Rapport Data Science

Mehdi KHAIROUN, William LAURENT, Pierre MARJOLLET, Afaf TABAMMOUT

9 avril 2018

# I. Matériel et pré-processing

## 1. Jeu de données

Les données étudiées sont deux bases issues de la librairie CASdatasets (CAS pour Computational Actuarial Sciences). **freMTPL2freq** et **freMTPL2sev** sont deux bases autos.

Brièvement **freMTPL2freq** est une base de fréquence, nous donnant des caractéristiques pour des polices d'assurances données et leurs nombre de sinitres respectifs.**Cette base dispose de 12 variables et 678 013 individus**, comprenant :

* **IDpol** [Clé primaire : Entier de 1 à 6 114 330] : l'identifiant de la police d'assurance
* **ClaimNb** [Entier de 0 à 16] : le nombre de sinistres de la police
* **Exposure** [Réel de 0 à 1] : l'exposition - la durée de vie de la police. De 0 pour une personne non assuré à 1 pour une personne assurée toute l'année
* **Area** [Modalité de A à F] : indication sur la zone d'apparition du sinistre
* **VehPower** [Entier de 5 à 15] : puissance du véhicule
* **VehAge** [Entier de 0 a 100] : age du véhicule
* **DrivAge** [Entier de 18 à 100] : age du conducteur
* **BonusMalus** [Entier de 50 à 230] : bonus/malus du conducteur. Plus le score est haut, moins l'assuré est sanctionné par son passé
* **VehBrabd** [Modalité de B1 à B12] : marque du véhicule
* **VehGas** [Booléent : 0 ou 1] : véhicule diéserl ou non
* **Density** [Entier de 1 à 1658] : densité de population de la police
* **Region** [24 modalité de R11 à R94] : code région de la police

Enfin, **freMTPLsev** est une base sévèrité, avec simplement un identifiant et une valeur de sinistre. **Cette base comprends 2 variables et 26 639 individus**:

* **IDpol** [Entier de 139 à 6113971] : l'identifiant de la police d'assurance
* **ClaimAmount** [Réel de 1 à 4 075 401] : coût du sinistre

## 2. Pré-processing

Nous avons calculé la moyenne des coûts des sinistres par polices sur la base **freMTPLsev**. Le but étant de savoir quelle est la moyenne des sinistres d'une police.

Nous avons ensuite fusionné les bases **freMTPLsev** et **freMTPLfreq** par **IDpol** maintenant unique sur les deux bases.

Il vient ensuite une base sur laquelle nous retirons les sinistres graves (ceux supérieures à ). Ce seuil a été déterminé rapidement par l'étude d'un mean-excess plot. Ces quelques sinistres biaisais fortement nos modélisations et seront répartie comme une sur-prime sur l'ensemble de nos assurés.

Nous basons nos modélisations sur seulement de nos données pour se laisser une bonne marge de test. Ensuite, pour les procédures plus lourde (typiquement le gradian boosting et les réseaux de neuronnes) les derniers modèles retenus ont été entraîner sur un sous jeux de cette base d'entraînement.

Pour les modélisations en sévérité, nous avons biensûre enlevé les polices ne présentant aucun sinistre.

## 3. R et ses extentions

Dans ce projet nous utilisons le logiciel opensource R et nous utilisons son extention RMarkdown pour la rédaction de ce rapport.

Pour pouvoir mener à bien ce projet, nous avons munit R des librairies suivantes :

* **caret** : **C**lassification **a**nd **re**gression **t**raining, utilisé ici pour utiliser nos algorithmes aisément sur des grilles de paramètre de tuning. Propose également une parrallélisation de ses algorithmes. **caret** dispose de tous les algorithmes nécessaire au bon déroulement de ce projet et pourrait nous suffire ici.
* **doParallel** : s'accouple à **caret** pour accéder à la parrallélisation des algorithmes et ainsi économiser quelques précieuses heures de run.
* **rpart** : permets de dresser des arbres (ici des arbres de régressions).
* **rpart.plot** : utilisé ici pour afficher des graph d'arbres de régressions.
* **magrittr** : donner au code plus de lisibilité.
* **etc**

# II. Arbres de régression

# III. Forêts aléatoires

## 1. Présentation du procédé

Avant de parler de random forests (forêts aléatoires) parlons de Bagging. Le bagging de Bootstrap Aggregating est simplement un bootstrap sur des modélisations données. C'est une méthode d'aggrégation de modèles. Nous avons fait poussé des arbres dans la partie précédente; nous allons ici en faire poussé beaucoup et les aggrégés pour fabriquer des forêts!

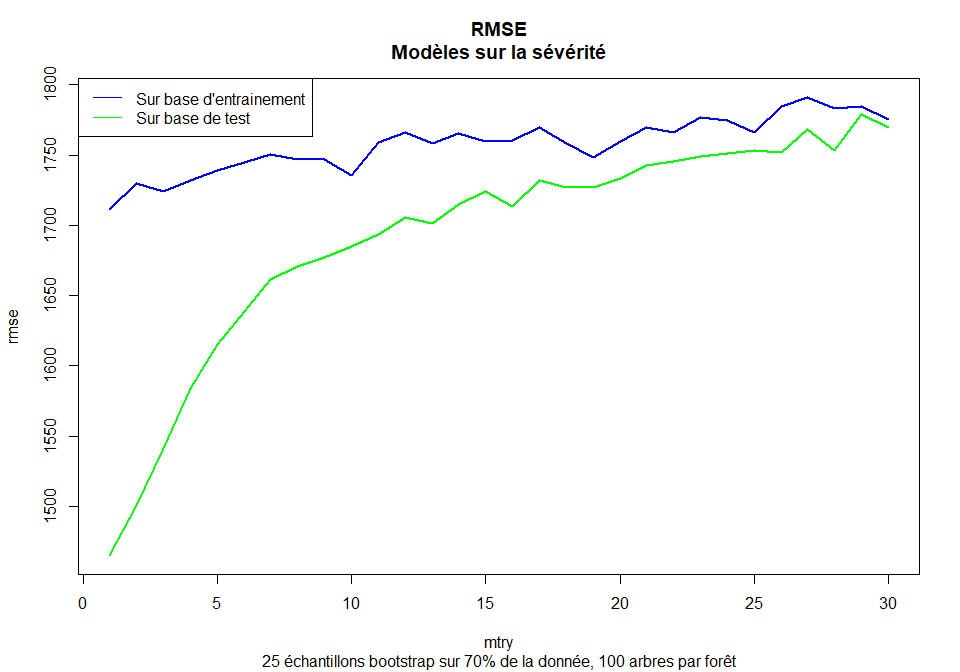
La méthode de random forest que nous allons exploiter ici est celle qui consiste à faire un Bagging sur des arbres de régressions maximaux. On va faire pousser une forêt de arbres et chacun de ces arbres aura un choix aléatoire de variables a chaque noeud. Le seul hyperparamètre qui nous intéressera par la suite est . lui est un paramètre sur la convergence des données, il va de sois qu'avec un grand nous convergerons que mieux. lui :

* **Pour des petites valeurs** : les noeuds disposerons que de peu de variable pour se former, l'arbre peu être unitairement pauvre en puissance prédictive mais la correlation entre les arbres est minimisée.
* **Pour des grandes valeurs** : les noeuds disposerons de beaucoup de variable pour se former, l'arbre peu être riche en puissance prédictive (comme un CART maximal) mais la correlation entre les arbres sera forte
* Ainsi, un sera déterminé par grille de paramêtres.

## 2. Modèlisation du coût moyen

### A. Recherche de

Nous avons fais tourner plusieurs forêts pour obtenir un satisfaisant :

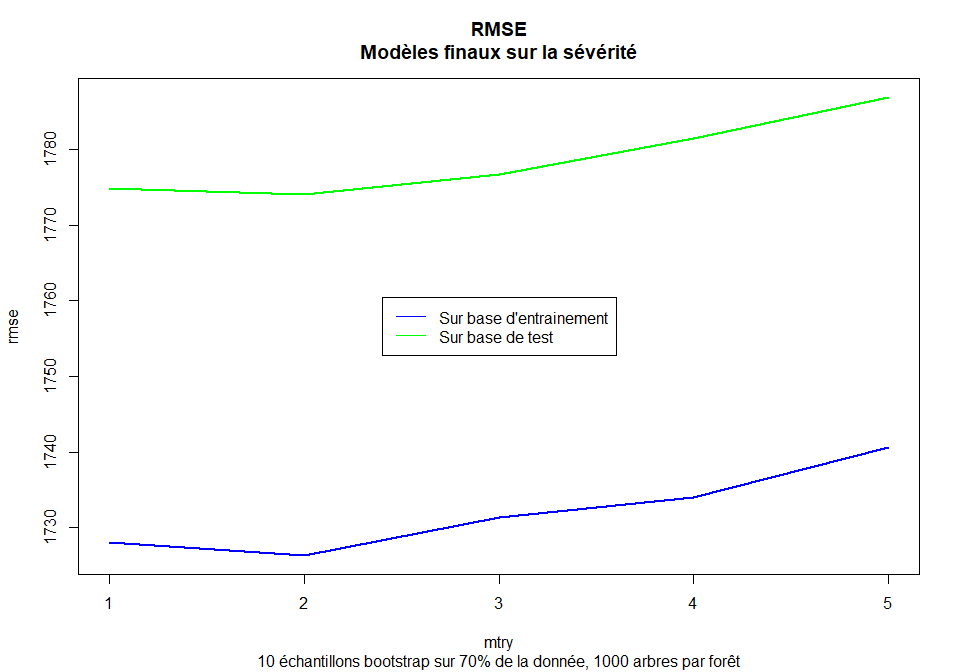


Sur ce graph on trouve que la région des bons se trouve entre et . Dans la pratique, pour un problème de régression, randomForest conseillera de prendre la racine du nombre de variable (vers , ici).

On va refaire pousser des fôrets cette fois-ci en poussant le nombre d'arbre à par forêt pour ensuite séléctionner un modèle final

## B. Comparaison des forêts retenues

Enfin,après avoir poussé le nombre d'arbre par forêts :



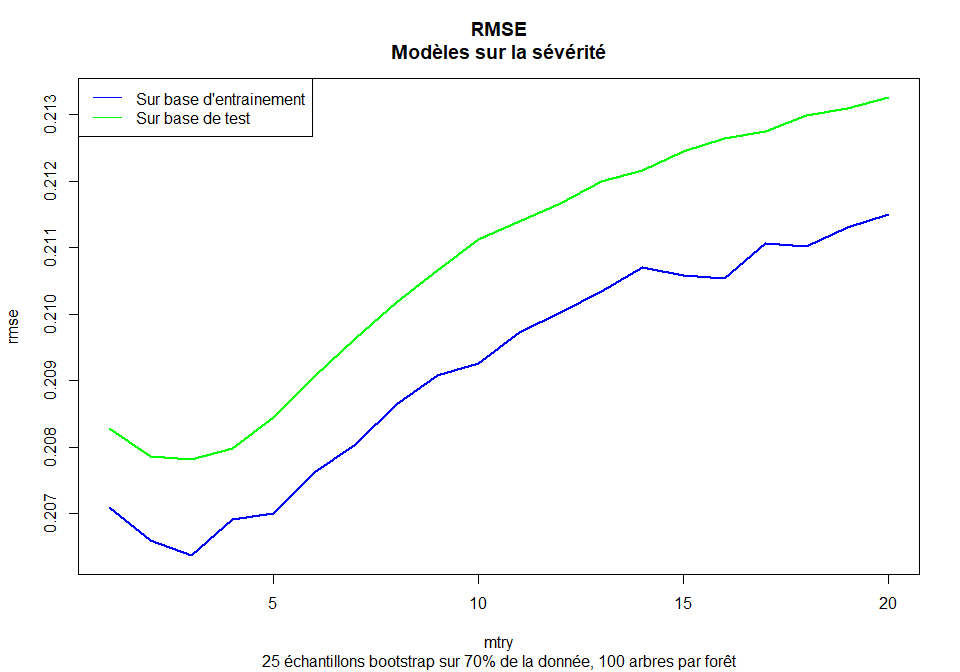
## Proportion de donnée RMSE entrainement RMSE test Biais test  
## mtry = 1 70% 1728.041 1774.857 -17.231970  
## mtry = 2 70% 1726.266 1774.113 -16.586121  
## mtry = 3 70% 1731.359 1776.706 -12.721779  
## mtry = 4 70% 1733.952 1781.493 -8.030443  
## mtry = 5 70% 1740.604 1787.002 -2.163842

Et le meilleur modèle est celui avec même s'il est plutôt biaisé par rapport aux autres, la RMSE est meilleure donc on se trompe moins pour chacune des polies mais en moyenne on se trompe plus.

## 3. Modélisation de la fréquence

### A. Recherche de

Comme pour la partie précédente, nous avons tourner nos algorithme sur sur la sévérité pour obtenir une idée d'un bon



On retient alors que sur les modèles de fréquences un faible (moins de ) fera l'affaire. Nous allons quand même essayer plusieurs statégies pour améliorer nos prédictions, notamment comment exploiter la variable ?. On prendra pour la séléction des meilleurs modèles.

### B. Réflexion sur la variable

Nous avons essayé trois modélisations différentes sur un petit set de donnée :

* En utilisant la variable comme variable explicative
* En utilisant la variable comme offset
* En divisant par

Les résultats sont les suivants :

## Gestion de l'exposition RMSE sur donnée de test  
## 1 Exposition en offset 0.2411243  
## 2 Division par Exposition 0.3343586  
## 3 Exposition en explicative 0.2400976

Les résultats sont légèrement meilleurs quand l'exposition est en variable explicative, ce qui est douteux. Probablement du fait que le dataset était tout petit. On va pousser la taille de l'échantillon et finir par convenir à un modèle final.

### C. Modèles finaux

Comparons maintenant les résultats obtenus avec les deux modèles retenus pour la fréquence

## Proportion de donnée  
## Exposition comme poids 15%  
## Exposition comme variable explicative 15%  
## RMSE entrainement RMSE test  
## Exposition comme poids 0.2379205 0.2404660  
## Exposition comme variable explicative 0.2369624 0.2394662  
## Biais test  
## Exposition comme poids 0.0001741779  
## Exposition comme variable explicative 0.0004887422

Les deux modèles se vaux. En effet, l'un est meilleur en mais est moins bien en Biais. On va préferer le modèle avec l'exposition comme offset car le biais est négatif; ainsi, on sur-estime la fréquence d'apparition des sinistres.

# IV. Gradiant boosting machine

## A. Préparation des données

Dans cette partie nous décrivons succintement les résultats obtenus par une modélisation GBM avec arbres de régression. Nous procédons à une décomposition fréquence coût moyen. Dans toute cette partie, nous utiliserons le package "xgboost", qui implémente de façon plus efficiente le GBM que le package "gbm".

Sans entrer dans les détails techniques, nous listons ici les différents problèmes rencontrés dans la mise en forme des données pour le package xgboost.

1. La fonction de fit ne prend en entrée que des variables numériques (c'est classique en machine learning): nous utilisons le package "caret" pour créer des flags de nos variables catégorielles.
2. Pour utiliser la fonction la plus complète du package pour l'entrainement (xgb.train), nous devons passer les données en entrée au format xgb.DMatrix
3. Pour la modélisation de la fréquence, où la base utilisée est déjà très volumineuse, créer des flags de variables catégorielles avec relativement beaucoup de modalités pose des problèmes de stockage (en effet, nos pc ne sont pas des machines de guerre):  
   3.1. Une des fonctions, xgboost, permet de passer en paramètre des sparse Matrix, mais la fonction est moins complète au niveau du contrôle des paramètres de tuning. 3.2. Nous créons sur une machine puissante les flags, et nos enregistrons les données au format xgb.DMatrix en dur.

## B. Modèle de coût

On optimise les paramètres de tuning avec une grid search à l'aide du package caret: on fixe un petit nombre d'arbres (10) pour entrainemer les modèles et on cherche les paramètre optimaux à l'aide d'une cross validation. On utilise le RMSE comme critère de choix de modèle.

On identifie des paramètres important pour éviter l'overfitting: min\_child\_weight, subsample, col\_sample.

Le RMSE semble être plus sensibles aux paramètres max\_depth (la profondeur maximale des arbres), nround (le nombre d'arbre).

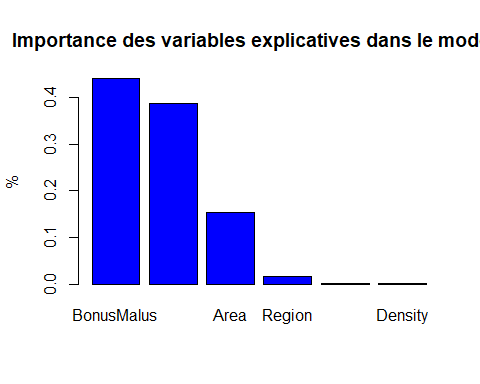
Le temps de calcul est très sensible aux paramètres max\_depth et nrounds. En outre, le temps de calcul est très sensibles au volume des données passées en entrée.

Nos paramètres optimaux sont:

Nous cherchons maintenantle nombre otpimal d'arbres. Nous obtenons 386 arbres.

Voici une représentation graphique du RMSE en fonction du nombre d'abres:

Nous pouvons également nous intéresser à l'influence respective de chaque variable explicative:



Arrive en tête la variable BonusMalus suive de très près par la variable VehAge.

Le biais que présente notre modèle d'environ

## [1] 1401.117

Cela signifie que l'on surestime en moyenne le montant des sinistres.

Le RMSE sur la base de test est

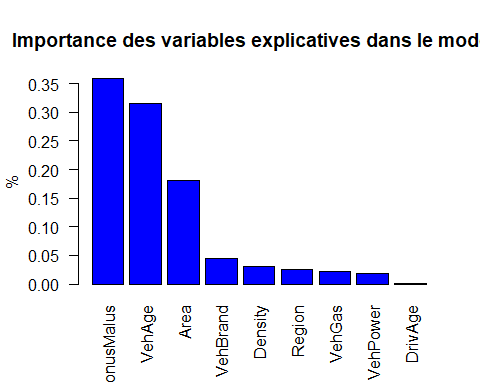
## [1] 1407.873

Le test de notre modèle sur la base de test révèle que notre modèle possède un biais de plusieurs dizaines, à la baisse. Cela signifie que nous sous-estimons les montants des sinistres attritionnels.

## C. Modèle de fréquence

Nous devons prendre en compte l'exposition. Nous définissons la variable à expliquer comme le nombre annuel de sinistres. Nous divisons donc le nombre de sinistres par l'exposition pour chaque police et nous pondérons également les observations par l'exposition.

Nous ne cherchons pas en profondeur les paramètres optimaux. Nous retenons les paramètres de tuning suivants à la main en parcourant une courte grille de paramètres. Nous ne parvenons pas à utiliser le package caret pour cette tâche. Nous avons donc créé une petite fonction pour palier à ça. Nous calcul simplement le RMSE sur la base de validation pour choisir les paramètres.

On évalue ensuite l'influence des différentes variables explicatives. 

Les variables arrivant en tête sont BonusMalus et VehAge.

Notre modèle présente également un biais de:

## [1] -0.01683648

Cela signifie que l'on sous-estime très légèrement en moyenne notre fréquence de sinistres.

Le RMSE sur la base de test est de:

## [1] 0.5939583

Les tests sur la base de test révèlent que notre modèle est également biaisé.

# V. Réseaux de neuronnes

A. Présentation du procédé

Le réseau de neurones artificiels a été inspiré du mécanisme du système nerveux humain, à travers l’histoire on a cherché à imiter son fonctionnement en commençant par le neurone formel apparu en 1943 avec une modélisation mathématique du neurone biologique, puis par les premières règles d’apprentissage de Hebb en 1949 et de l’algorithme d’apprentissage de rétropropagation en 1986 par Rumelhart.

Cet algorithme est utilisé dans le cadre des problématiques de classification, de reconnaissance de formes, d’association, d’extractions de caractéristiques, d’identification…

En assurance, l’avantage de cet algorithme est de pou voir capter les dépendances non linéaires entre les variables explicatives pour pouvoir affiner le tarif et appréhender le risque.

Pour les parties qui suivent, le paramètre le plus important qui serait à déterminer est représenté par le nombre de neurones sur la couche cachée parallèlement aux conditions d’apprentissage que nous citerons par la suite:

Pour le calibrage des deux modèles Fréquence et Coût, nous avons utilisé plusieurs fonctions intégrées sous R:

* Tune.nnet du package
* Nnet du package

# VI. Conclusion

# VII. Annexes et références

## 1. Annexes

## 2. Références