyper_grid.h2o,
  search_criteria = list(strategy = "Cartesian")
)
```

Nous trouvons la meilleure combinaison d'hyperparamètres: $max\ depth = 10$, $min\ rows = 3$, $nbin = 20$, $ntrees = 100$, $mtries = 3$, $sample\ rate = 0.55$ Après cela, nous exécutons à nouveau le modèle avec une combinaison améliorée d'hyperparamètres. Voici lanouveau modèle et la resultal

```
#best_model1 <- h2o.getModel(grid_perf@model_ids[[1]])
rf.better<-randomForest(formula = claimValue~.,
                        max_depth = 10,
                        min_rows = 3,
                        nbin = 20,
                        ntrees      = 100,
                        mtries      = 3,
                        sample_rate     = 0.55,
                        data=train.rf,
                        method = "rf"
)

rf.pred2 <- predict(rf.better,test.rf)
rf.better.pre2 <- predict(rf.better,test.rf)
rsme_rf_tuned<-rmse(rf.better.pre2,test.rf$claimValue)
rsme_rf_tuned
```

```
## [1] 4536.27
```

Nous pouvons voir que l'efficacité du modèle a augmenté de manière significative.

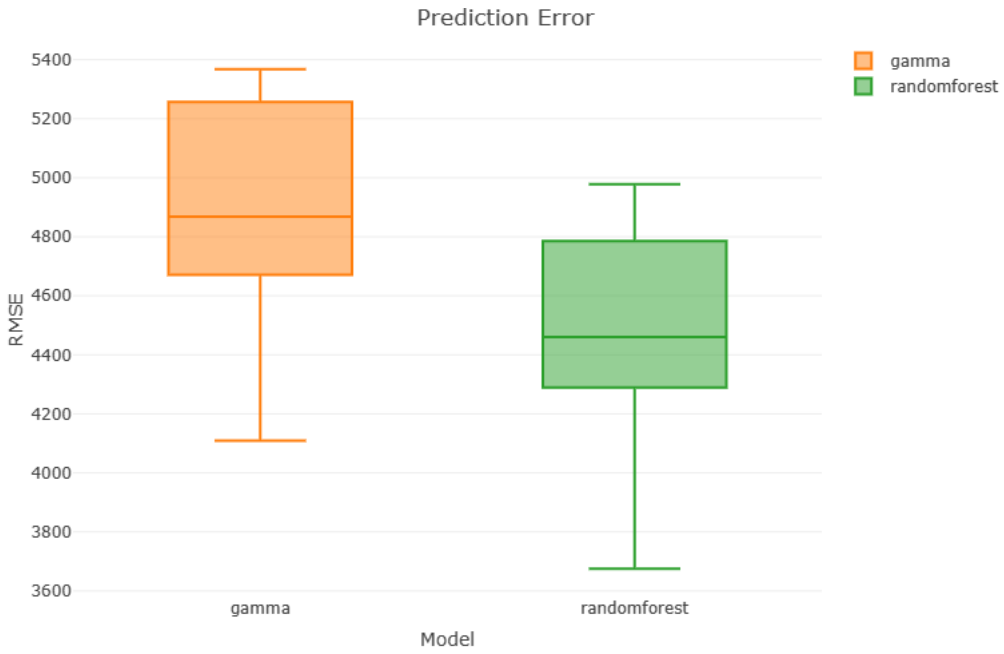
2.2.4 Selection du Modèle

Afin d'évaluer les performances du modèle, nous effectuons Cross-validation avec les mêmes folds pour tous les modèles et comperons les RMSE de chaque modèle. Dans un premier temps, nous allons comparer GLM et RandomForest

```
# Generate some test data
k <- 10 #the number of folds
folds <- cvFolds(NROW(data_rf), K=k)
error_glm<-c()
error_rf<-c()

for(i in 1:k){
  train <- data_rf[folds$subsets[folds$which != i], ] #Set the training set
  validation <- data_rf[folds$subsets[folds$which == i], ] #Set the validation set
  newrf <- randomForest(data=train,
                        claimValue~.,
                        max_depth = 20,
                        min_rows = 3,
                        nbin = 20,
                        ntrees      = 100,
                        mtries      = 3,
                        sample_rate     = 0.55,
                        method = "rf"
)
  newglm_gamma<-glm(formula = claimValue ~ gender+
              carType+Employ_situation+AgeGroup+material+region+cityDensity
            ,data=train,  family = Gamma(link="log"))
#Get the predicitons for the validation set (from the model just fit on the train data)
    res_glm <- sqrt(mean((predict(newglm_gamma,newdata=validation)-validation$claimValue)^2))
    res_rf <- sqrt(mean((predict(newrf,newdata=validation)-validation$claimValue)^2))
    #print(error_glm)
    error_glm<-c(error_glm,res_glm)
    error_rf<-c( error_rf,res_rf)
}
```

Voici le results.



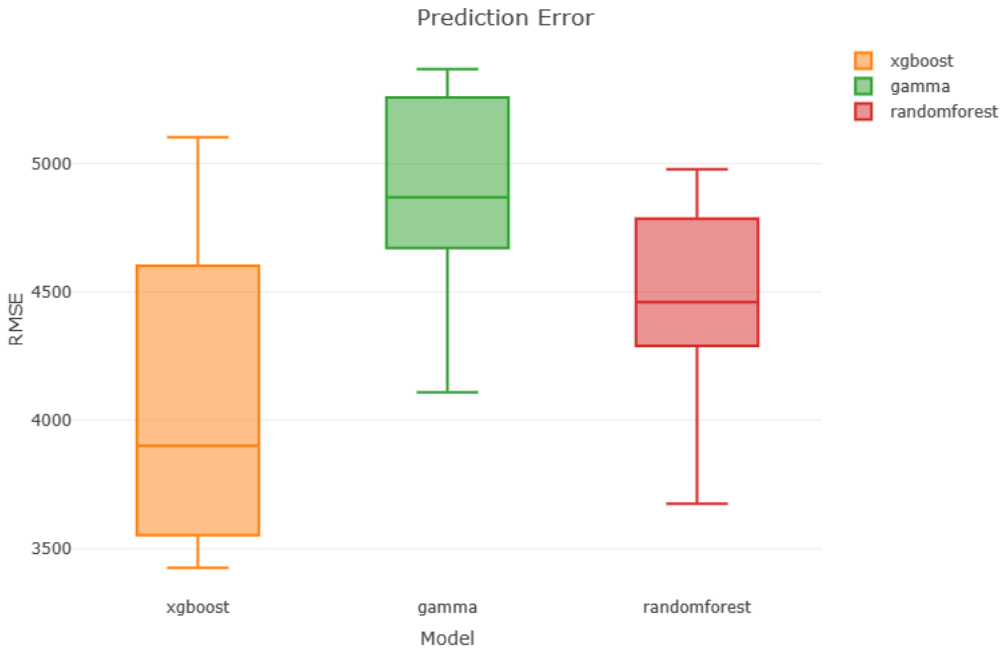
Nous voyons que le modèle Random Forest est plus performant que GLM

Ensuite, nous allons faire validation croisée pour Xgboost. Étant donné que xgboost nécessite une conversion catégorique. Nous ferons une validation croisée séparée. Puisque les données sont assez grandes. Le résultat de cette validation croisée peut encore être valide.

```
nfold <- 10
folds <- createFolds(y =data.ohe1[,25],k=10)
error_xgb <- lapply(1:10, function(kk){
 ## Data management
    # Train
    train <- data.ohe1[which(1:nrow(data.ohe1) %in% folds[[kk]]),]
    dbX <- as(as.matrix(train[,-ncol(train)]),"dgCMatrix")
    dbY <- as.numeric(train$claimValue)
    dbTrain <- xgb.DMatrix(data = dbX, label = dbY)
 # Test
    test <- data.ohe1[which(1:nrow(data.ohe1) %in% folds[[kk]]),]
    dbX <- as(as.matrix(test[,-ncol(test)]),"dgCMatrix")
    dbY <- as.numeric(test$claimValue)
    dbTest <- xgb.DMatrix(data = dbX, label = dbY)
    xgb <- xgb.train(nround = 100,
                     max_depth = 2,
                     eta = 0.1,
                     gamma = 0,
                     min_child_weight = 3,
                     colsample_bytree = 0.4,
                     subsample = 0.5,
                     objective = "reg:squarederror",
                     verbose = FALSE, data = dbTrain
                    )

  res_xgb <- sqrt(mean((predict(xgb,dbTest)-dbY)^2))
  return(c(res_xgb))
 })
 error_xgb <- as.data.frame(do.call("rbind",error_xgb))
```

Nous comparons les résultats entre 3 modèles.



D'après la figure, nous pouvons voir que les performances de xgboost sont les meilleures des 3 modèles au sens du RMSE. Donc, nous utiliserons xgboost pour l'estimation de La valeur attendue de sinistre en sachant qu'elle est positive.

TARIFICATION

Nous appliquons notre modèle pour calculer le nombre premier pur. Le modèle prendra le data.test en entrée et renverra la valeur principale de chaque individu. Nous utiliserons RandomForest pour estimer la Probabilité de Sinistre et xgboost pour estimer Cout de Sinistre. C'est les résultats

id	gender	carType	arCategor	occupation	age	carGroup	bonus	carValue	material	subRegion	region	cityDensity	P(Y>0)	E[Y|X,Y>0]	Prime pure
1	Male	C	Medium	Employed	38	16	-30	47095	1	Q18	Q	211.8309	0.014	1472.624	20.61674
2	Female	C	Medium	Unemploy	20	8	-10	8720	1	R5	R	258.9237	0.2	2784.527	556.9054
3	Female	B	Small	Housewife	21	8	0	8145	0	Q23	Q	205.4308	0.052	2527.335	131.4214
4	Male	A	Large	Housewife	45	13	-50	19790	0	R20	R	290.1327	0.001	1602.478	1.602478
5	Male	D	Small	Self-emplc	49	10	-50	18265	1	R14	R	275.0923	0.004	911.4429	3.645772
6	Female	F	Small	Employed	34	8	-50	5475	1	R29	R	291.3195	0.108	1739.027	187.8149
7	Female	E	Large	Retired	69	19	-40	18250	0	P3	P	34.88186	0.042	1958.81	82.27003
8	Male	D	Small	Employed	48	14	130	4690	1	P11	P	28.38649	0.002	1414.88	2.82976
9	Male	D	Large	Self-emplc	49	11	-50	1460	1	L116	L	100.3049	0.018	1435.892	25.84606
10	Male	B	Large	Retired	48	9	-50	24950	1	U3	U	94.65752	0.01	1535.563	15.35563
11	Female	B	Large	Unemploy	40	11	20	8260	0	L83	L	69.7282	0.016	2173.944	34.78311
12	Female	A	Small	Employed	35	15	110	14695	0	M22	M	153.4523	0.076	1702.98	129.4265
13	Male	B	Large	Retired	49	7	-30	21380	0	T27	T	17.999	0.058	1186.787	68.83367
14	Female	B	Large	Self-emplc	55	10	-30	24325	0	L35	L	61.69064	0.02	2141.247	42.82494
15	Male	D	Large	Retired	63	17	-50	28155	0	P15	P	25.63802	0.04	1379.282	55.1713
16	Female	A	Medium	Housewife	24	4	-10	9870	1	L86	L	50.62578	0.01	2206.158	22.06158
17	Female	B	Small	Unemploy	27	3	-20	17340	0	L93	L	63.89431	0.012	2642.235	31.70682
18	Male	A	Small	Employed	26	4	130	12015	0	L21	L	67.84791	0.006	1944.936	11.66962

RÉFÉRENCES

[1] Zach, Statology (2021). How to Interpret glm output in R.

https://www.statology.org/interpret-glm-output-in-r/

[2] Wikipedia. Courbe ROC.

https://fr.wikipedia.org/wiki/Courbe$_R OC$

[3] Chapman Hall Book (2014), The R Series, Computational Actuarial Science with R

[4] Machine Learning Mastery,

https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

[5] Esbjörn Ohlsson and Börn Johansson (2010), R code for Chapter 2 of Non-Life Insurance Pricing with GLM

[6] What is One Hot Encoding? Why And When do you have to use it?

https://medium.com/hackernoon/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f