LDAT2310 - Project

Lucien Ledune

novembre 30, 2018,10:29

Contents

Introduction

Le but de ce projet est d'appliquer les différentes méthodes vues au cours LDAT2310 afin de prédire au mieux les Claim Frequencies des clients présents dans notre dataset.

Ainsi nous pourrons par exemple appliquer des modèles tels que le GLM, GBM ou encore les regression trees à nos données et déterminer quel modèle sera le plus adéquat pour une utilisation future de celui-ci dans l'estimation des Claim Frequencies des prochains clients de l'entreprise.

Le risque d'accident n'est pas le même selon le conducteur, il peut dépendre de différents facteurs (rating factors). En déterminant la Claim Frequency d'un client il sera plus aisé de proposer une prime d'assurance adéquate à chaque client.

Dataset

```
#Change path to the dataset location
pathTrain = 'C:/Users/Lucien/Downloads/DBtrain.csv'
pathTest = 'C:/Users/Lucien/Downloads/DBtest.csv'
dataTrain = read.csv(pathTrain)
dataTest = read.csv(pathTest)
```

Avant toute chose, concentrons nous sur le dataset mis à notre disposition. Il s'agit d'un dataset d'assurances moto référencant différentes informations sur des conducteurs ayant contracté une assurance dans la même compagnie.

Les différentes variables présentent dans le jeu de données sont :

- Gender : 1 = Homme, 2 = Femme
- DriverAge : Age du conducteur
- CarAge : Age du véhicule
- Area : Variable représentant les différentes régions possibles (1) suburban, 2) urban, 3) countryside low altitude, 4) countryside high altitude (mountain regions))
- Leasing : 1 = Oui, 2 = Non
- Power : Variable catégorielle représantant la puissance du véhicule (Horsepower)
- Split : Fréquence de paiement de l'assurance (1 = Mois, 2 = Trimestre, 3 = Année)
- Contract : Type de couverture du contrat (1 = basique, 2 = intermédiaire, 3 = complète)
- Exposure : Durée du contrat en années
- NbClaims : Nombre de sinistres sur la durée du contrat

Les variables Exposure et NbClaims sont les variables expliquées que nous allons tenter d'estimer au mieux à l'aide de différents modèles au cours de ce projet.

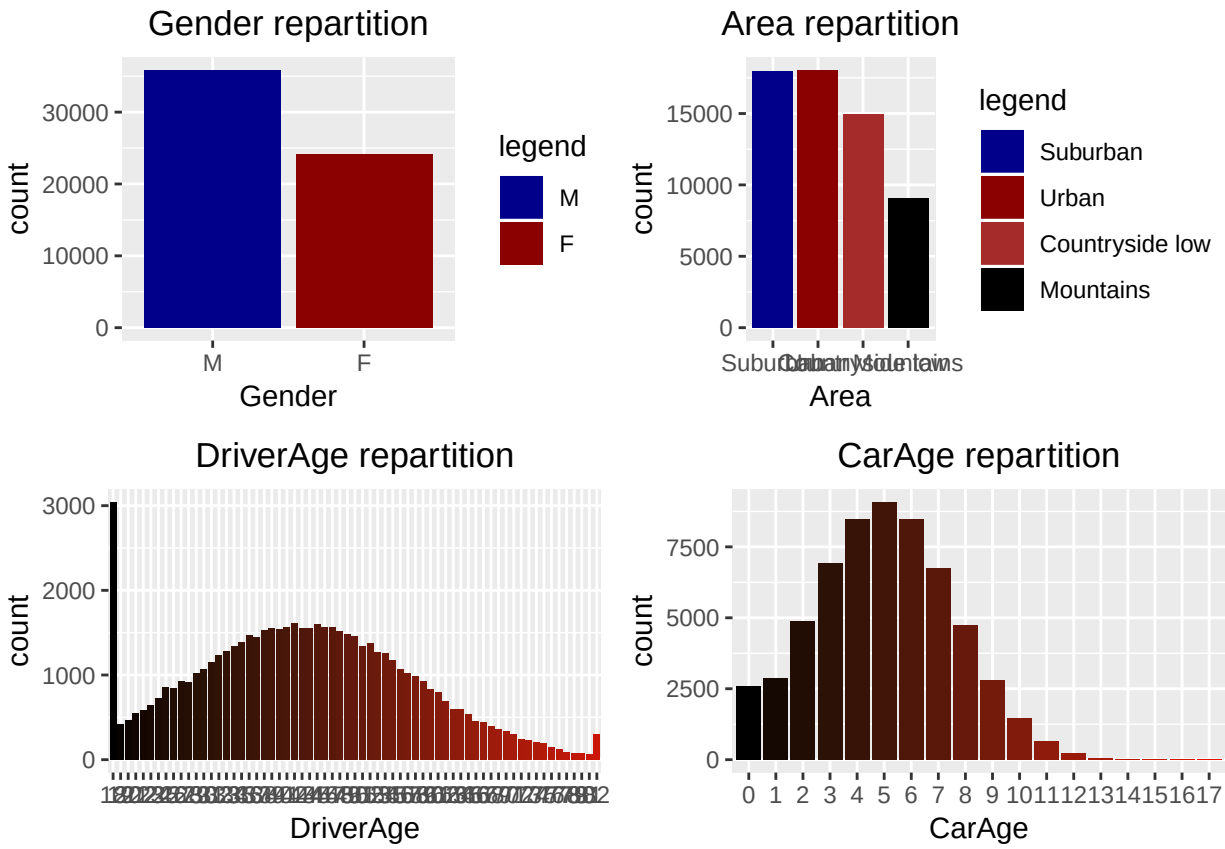
Pour l'entrainement des différents modèles, le Dataset a été divisé en deux parties, le Train set et le Test set (75%/25%).

Nous pouvons maintenant nous intéresser aux différentes variables d'un peu plus près.

```
library(ggplot2)
library(gridExtra)

#Convert as factors for ggplot
for(i in 1:9){
  dataTrain[,i] = as.factor(dataTrain[,i])
}
dataTrain$Nbclaims = as.factor(dataTrain$Nbclaims)

levels(dataTrain$Gender) = c("M", "F")
levels(dataTrain$Area) = c("Suburban", "Urban", "Countryside low", "Mountains")

cc <- scales::seq_gradient_pal("black", "red", "Lab")(seq(0,1,length.out=80))
cc2 <- scales::seq_gradient_pal("black", "red", "Lab")(seq(0,1,length.out=20))


ggGender = ggplot(data=dataTrain, aes(x=Gender, fill = Gender)) + geom_bar(stat="count") +
  labs(title = "Gender repartition") + theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("M" = "darkblue", "F" = "darkred", "3" = "black"))

ggDriverAge = ggplot(data=dataTrain, aes(x=DriverAge, fill = DriverAge)) +
  geom_bar(stat="count") + labs(title = "DriverAge repartition") +
  theme(plot.title = element_text(hjust = 0.5)) + guides(fill=FALSE) + scale_fill_manual(values = cc)

ggCarAge = ggplot(data=dataTrain, aes(x=CarAge, fill = CarAge)) + geom_bar(stat="count") +
  labs(title = "CarAge repartition") + theme(plot.title = element_text(hjust = 0.5)) +
  guides(fill=FALSE) + scale_fill_manual(values = cc2)

ggArea = ggplot(data=dataTrain, aes(x=Area, fill = Area)) + geom_bar(stat="count") +
```

```
  labs(title = "Area repartition") + theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("Suburban" = "darkblue", "Urban" = "darkred",
                                         "Countryside low" = "brown", "Mountains" = "black"))

ggLeasing = ggplot(data=dataTrain, aes(x=Leasing, fill = Leasing)) +
  geom_bar(stat="count") + labs(title = "Leasing repartition") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("1" = "darkblue", "2" = "darkred", "3" = "black"))

ggPower = ggplot(data=dataTrain, aes(x=Power, fill = Power)) + geom_bar(stat="count") +
  labs(title = "Power repartition") + theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("1" = "darkblue", "2" = "darkred", "3" = "brown",
                                         "4" = "black"))

ggFract = ggplot(data=dataTrain, aes(x=Fract, fill = Fract)) + geom_bar(stat="count") +
  labs(title = "Fract repartition") + theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("1" = "darkblue", "2" = "darkred", "3" = "black"))

ggContract = ggplot(data=dataTrain, aes(x=Contract, fill = Contract)) +
  geom_bar(stat="count") +
  labs(title = "Contract repartition") + theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("1" = "darkblue", "2" = "darkred", "3" = "black"))

ggExposure = ggplot(data=dataTrain, aes(x=Exposure)) +
  geom_density(colour = "darkred", fill = "white") +
  labs(title = "Exposure repartition") + theme(plot.title = element_text(hjust = 0.5))

ggNbClaims = ggplot(data=dataTrain, aes(x = Nbclaims, fill = Nbclaims)) +
  geom_bar(stat="count") + labs(title = "Claims repartition") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_fill_manual("legend", values = c("0" = "darkblue", "1" = "darkred", "2" = "black"))

#reset dataset (not factors)
dataTrain = read.csv(pathTrain)
dataTest = read.csv(pathTest)
```

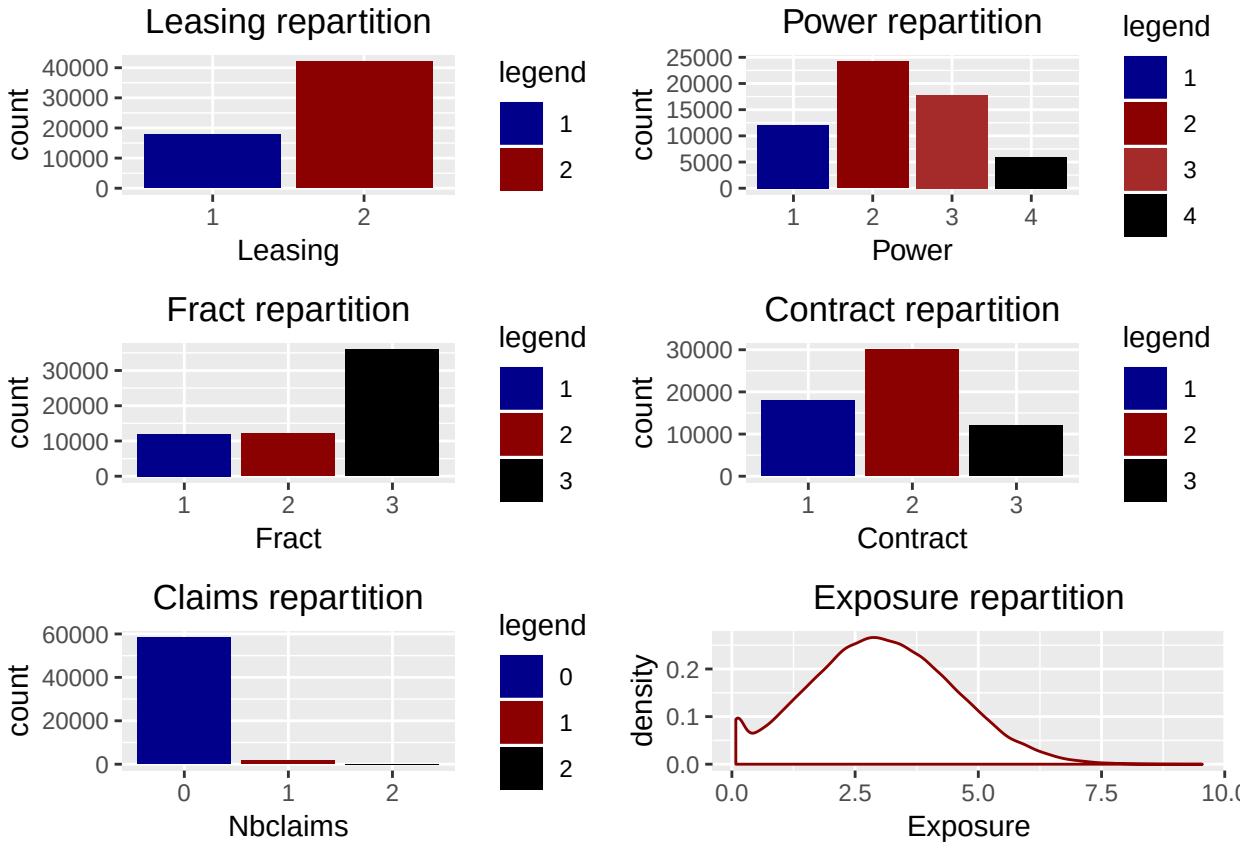
```
grid.arrange(ggGender, ggArea, ggDriverAge, ggCarAge, ncol = 2, nrow = 2)
```



Les graphiques ci-dessus nous informe sur la distribution des différentes variables du jeu de données. On observe ainsi assez vite que le dataset est composé de plus d'hommes que de femmes. En ce qui concerne la variable région, les contractants suburban et urban sont les deux majoritaires, la région montagnarde étant minoritaire.

L'âge des conducteurs : les conducteurs de 18 ans sont largement majoritaires avec plus de 3000.

```
grid.arrange(ggLeasing, ggPower, ggFract, ggContract, ggNbClaims, ggExposure, ncol = 2, nrow = 3)
```



Nous constatons ici que la plupart des véhicules ne sont pas en leasing et que la majorité des primes sont payées annuellement.

De plus il est intéressant de constater (bien que cela relève du bon sens) que la vaste majorité des contractants n'ont aucun sinistre déclaré durant la durée du contrat. Voici la table de répartition:

```
library(knitr)
kable(table(dataTrain$Nbclaims))
```

Var1	Freq
0	58380
1	1597
2	23

Preprocessing

En prenant :

$Y = X/w$

Où $X = numberofclaims$ et $w = Duration$

Nous obtenons la claim frequency qui est la variable à estimer.

La cross validation sera utilisée pour la validation.

On divise le dataset en k parties, ensuite pour chaque k on garge les k-1 autres parties comme échantillon d'apprentissage, avec k comme validation. L'opération est répétée pour tous les k. L'erreur de prédiction est ensuité estimée comme la moyenne quadratique des k erreurs estimées.

Enfin, nous allons transformer les variables catégorielles en facteurs afin de pouvoir les convertir en dummy variables, plus pratiques pour la construction de modèles. Les varaibles dummy seront nommées Var.factor (example : Gender.1)

```
set.seed(100)
library(caTools)
library(caret)
```

```
## Loading required package: lattice
```

```
#convert as factors so it can be turned into dummy vars
dataTrain$Gender = as.factor(dataTrain$Gender)
dataTrain$Area = as.factor(dataTrain$Area)
dataTrain$Leasing = as.factor(dataTrain$Leasing)
dataTrain$Power = as.factor(dataTrain$Power)
dataTrain$Fract = as.factor(dataTrain$Fract)
dataTrain$Contract = as.factor(dataTrain$Contract)

#convert as factors so it can be turned into dummy vars
dataTest$Gender = as.factor(dataTest$Gender)
dataTest$Area = as.factor(dataTest$Area)
dataTest$Leasing = as.factor(dataTest$Leasing)
dataTest$Power = as.factor(dataTest$Power)
dataTest$Fract = as.factor(dataTest$Fract)
dataTest$Contract = as.factor(dataTest$Contract)
#This column is added so the dummy function works, butwill not be used in predictions
dataTest = cbind(dataTest, "Nbclaims" = 0)

dummies <- dummyVars(Nbclaims ~ Gender + Area + Leasing + Power + Fract +
                       Contract, data = dataTrain)
dummiesTest <- dummyVars(Nbclaims ~ Gender + Area + Leasing + Power + Fract +
                           Contract, data = dataTest)


dumT = predict(dummies, dataTrain)
dumTe = predict(dummiesTest, dataTest)

#Now create data with dummies and attach the non categorical vars to it
dumTT = cbind(dumT, dataTrain$DriverAge, dataTrain$CarAge, dataTrain$Exposure,
              dataTrain$Nbclaims)
colnames(dumTT)[19:22] = c("DriverAge", "CarAge", "Exposure", "Nbclaims")

dumTTe = cbind(dumTe, dataTest$DriverAge, dataTest$CarAge, dataTest$Exposure,
               dataTest$Nbclaims)
colnames(dumTTe)[19:22] = c("DriverAge", "CarAge", "Exposure", "Nbclaims")

dumTT = as.data.frame(dumTT)
dumTTe = as.data.frame(dumTTe)
```

```
train = dumTT
test = dumTTe
```

Modeling

Nous allons maintenant tenter d'estimer les claim frequencies à l'aide de différents modèles. Pour la model validation, nous utiliserons une cross validation avec K = 10.

Pour estimer la claim frequency nous allons en réalité estimer NbClaims, il suffira ensuite de diviser cette valeur par l'exposure pour obtenir le key ratio souhaité.

```
fitControl <- trainControl(## 10-fold CV
                           method = "cv",
                           number = 10
                           )
```

GLM

Pour commencer, un modèle GLM sera construit, tentant d'estimer au mieux la claim frequency.

Le GLM est une généralisation souple de la régression linéaire. Il permet de travailler avec des variables expliquées qui ne sont pas distribuées selon une loi normale. Pour certaines variables donc, la régression linéaire n'est pas possible (c'est notre cas).

Un premier modèle GLM est estimé sur toutes les variables.

```
set.seed(100)
glmt = caret::train(Nbclaims ~ Gender.1 + DriverAge + CarAge + Area.1 + Area.2 + Area.3 +
                      Leasing.1 + Power.1 + Power.2 + Power.3 + Fract.1 + Fract.2 +
                      Contract.1 + Contract.2 + offset(log(Exposure)), data = train,
                    method = "glm", family = poisson(link = "log"), trControl = fitControl)

#summary(glmt)
glmt
```

```
## Generalized Linear Model
##
## 60000 samples
##    15 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 54000, 54000, 54000, 54000, 54000, 54000, ...
## Resampling results:
##
##   RMSE       Rsquared     MAE
##   0.1653687  0.001026019  0.05324031
```

On se rend vite compte que certaines variables sont peu pertinentes. Ainsi Power.3 peut être retirée du modèle sans trop de crainte étant donné sa pvalue élevée.

De manière similaire nous supprimons la variable Fract.2 du modèle ainsi que Area.1 et Leasing.1 (de manière itérative).

Pour obtenir le modèle final :

```
glmt5 = caret::train(Nbclaims ~ Gender.1 + DriverAge + CarAge + Area.2 + Area.3  +
                       Power.1 + Power.2 + Fract.1 + Contract.1 + Contract.2 +
                       offset(log(Exposure)), data = train, method = "glm",
                     family = poisson(link = "log"), trControl = fitControl)

summary(glmt5)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.3314  -0.2464  -0.2301  -0.2146   3.7345
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.212167   0.113873 -28.208  < 2e-16 ***
## Gender.1     0.099416   0.050867   1.954  0.05065 .
## DriverAge   -0.002307   0.001731  -1.333  0.18266
## CarAge      -0.016401   0.009694  -1.692  0.09066 .
## Area.2       0.127747   0.056001   2.281  0.02254 *
## Area.3      -0.175582   0.065338  -2.687  0.00720 **
## Power.1     -0.207943   0.070140  -2.965  0.00303 **
## Power.2     -0.074919   0.054339  -1.379  0.16798
## Fract.1      0.124563   0.059528   2.093  0.03639 *
## Contract.1  -0.319159   0.068384  -4.667 3.05e-06 ***
## Contract.2  -0.270291   0.060924  -4.437 9.14e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 11886  on 59999  degrees of freedom
## Residual deviance: 11819  on 59989  degrees of freedom
## AIC: 15095
##
## Number of Fisher Scoring iterations: 6
```

```
glmt5
```

```
## Generalized Linear Model
##
## 60000 samples
##    11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 54000, 54000, 54000, 54000, 54000, 54000, ...
## Resampling results:
##
```

```
##   RMSE       Rsquared      MAE
##   0.1654233  0.0009389389  0.053238
```

```
#rmse
RMSEGLM = glmt5$results$RMSE
cat("GLM RMSE : ", RMSEGLM)
```

```
## GLM RMSE :  0.1654233
```

```
library(dismo)
```

```
## Loading required package: raster
```

```
## Loading required package: sp
```

```
#deviance
predictions = predict(glmt5, newdata = train)
actual = train$Nbclaims
DEVGLM = calc.deviance(actual, predictions, family = "poisson", calc.mean = T)

cat("GLM Deviance : ", DEVGLM)
```

```
## GLM Deviance :  0.1969857
```

Nous utiliserons ces variables pour la random forest et le gbm, qui prennent trop de temps à s'éxecuter sans une feature selection.

Regression trees

Un arbre de décision est une méthode prédictive. L'arbre se divise à chaque feuille et les données passe dans la catégorie correspondant aux valeurs de leurs variables.

Par exemple pour déterminer si quelqu'un est majeur, une règle de décision simple serait $Age >= 18$.

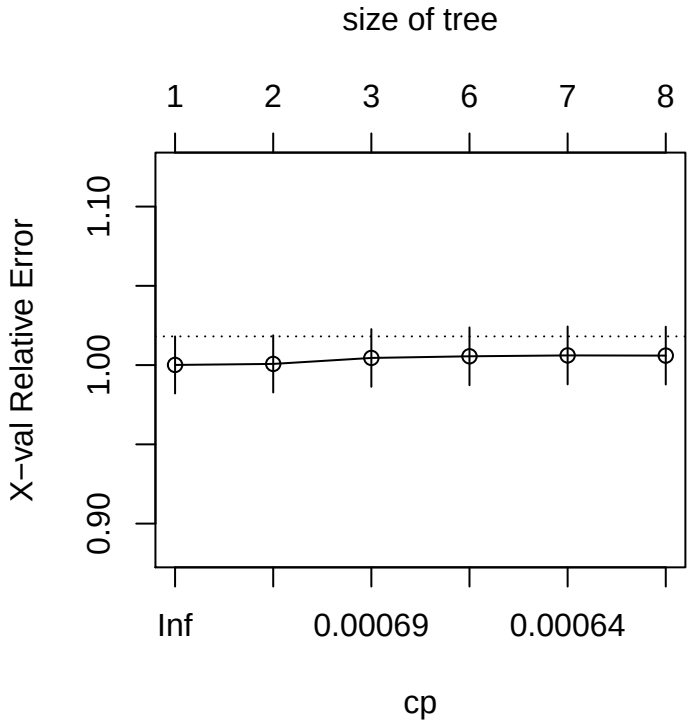
Le modèle sera réalisé avec $cp = 0.0006$.

```
library(rpart)
library(Metrics)
```

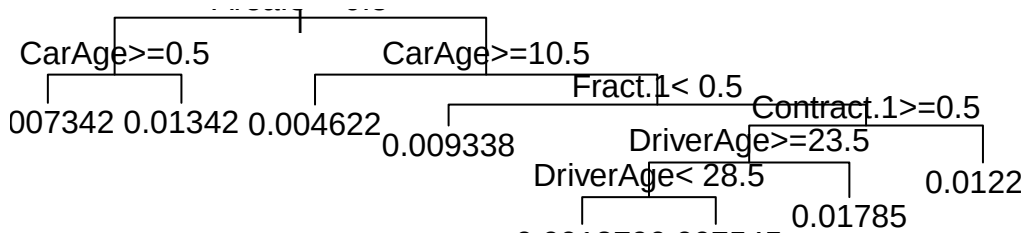
```
##
## Attaching package: 'Metrics'
```

```
## The following objects are masked from 'package:caret':
##
##     precision, recall
```

```r
#Using rpart
set.seed(100)

rpart2 = rpart(Nbclaims ~ Gender.1 + DriverAge + CarAge + Area.1 + Area.2 + Area.3 +
                 Leasing.1 + Power.1 + Power.2 + Power.3 + Fract.1 + Fract.2 +
                 Contract.1 + Contract.2 + offset(log(Exposure)), data = train,
               method = "poisson", control = rpart.control(cp = 0.0006, xval = 10), parms = list(shrink
#summary(rpart2)

plotcp(rpart2)
```




```r
plot(rpart2)
text(rpart2)
```



L'arbre de décision est visible sur le graphique ci-dessus.

```
predictions = predict(rpart2, newdata = train)
actual = train$Nbclaims
library(Metrics)
RMSERCART = rmse(actual, predictions)

library(dismo)
DEVRCART = dismo::calc.deviance(actual, predictions, family = "poisson", calc.mean = T) #The deviance h

cat("RMSE : ", RMSERCART)
```

```
## RMSE :  0.1664942
```

```
cat("Deviance : ", DEVRCART)
```

```
## Deviance :  0.2209103
```

Random forest

La "random forest" ou "Forêts aléatoire" est une méthode construisant de multiples arbres de décision et renvoie la prédiction moyenne comme résultat (pour la régression) ou le mode (classification).

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
rf = randomForest(Nbclaims ~ Gender.1 + DriverAge + CarAge + Area.2 + Area.3  +
                    Power.1 + Power.2 + Fract.1 + Contract.1 + Contract.2 +
                    offset(log(Exposure)), data = train)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
mses = rf$mse
rmses = sqrt(mses)
RMSERF = mean(rmses)
cat("RMSE random forest : ", RMSERF)
```

```
## RMSE random forest :  0.1663828
```

```
#dev rf
predictions = predict(rf, train)
actual = train$Nbclaims
DEVRF = calc.deviance(actual, predictions, family = "poisson", calc.mean = T)
```

GBM

```
library(gbm)
```

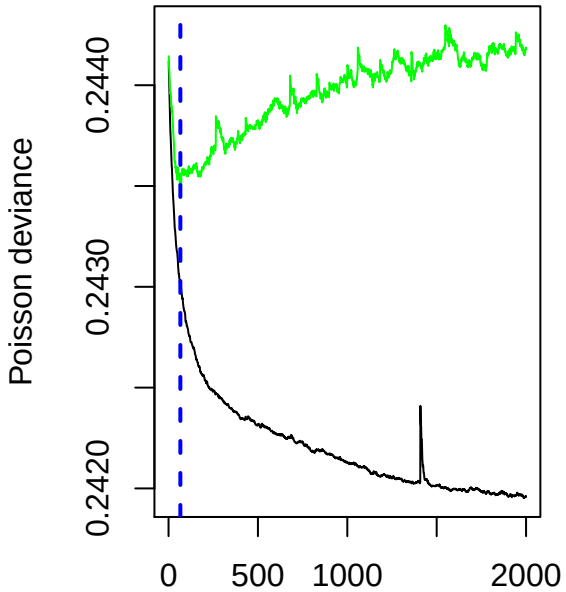
```
## Loaded gbm 2.1.4
```

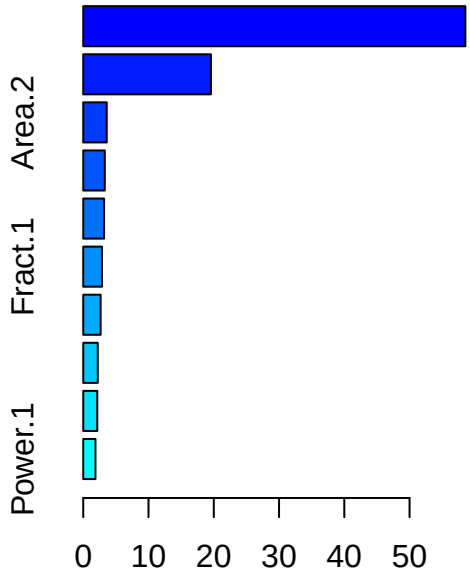
```
gbmFit = gbm(Nbclaims ~ Gender.1 + DriverAge + CarAge + Area.2 + Area.3  +
               Power.1 + Power.2 + Fract.1 + Contract.1 + Contract.2 +
               offset(log(Exposure)), distribution = "poisson", data = train,
             n.trees = 2000, cv.folds = 10, verbose = "CV")
par(mfrow = c(1,2))

gbmFit
```

```
## gbm(formula = Nbclaims ~ Gender.1 + DriverAge + CarAge + Area.2 +
##     Area.3 + Power.1 + Power.2 + Fract.1 + Contract.1 + Contract.2 +
##     offset(log(Exposure)), distribution = "poisson", data = train,
##     n.trees = 2000, cv.folds = 10, verbose = "CV")
## A gradient boosted model with poisson loss function.
## 2000 iterations were performed.
```

```
## The best cross-validation iteration was 66.
## There were 10 predictors of which 10 had non-zero influence.
```

```
summary(gbmFit)
```






```
##                   var   rel.inf
## DriverAge   DriverAge 58.553026
## CarAge         CarAge 19.567611
## Area.2         Area.2  3.599915
## Contract.2 Contract.2  3.300427
## Gender.1     Gender.1  3.190017
## Fract.1       Fract.1  2.874723
## Contract.1 Contract.1  2.666746
## Area.3         Area.3  2.226920
## Power.2       Power.2  2.148278
## Power.1       Power.1  1.872337
```

```
original = train$Nbclaims
RMSEGBM = rmse(original, fit)

cat("GBM RMSE : ",RMSEGBM)
```

```
## GBM RMSE :  4.753501
```

Le RMSE est ici beaucoup trop élevé, cela vient peut être d'une erreur dans la construction du modèle. Quoiqu'il en soit il ne sera pas utilisé pour les prédictions.

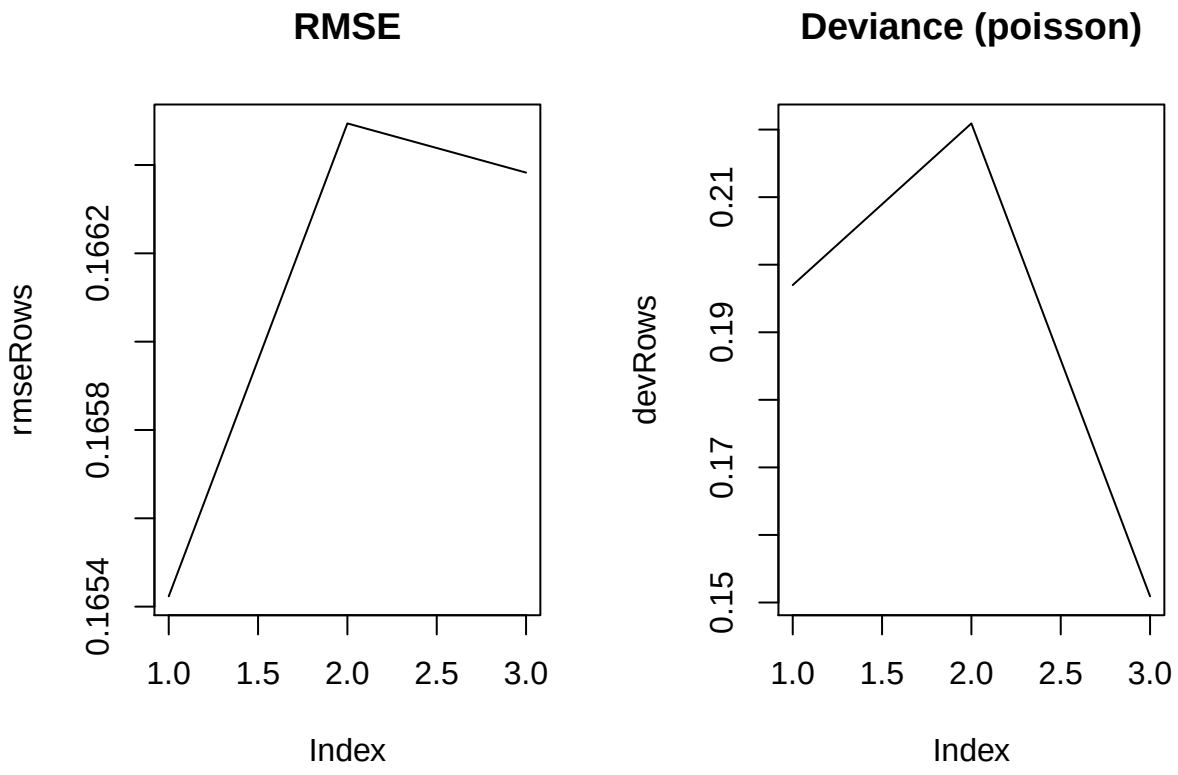
Model selection

Pour choisir notre modèle, comparons leurs résultats des différents modèles (sauf le GBM). Nous pouvons regarder le RMSE et la deviance.

Pour une loi poisson (claim frequency), la deviance se calcule comme :

$2 * \sum_{n=1}^{n}(Y_i log(Y_i/\mu_i) - (Y_i - \mu_i))$

```
par(mfrow = c(1,2))
rmseRows = rbind(RMSEGLM, RMSERCART, RMSERF)
plot(rmseRows, type = "l", main = "RMSE")
devRows = rbind(DEVGLM, DEVRCART, DEVRF)
plot(devRows, type = "l", main = "Deviance (poisson)")
```



Voici les RMSE de nos modèles, de gauche à droite respectivement : GLM, CART, et RF.

Il est clair ici que le GLM est le meilleur modèle pour le rmse.

Si nous regardons la deviance nous choisissons le modèle random forest.

Pour les résultats finaux nous nous baserons sur le rf (modèle minimisant la deviance).

Afin d'être sûr des résultats et d'éviter l'overfitting, nous pourrions diviser le trainSet(75/25) et tester les modèles sur la partie ainsi créée pour la validation des modèles. Cependant par manque de temps ceci ne sera pas fait dans ce projet.

Predictions

Pour prédire le key ratio (claim frequency), nous devons d'abord prédire la variable Nbclaims. Une fois celle-ci prédite, il suffit de la diviser par la variable Exposure afin d'obtenir les résultats souhaités.

Comme demandé dans l'énoncé, le fichier est sous format csv séparé par des points. Le séparateur décimal a donc été changé en une virgule (sinon des problèmes surviendraient lors de l'importation des données pour la variable Exposure).

```
rfPredictions = predict(rf, newdata = test)
rfPredFreq = rfPredictions/test$Exposure

#reset for dummies
dataTest = read.csv(pathTest)
#export to csv
DBtest = cbind(dataTest, rfPredFreq)
colnames(DBtest)[11] = "ClaimFrequency"
write.table(DBtest, file = "DBtest.csv", sep = ".", dec = ",")
```

Rating factors

Nous choisissons le modèle random forest, nous pouvons donc nous baser sur celui-ci pour déterminer quelles seront les variables les plus discriminantes.

```
discr = rf$importance
kable(discr)
```

	IncNodePurity
Gender.1	7.223752
DriverAge	67.623211
CarAge	36.677995
Area.2	4.487195
Area.3	3.510021
Power.1	4.804228
Power.2	6.992258
Fract.1	6.445697
Contract.1	5.243723
Contract.2	5.906878

On voit ici que DriverAge et CarAge sont les deux facteurs les plus importants. Cela peut être assez intuitif car les jeunes conducteurs sont plus propices à causer des accidents et un véhicule datant de plusieurs années peut ne pas être équipé de toutes les options modernes d'aide à la conduite (ABS, ...).