EURIA
Euro-Institut d'Actuariat

optimind winter

Rapport de Bureau d'étude - Groupe n°8

MASTER 1 Actuariat

Provisionnement à l'aide de modèles de machine learning

Yoan CALAS - Coralie CAPPE - François-Henri TOUTAIN

Travail supervisé par :

Emmanuel BERTHELE (Actuaire) - André GRONDIN (Actuaire)
Nicolas LE BERRIGAUD (Actuaire) - Tristan MUSCAT (Actuaire)
Franck VERMET (Enseignant - Chercheur)

2016-2017

Remerciements

Avant toute chose, nous tenons à remercier Messieurs Emmanuel BERTHELE, André GRONDIN, Nicolas LE BERRIGAUD et Tristan MUSCAT - actuaires chez Optimind Winter et tuteurs de notre bureau d'étude tout au long de cette année - pour leur disponibilité, leur attention en particulier lors de nos rendez-vous téléphoniques, ainsi que pour leurs précieux conseils qui nous ont permis d'avancer dans notre travail.

Nous tenons, par ailleurs, à exprimer notre reconnaissance à Monsieur Franck VERMET, directeur des études de l'EURIA, responsable de l'encadrement des bureaux d'étude et enseignant – chercheur également tuteur de notre bureau d'étude pour l'aide qu'il nous a apportée et les indications qu'ils nous a fournies pour nous guider au mieux dans notre travail.

Enfin, nous souhaitons remercier Monsieur Anthony NAHELOU pour les conseils complémentaires qu'il a pu nous donner lorsque nous en avions besoin.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Dans le cadre de l'année de Master 1 à l'Euro-Institut d'Actuariat de Brest, des professionnels proposent aux élèves différents sujets concrets accompagnés de problématiques, auxquels ils pourraient être confrontés dans leur vie professionnelle. L'étude de ces sujets fait l'objet d'un travail de groupe appelé « bureau d'étude », encadré par un ou plusieurs professionnels ainsi que par un enseignant. Il a pour but de tester la capacité des étudiants à travailler en équipe sur un sujet professionnel encore « inconnu » à leurs yeux, tout en utilisant les connaissances acquises lors de leurs deux premières années d'études au sein de l'EURIA.

Le sujet sur lequel notre groupe a travaillé nous a été confié par l'entreprise Optimind Winter, et consiste à provisionner des dossiers de sinistres en IARD à l'aide de modèles de Machine Learning.

De nos jours, le marché de l'assurance automobile est un marché concurrentiel avec une centaine d'opérateurs sur le marché. L'assurance automobile représente une part importante dans le marché de l'assurance. En effet, elle représente presque 40% de l'ensemble des cotisations des assurances de dommages aux biens et responsabilité civile (en 2014), et on compte à cette époque environ 40,2 millions de véhicules assurés en France. La tarification automobile est en quelque sorte variable selon des critères propres au véhicule ou au conducteur. En effet, les tarifs d'assurance automobile sont établis à partir de statistiques qui portent sur le nombre et le coût des accidents. Ces statistiques montrent que certaines catégories de véhicules et de conducteurs sont à l'origine de sinistres plus nombreux ou plus graves, ainsi la cotisation d'assurance n'est pas la même pour tous, d'où l'importance de la tarification.

L'objectif final de notre projet était de réaliser un modèle prédictif du montant total d'un sinistre automobile en fonction de plusieurs critères, que l'on aura ensuite comparé avec les modèles déjà utilisés dans les compagnies d'actuariat après avoir testé la qualité de chacun des modèles créés.

Le premier travail à réaliser, et un des plus importants dans notre étude, était la prise de connaissance de la base de données, ainsi que l'apport de modifications sur cette dernière afin de la rendre la plus opérationnelle possible. Cette étape du travail est importante car c'est à partir de cette base de données optimisée que tous les modèles et calculs seront réalisés.

La deuxième partie de notre travail était alors dédiée à la réalisation de différents modèles de machine learning. Dans un premier temps, nous avons réalisé un modèle simple se basant sur les moyennes des montants à prédire, puis nous nous sommes concentrés sur trois autres modèles qui sont le modèle linéaire généralisé (GLM), le modèle RandomForest, et le modèle gradient boosting (GBM).

Chapitre 2

L'assurance automobile en France

2.1 Quelques chiffres clés [1]

Comme énoncé précédemment, l'assurance automobile prend de plus en plus d'importance sur le marché des assurances, en effet, avec une représentation de presque 40% de l'ensemble des cotisations des assurances de dommages aux biens et de responsabilité civile. Le nombre de véhicules assurés aujourd'hui dépasse les 40 millions de véhicules (de première catégorie hors flottes).

Ce nombre ne cessant d'augmenter, des variations dans les cotisations, dans les primes ou encore dans la sinistralité sont observées.

– Hausse des cotisations :

Le nombre de véhicules en service étant en hausse chaque année, les cotisations en assurance automobile augmentent également, atteignant alors plus de 16 millions d'euros.

	2012	**2011**	**2012**	**2013**	**2014**
Cotisations automobiles des particuliers	14,4 Md€	14,9 Md€	15,5Md€	15,8 Md€	16,1 Md€
Variation annuelle	*+ 3,1%*	*+ 3,6%*	*+ 3,5%*	*+ 2,2%*	*+ 2,2%*
Nombre de véhicules 4 roues assurés	38,1 M	38,6 M	39,2 M	39,6 M	40,2 M
Variation annuelle	*+ 1,3%*	*+ 1,4%*	*+ 1,4%*	*+ 1,1%*	*+ 1,5%*

TABLE 2.1 – Variations des cotisations automobiles annuelles des particuliers

– Baisse de la sinistralité :

En 2014, on estime le nombre de sinistre indemnisés pour les véhicules de première catégorie hors flotte à 7,7 millions, soit une baisse de 3,3% par rapport à l'année précédente.

Néanmoins, on observe une augmentation des coûts moyens des sinistres matériels, avec notamment une hausse de 2,4% pour les sinistres RC matériels, ainsi qu'une hausse de 3,2% pour les bris de glace.

Par ailleurs, le nombre d'accidents corporels est en baisse depuis plusieurs années ; et représente une forte part des indemnisations versées. En effet, alors que les blessés légers représentent la majorité des sinistres corporels avec près de 71%, leur indemnisation ne représente que 5% de l'ensemble des indemnités versées, tandis que les blessés les plus graves ne représentent que 2% des cas pour 58% de la charge totale des indemnités versées.

1. Chiffres donnés à partir des statistiques de l'année 2014

Enfin, sur la période 1999-2014, le montant des garanties directement liées aux aléas naturels et climatiques est compris entre 1 et 3,5% de la charge annuelle des sinistres, avec une moyenne annuelle de 1,8%.

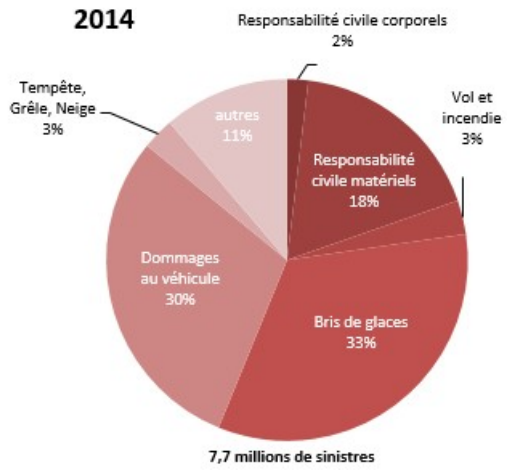


FIGURE 2.1 – Répartition du nombre de sinistres selon la garantie

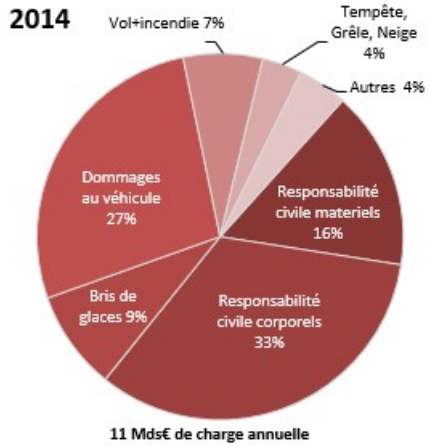


FIGURE 2.2 – Répartition de la charge de sinistres selon la garantie

– Hausse de la prime moyenne :

Quelle que soit la formule d'assurance, le montant de la prime moyenne augmente chaque année.

	Prime moyenne 2014	**Variation 2014/2013**	**Variation 2013/2012**	**Variation 2012/2011**
Ensemble des véhicules de 1ère catégorie	400€	+ 0,9%	+ 0,9%	+ 2,1%
Véhicules assurés au tiers [2]	236€	+ 1,2%	+ 1,6%	+ 2,6%
Dont responsabilité civile	145€	+ 1,0%	+ 1,8%	+ 1,6%
Tous risques (hors assistance)	485€	+ 1,4%	+ 1,2%	+ 2,5%

TABLE 2.2 – Niveau de prime moyenne selon le type de contrat

2.2 Critères de tarification

La cotisation n'est pas uniforme pour tous les assurés automobiles. En effet, selon des critères propres au véhicule ou selon des critères propres au conducteur, le tarif peut varier.

2.2.1 Les critères propres au véhicule

– Les caractéristiques du véhicule :

Deux critères principaux de tarification :

2. Véhicules assurés en RC, vol, incendie et bris de glaces

• La puissance du véhicule :

Les véhicules les plus puissants sont à l'origine d'un plus grand nombre d'accidents. La vitesse excessive constitue une des causes majeures de la sinistralité corporelle de la circulation.

• La valeur à neuf du véhicule :

Les réparations d'un véhicule haut de gamme sont plus onéreuses que celles d'un véhicule bas de gamme. Les frais à indemniser ne seront pas les mêmes pour les deux catégories de véhicule en cas d'accident.

– La zone géographique de circulation ou de garage :

La zone dans laquelle un automobiliste est amené à circuler va influer sur le tarif de la cotisation. En effet, les habitants d'une région rurale bénéficient d'une cotisation d'assurance plus avantageuse que les habitants d'une grande agglomération, car les automobilistes qui circulent habituellement dans des zones à forte concentration urbaine provoquent plus d'accidents que les autres.

Pour connaître ou choisir la zone de tarification, on regarde généralement la zone de stationnement du véhicule.

– L'usage du véhicule :

Selon le type d'usage de leur véhicule, les tarifs peuvent varier. Les automobilistes qui utilisent leur véhicule pour les besoins de leur profession provoquent plus d'accidents que ceux qui s'en servent uniquement pour un usage privé.

C'est pourquoi certaines sociétés d'assurances proposent des tarifs plus avantageux aux conducteurs qui effectuent un kilométrage annuel faible.

2.2.2 Les critères propres au conducteur

Parfois, il arrive que plusieurs conducteurs viennent à utiliser régulièrement un même véhicule, dans ce cas, les critères propres à chacun des conducteurs sont à prendre en compte dans la tarification de la cotisation d'assurance.

– La conduite occasionnelle :

Lorsqu'un conducteur (considéré comme jeune conducteur : âgé de moins de 25 ans ou titulaire d'un permis datant de moins de deux ou trois ans) emprunte occasionnellement un véhicule, en cas d'accident, généralement, une franchise supplémentaire reste à la charge de l'assuré.

– La conduite exclusive :

Souvent, lorsqu'un assureur s'engage à ce que lui seul (ou lui et son/sa conjoint(e) seuls) soit conducteur de son véhicule, il bénéficie d'une diminution tarifaire. Une franchise supplémentaire sera appliquée dans le cas où un sinistre serait provoqué par un tiers conducteur.

– L'âge du conducteur :

Les personnes titulaires d'un permis récent, particulièrement les jeunes conducteurs, provoquent plus d'accidents que la moyenne des conducteurs.

– Le passé du conducteur :

Les assureurs ont le droit d'imposer des majorations de cotisation selon certains cas fixés par des textes règlementaires.

Comportement du conducteur	Majoration maximale
– Assuré responsable d'un accident alors qu'il conduisait sous l'emprise d'alcool	– 150%
– Assuré responsable d'un accident, ou coupable d'une infraction, ayant entraîné la suspension ou l'annulation du permis de conduire : – Suspension de deux à six mois – Suspension de plus de six mois – Annulation ou plusieurs suspensions de plus de deux mois au cours de la même période annuelle de référence (la période annuelle de référence est la période annuelle précédant de deux mois la date d'échéance principale)	– 50% – 100% – 200%
– Délit de fuite après accident	– 100%
– Non déclaration des accidents ou des circonstances aggravantes précipitées ou des accidents dont ils ont été responsables au cours des trois années précédentes	– 100%
– Fréquence d'accidents anormale par rapport à la fréquence moyenne (il s'agit de trois accidents et plus au cours d'une période d'un an précédent de deux mois l'échéance annuelle du contrat)	– 100%

TABLE 2.3 – Tableau des majorations selon le comportement du conducteur

Les différentes majorations peuvent se cumuler sans toutefois dépasser 400%, et sont supprimées après deux années.

– Le type d'apprentissage de conduite :

Les personnes qui ont suivi un apprentissage anticipé de la conduite bénéficient de certains avantages tarifaires.

En effet, lors de sa première année de souscription, un jeune conducteur ayant suivi un apprentissage anticipé de la conduite bénéficie d'une réduction de 50% sur la surprime à verser, puis bénéficie à nouveau de 50% de réduction supplémentaires après la première année d'assurance s'il n'a été responsable d'aucun accident.

Majoration maximale	Apprentissage normal	Apprentissage anticipé
À la souscription	100%	50%
Après un an d'assurance sans accident	50%	25%
Après deux ans d'assurance sans accident	Plus de surprime	Plus de surprime

TABLE 2.4 – Tableau des majorations maximales dans le cas d'un apprentissage anticipé de la conduite

Chapitre 3

La base de données

Un travail important sur la base de données est nécessaire lors d'une étude de prédictions de montants de sinistres, notamment une bonne prise de connaissance des données, la relève de données aberrantes, le recodage de certaines variables, la conversion d'autres variables (permettant de rendre la base la plus manipulable possible), ou encore l'étude des statistiques descriptives.

3.1 Présentation de la base de données

La réalisation de nos modèles de prédiction du montant des sinistres a reposé sur l'étude d'une base de données comprenant les caractéristiques de 2837 sinistres automobiles. C'est un faible nombre de sinistres pour les modèles prédictifs soient significatifs, notamment en comparaison du nombre de variables : la base de données comprend 38 variables, dont 12 variables quantitatives et 26 variables qualitatives.

Dans la base de données initiale, on trouve les différentes variables suivantes :

- **NumSin** : le numéro du contrat du sinistre
- **NatSin** : la nature du sinistre
- **DateSurv** : la date de survenance du sinistre
- **DateOuv** : la date d'ouverture du contrat
- **DateClo** : la date de clôture du contrat
- **GarPr** : le type de garantie principale
- **EtatSin** : l'état du sinistre
- **NbGarSin** : le nombre de garanties qui interviennent
- **TauxResp** : le taux de responsabilité de l'assuré dans le sinistre
- **TypeEval** : le type d'évaluation du sinistre
- **ConvIDA** : la présence d'une convention IDA ou non
- **CasIDA** : le cas de la convention IDA lorsqu'elle est présente
- **ConvIRCA** : la présence d'une convention IRCA ou non
- **CodePostal** : le code postal de l'endroit dans lequel a eu lieu le sinistre
- **AnneeNai** : l'année de naissance de l'assuré
- **AnneePermis** : l'année d'obtention du permis de l'assuré
- **DateEffetContrat** : la date d'effet du contrat d'assurance
- **DateMiseEnCirculation** : la date de mise en circulation du véhicule
- **Formule** : la formule d'assurance utilisée
- **Usage** : le type d'usage du véhicule
- **CSP** : la catégorie socio-professionnelle de l'assuré
- **CRM** : le coefficient de réduction-majoration appliqué
- **NbAnneeCRM50** : le nombre d'années où le CRM valait 50%

- **CodeAuto** : le code automobile du véhicule
- **GroupeSRA** : la référence en sécurité et réparation automobile
- **ClasseDePrix** : la classe de prix du véhicule
- **Financement** : le type de financement
- **NbAnneeAss** : le nombre d'années depuis lequel l'assuré a souscrit le contrat
- **Segment** : le segment automobile ou catégorie de voiture
- **Carrosserie** : le type de carrosserie du véhicule
- **Energie** : le type d'énergie utilisée pour faire fonctionner le véhicule
- **TypeGarage** : le type d'endroit où est généralement conservé le véhicule lorsqu'il n'est pas utilisé
- **Protection** : la catégorie de protection du véhicule
- **FraisExp** : les frais d'expert relatifs au sinistre
- **MontantPaye** : le montant payé
- **MontantRecours** : le montant du recours
- **MontantTotal** : le montant total du sinistre

Il y a parmi toutes ces variables, une variable à expliquer qui est le montant total du sinistre. L'objectif de notre étude est de provisionner ce montant, que l'on calcule à partir des montants payé et recours :

$$MontantTotal = MontantPayé + MontantRecours \tag{3.1}$$

3.2 Modification de la base de données

3.2.1 Recodage des variables

La présence de valeurs aberrantes dans une base de données est toujours handicapante dans la réalisation de modèles prédictifs, c'est pourquoi il convient de les traiter. Dans la base de données à notre disposition, nous avons pu relever comme valeurs aberrantes :

- Des CRM évalués à -999, que l'on a recodés en CRM évalués à 0.
- Une valeur manquante (*NA*) pour le taux de responsabilité, que l'on a remplacée par un taux de responsabilité égal à 0.
- La présence de quatre garanties pour un même sinistre, que nous avons ramenées à seulement trois garanties.
- Pour la variable ConvIRCA, on sait seulement que certains sinistres font intervenir une convention IRCA, pour tous les sinistres où nous n'avons pas d'information, nous considérons qu'il n'y en a pas.
- Pour toutes les autres variables présentant des valeurs manquantes, nous avons créé un vecteur **"tropdeNA"** dans lequel est contenu le numéro des colonnes contenant les variables dont plus de 5% des modalités sont manquantes. De la sorte, nous gardons la totalité des données, mais dans les cas où ces variables pourraient avoir un fort impact, et par conséquent, pourraient remettre en cause les résultats obtenus, il nous sera possible de les retirer facilement.

3.2.2 Conversion de certaines variables

La base de données initiale contenait un certain nombre de dates, dates qu'il est difficile d'exploiter telles quelles. C'est pourquoi il a été plus judicieux de convertir ces dates en nombre de jours ou d'années selon la caractéristique concernée. Ces conversions concernent notamment la date de naissance du conducteur, que l'on a convertie en âge du conducteur, la date de mise en circulation du véhicule a

été convertie en ancienneté du véhicule, et avec les dates d'ouverture et de clôture du contrat, on a pu calculer l'ancienneté du contrat, ce qui a permis de faciliter la manipulation des données.

Par ailleurs, la localisation des sinistres était donnée au travers de codes postaux. Cette variable proposant un trop grand nombre de critères, il a été préférable de rassembler ces données en fonction des régions auxquelles les codes postaux correspondent, ceci réduisant alors le nombre de classes différentes (22 régions – ancienne répartition).

Les nouvelles variables crées sont alors :

- **Region** : la région dans laquelle le sinistre a eu lieu
- **Age** : l'âge de l'assuré
- **AncContrat** : l'ancienneté du contrat
- **AncVehicule** : l'ancienneté du véhicule

3.3 Quelques statistiques descriptives

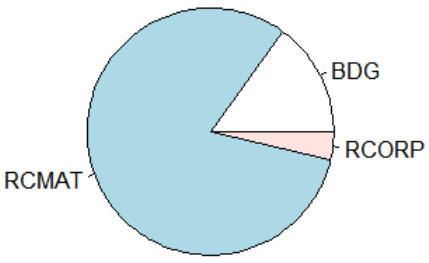
Avant de réaliser tout calcul, nous avons commencé par étudier les statistiques descriptives des différentes données. Cette étude est nécessaire et très importante car elle sert à fiabiliser la base de données, et permet de se familiariser avec la base ainsi que de relever les éventuelles données aberrantes supplémentaires qui ne sont pas forcément visibles lorsque que l'on étudie la base ligne à ligne.

Nous avons réalisé des statistiques descriptives pour chacune des variables de la base de données. Nous faisons le choix de présenter celles des variables les plus corrélées par rapport aux montant recours et payé.

Les caractéristiques ont été calculées par rapport au montant total des sinistres, puis nous avons fait le choix de les calculer par rapport aux montants recours et payé séparément, afin de voir si les résultats varient beaucoup d'un montant à un autre.

3.3.1 Pour le type de garantie principale

Il y a trois types de garantie principale qui interviennent : la garantie corporelle (RCORP), la garantie matérielle (RCMAT) et la garantie bris de glaces (BDG).



Modalité	BDG	RCMAT	RCORP
Effectif	427	2304	106

FIGURE 3.1 – Répartition des sinistres en fonction du type de garantie principale

Sur la FIGURE 3.1, on remarque une répartition inégale des types de garantie : il y a beaucoup moins de sinistres engageant les garanties corporelles que les garanties matérielles ou bris de glaces (seulement une centaine de garanties corporelles contre environ 2700 garanties matérielles et bris de glaces). Ceci s'explique par le fait que lorsqu'un sinistre automobile survient (accident de la route ou autre), il y a très

souvent des dégâts matériels, quelle que soit l'ampleur du sinistre ; tandis que des dégâts corporels sont généralement provoqués par des sinistres plus importants ou d'une plus grande violence.

– Montant total :

Les histogrammes suivants représentent les fréquences d'apparition de chaque type de garantie en fonction des montants totaux observés.

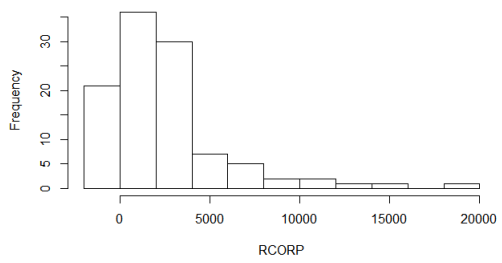


FIGURE 3.2 – Répartition des sinistres ayant une RCORP comme garantie principale en fonction du montant total observé

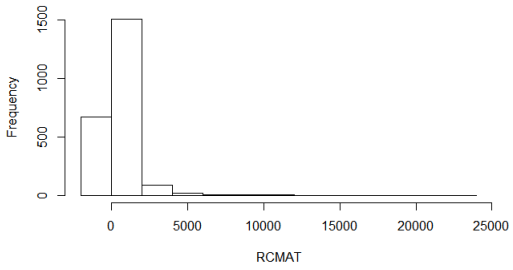


FIGURE 3.3 – Répartition des sinistres ayant une RCMAT comme garantie principale en fonction du montant total observé

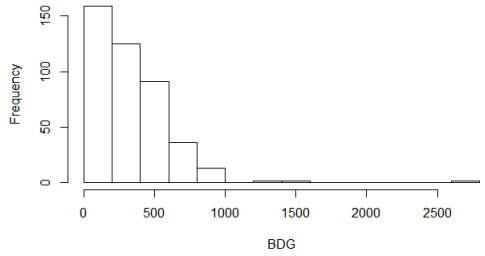


FIGURE 3.4 – Répartition des sinistres ayant une BDG comme garantie principale en fonction du montant total observé

valeur	moyenne	ecart type	min	max
BDG	318	70075	0	2766
RCMAT	845	1416636	-1611	22753
RCORP	2550	10549975	-691	19161
Total	829.610486429327	1701628.38110414	-1611	22753.51

FIGURE 3.5 – Statistiques sur les garanties principales à partir des montants totaux observés

On remarque que les montants totaux sont relativement plus élevés lorsque la garantie corporelle entre en jeu. En effet, d'après les statistiques de la FIGURE 3.5, les sinistres ayant pour garantie principale une RCORP ont un montant total moyen de 2550€, contre 845€ pour les sinistres ayant pour garantie principale une RCMAT ou encore un peu moins de 320€ pour une garantie principale BDG.

– Montant payé :

Les histogrammes suivants représentent les fréquences d'apparition de chaque type de garantie en fonction des montants payés observés.

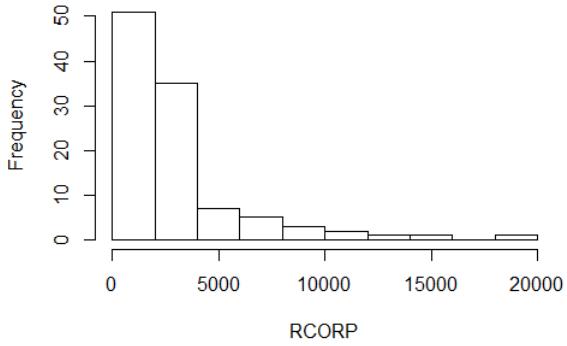


FIGURE 3.6 – Répartition des sinistres ayant une RCORP comme garantie principale en fonction du montant payé observé

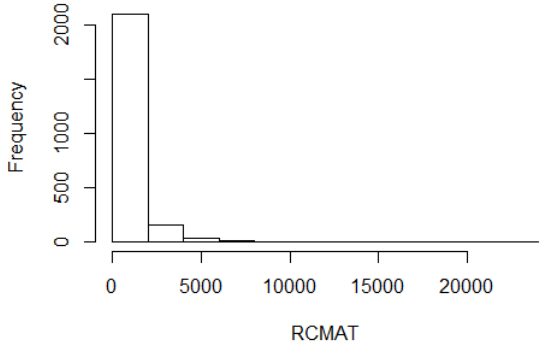


FIGURE 3.7 – Répartition des sinistres ayant une RCMAT comme garantie principale en fonction du montant payé observé

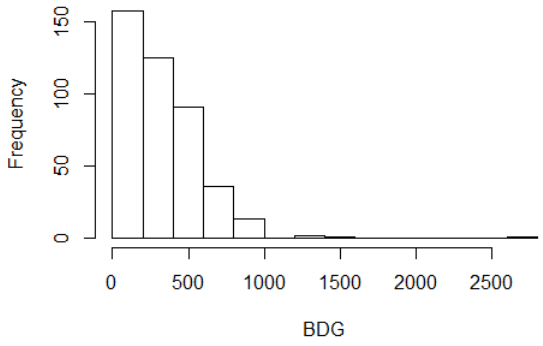


FIGURE 3.8 – Répartition des sinistres ayant une BDG comme garantie principale en fonction du montant payé observé

valeur	moyenne	ecart type	min	max
BDG	321	72256	0	2766
RCMAT	1219	1242966	0	22753
RCORP	2803	10180513	0	19161
Total	1143.23001057455	1606440.85218473	0	22753.51

FIGURE 3.9 – Statistiques sur les garanties principales à partir des montants payés observés

On remarque que les montants payés sont plutôt faibles dans les cas de sinistres matériels (85% de ces sinistres sous la barre des 2000€), tandis que ceux des sinistres corporels sont beaucoup plus étendus (variance plus grande), et beaucoup plus important (plus de 50% des sinistres corporels au-dessus de 2000€).

– Montant recours :

Les histogrammes suivants représentent les fréquences d'apparition de chaque type de garantie en fonction des montants recours observés.

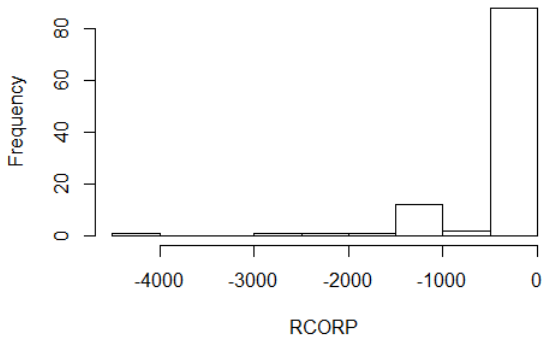


FIGURE 3.10 – Répartition des sinistres ayant une RCORP comme garantie principale en fonction du montant recours observé

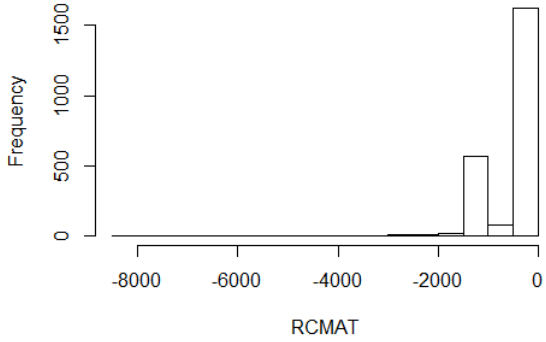


FIGURE 3.11 – Répartition des sinistres ayant une RCMAT comme garantie principale en fonction du montant recours observé

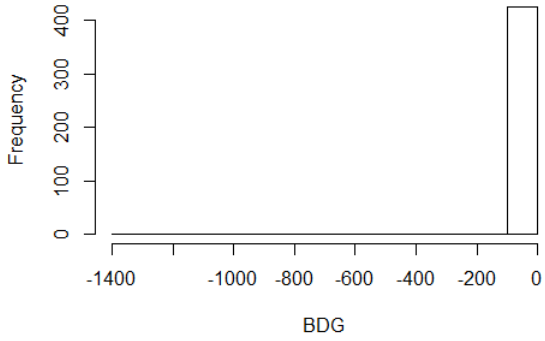


FIGURE 3.12 – Répartition des sinistres ayant une BDG comme garantie principale en fonction du montant recours observé

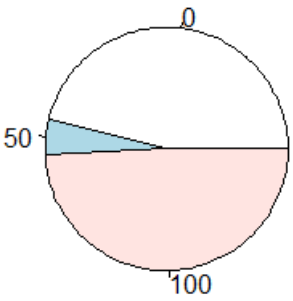
valeur	moyenne	ecart type	min	max
BDG	-4	4181	-1337	0
RCMAT	-376	416242	-8134	0
RCORP	-266	462328	-4312	0
Total	-315.515534014804	373489.835754554	-8133.98	0

FIGURE 3.13 – Statistiques sur les garanties principales à partir des montants recours observés

On remarque que les montants recours perçus dans le cas de l'intervention d'une garantie corporelle sont plus élevés que dans le cas où c'est une garantie matérielle qui intervient. On observe par ailleurs que le montant recours est nul dans le cas de simple bris de glace. Ceci peut s'expliquer par le fait que généralement, les frais engendrés par un sinistre bris de glace sont moins élevés que ceux engendrés par un sinistre corporel et nombreux sinistres matériels.

3.3.2 Pour le taux de responsabilité

Il y a trois niveaux de taux de responsabilité : 0%, 50% ou 100%, correspondants alors au niveau de responsabilité de chaque assuré dans les sinistre causés.



Modalité	0%	50%	100%
Effectif	1307	132	1397

FIGURE 3.14 – Répartition des sinistres en fonction du taux de responsabilité de l'assuré

Sur la FIGURE 3.14, on remarque une répartition inégale des taux de responsabilité : il y a quasiment 50% des sinistres dont le taux de responsabilité de l'assuré est de 0% et 50% des sinistres dont le taux de responsabilité de l'assuré est de 100% (un peu plus de 1300 assurés concernés dans chaque cas), tandis qu'une très faible proportion des assurés a un taux de responsabilité de 50% dans le sinistre (seulement une centaines d'assurés dans ce cas).

– Montant total :

Les histogrammes suivants représentent les fréquences d'apparition de chaque taux de responsabilité en fonction des montants totaux observés.

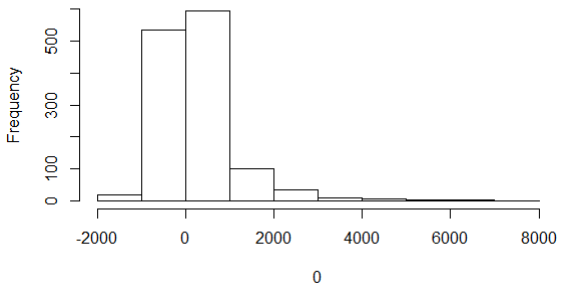


FIGURE 3.15 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 0% en fonction du montant total observé

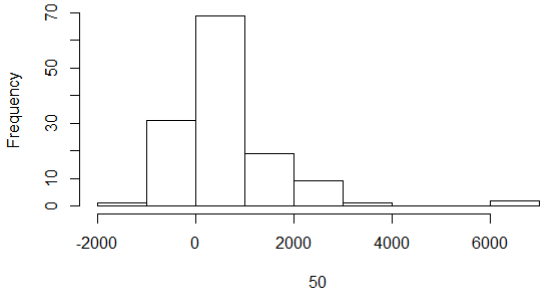


FIGURE 3.16 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 50% en fonction du montant total observé

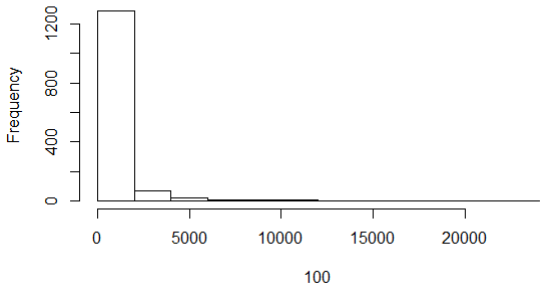


FIGURE 3.17 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 100% en fonction du montant total observé

valeur	moyenne	ecart type	min	max
0	257	763364	-1201	7350
50	692	1089270	-1611	6685
100	1377	2030860	0	22753
Total	829.610486429327	1701628.38110414	-1611	22753.51

FIGURE 3.18 – Statistiques sur les taux de responsabilité à partir des montants totaux observés

On remarque que les montants sont relativement plus élevés lorsque le taux de responsabilité de l'assuré est de 100%. En effet, d'après les statistiques de la FIGURE 3.18, les sinistres pour lesquels l'assuré a un taux de responsabilité de 100% ont un montant total moyen de plus de 1370€, contre un peu moins de 700€ pour les sinistres pour lesquels l'assuré a un taux de responsabilité de 50%, ou encore un peu moins de 270€ pour un taux de responsabilité nul.

– Montant payé :

Les histogrammes suivants représentent les fréquences d'apparition de chaque taux de responsabilité en fonction des montants payés observés.

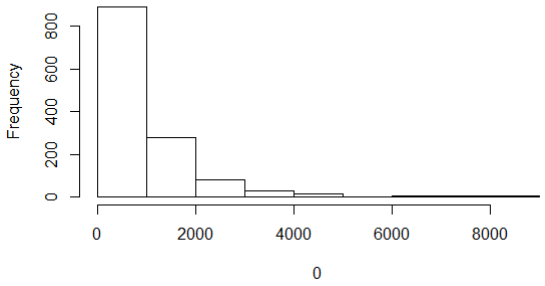


FIGURE 3.19 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 0% en fonction du montant payé observé

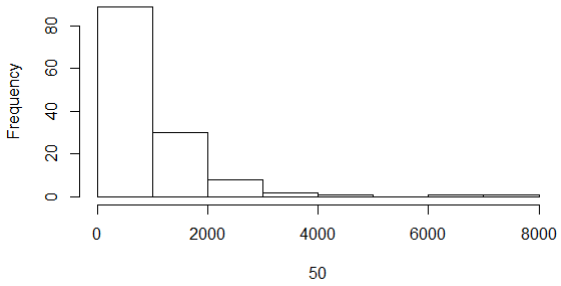


FIGURE 3.20 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 50% en fonction du montant payé observé

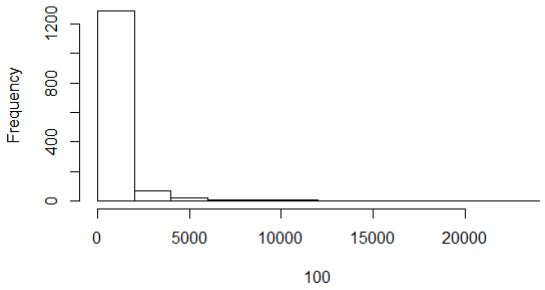


FIGURE 3.21 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 100% en fonction du montant payé observé

valeur	moyenne	ecart type	min	max
0	906	1074981	0	8566
50	972	1086003	0	7306
100	1380	2043513	0	22753
Total	1143.23001057455	1606440.85218473	0	22753.51

FIGURE 3.22 – Statistiques sur les taux de responsabilité à partir des montants payés observés

De même que pour les montants totaux, on remarque que les montants payés sont plus élevés dans les cas de sinistres pour lesquels l'assuré a un taux de responsabilité de 100% (prix payé moyen de 1300€), tandis que ceux des sinistres dont l'assuré n'est pas responsable ou responsable à seulement 50% sont à peu près équivalents plus faibles (prix payés moyen de 900€).

– Montant recours :

Les histogrammes suivants représentent les fréquences d'apparition de chaque taux de responsabilité en fonction des montants recours observés.

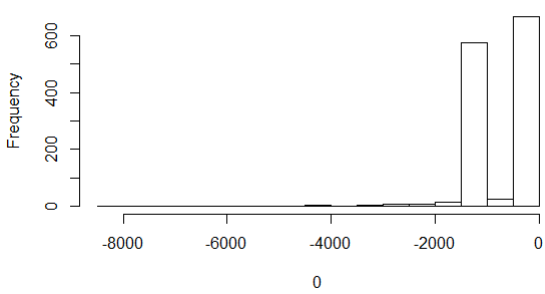


FIGURE 3.23 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 0% en fonction du montant recours observé

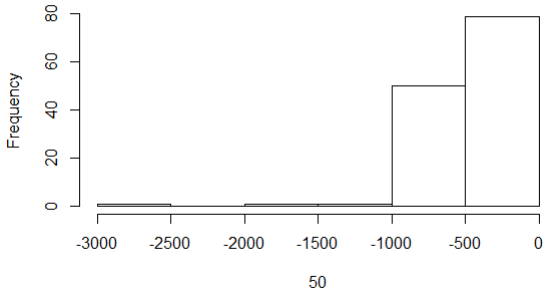


FIGURE 3.24 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 50% en fonction du montant recours observé

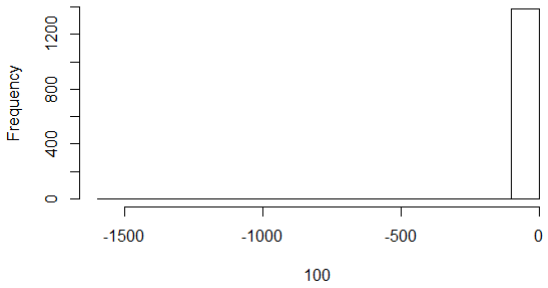


FIGURE 3.25 – Répartition des sinistres pour lesquels l'assuré a un taux de responsabilité de 100% en fonction du montant recours observé

valeur	moyenne	ecart type	min	max
0	-652	573410	-8134	0
50	-283	163322	-2700	0
100	-5	4203	-1519	0
Total	-315.515534014804	373489.835754554	-8133.98	0

FIGURE 3.26 – Statistiques sur les taux de responsabilité à partir des montants recours observés

On remarque que les montants recours perçus dans le cas d'un sinistre pour lequel l'assuré est 100% responsable sont très rares (en moyenne ils sont de seulement 5€), tandis que presque la totalité des sinistres pour lesquels les assurés ont un taux de responsabilité nul ou égal à 50% engendrent des montants plus élevés (pouvant aller parfois jusqu'à 2700€ pour un taux de responsabilité de 50%, ou près de 8000€ pour un taux de responsabilité nul). On observe par ailleurs que les montants recours sont beaucoup plus étendus pour les sinistres dont l'assuré n'est pas responsable (variance très élevée). Ceci peut s'interpréter avec le fait que généralement, les frais engendrés par un sinistre dont l'assuré est responsable sont plus élevés pour l'assureur, et il est plus difficile à l'assureur de faire une demande de recours dans le cas où son assuré est totalement responsable du sinistre causé.

3.4 Base d'apprentissage - Base de test

Après toute cette étude sur la base de données, pour commencer à mettre en place des modèles prédictifs de la sinistralité, nous avons dû procéder à un retraitement de la base, notamment en créant une base d'apprentissage ainsi qu'une base de test. Nous avons choisi un découpage selon une répartition de données 80% pour la base d'apprentissage, 20% pour la base de test.

La première base essentielle à la réalisation de nos modèles est la base d'apprentissage. Elle contient un portefeuille d'assurés avec des variables explicatives liées à l'assuré (telles que son âge, sa catégorie socio-professionnelle,...) et au véhicule (catégorie du véhicule, ancienneté,...), ainsi qu'une variable à expliquer relevant de la sinistralité liée à l'assuré (montant total, obtenu à partir des montants recours et payé). C'est à partir de cette base là que seront réalisés tous les modèles qui suivront.

Ces modèles ainsi créés devront alors être exécutés sur une seconde base, que l'on appelle base de test. Cette base contient alors un autre portefeuille d'assurés. C'est sur ce portefeuille que seront estimées les prédictions. C'est donc en utilisant les méthodes, modèles et relations établies entre les différentes variables explicatives et à expliquer de la base d'apprentissage sur les variables de la base de test que l'on pourra prédire le montant de la sinistralité lié à chacun des assurés du portefeuille de la base de test. Ce qui entraine en quelque sorte, une comparaison entre les assurés de la base de test et les assurés de la base d'apprentissage, du fait que les prédictions de sinistres seront établies à partir des profils des assurés du portefeuille de la base de test.

Chapitre 4

Un premier modèle simple

4.1 Présentation du modèle

4.1.1 Principe du modèle

Dans un premier temps, il semblerait intéressant de réaliser un premier modèle simple. Ce modèle serait basé sur une méthode prenant en compte les moyennes des montants payés et recours sur des classes de variables les plus corrélées à chacun des montants à prédire.

Nous rappelons que l'on cherche à prédire les montants payés et les montants recours séparément puisque différents groupes de variables expliquent mieux chaque montant.

4.1.2 Objectif du modèle

Ce modèle a pour objectif d'obtenir une table telle que pour chaque combinaison de critères créée, il y ait un montant qui corresponde. Ce montant sera alors implémenté dans la base de données, et les tests seront réalisés à partir de ces montants moyens obtenus.

4.2 Recherche des variables les plus corrélées

La première étape de ce modèle simple consiste à trouver quelles sont les variables les plus corrélées aux montants à prédire, c'est-à-dire au montant payé et au montant recours.

Pour cela, nous avons réalisé une AOV (Analysis Of Variance) sur chacune des variables explicatives.

À partir de la dernière colonne renvoyée par l'AOV, on obtient alors parmi les variables les plus corrélées pour le montant payé : la garantie principale (Bris de glace - BDG, Matérielle - RCMAT, Corporelle - RCORP), le nombre de garanties qui interviennent (de 1 à 4), le taux de responsabilité (0%, 50%, 100%), et la formule (tiers, intermédiaire, tous risques) ; quant au montant du recours, on obtient comme variables les plus corrélées : la garantie principale, le taux de responsabilité, et la convention IDA [1] (présente ou non).

1. **Convention IDA** : Convention établie entre assureurs, qui régit le dédommagement rapide des sinistres automobiles (IDA=Indemnisation Directe de l'Assuré). L'assureur de responsabilité civile rembourse directement les dommages subis par le véhicule de son propre assuré, à la place du conducteur responsable.

```
> summary(fitMP)
              Df    Sum Sq   Mean Sq F value   Pr(>F)
NatSin         1 3.302e+08 330235356 292.145  < 2e-16 ***
DateSurv       1 9.747e+05    974719   0.862   0.3534
DelaiDecla     1 4.905e+06   4904515   4.339   0.0376 *
DureeOuvClo    1 1.753e+06   1752663   1.551   0.2134
GarPr          2 3.007e+08 150360356 133.017  < 2e-16 ***
EtatSin        2 4.740e+08 236994828 209.659  < 2e-16 ***
NbGarSin       1 7.104e+07  71039864  62.846 7.41e-15 ***
TauxResp       1 5.038e+06   5037815   4.457   0.0351 *
TypeEval       1 2.581e+06   2580627   2.283   0.1312
ConvIDA        1 7.142e+07  71420973  63.183 6.32e-15 ***
Region        20 2.818e+07   1408803   1.246   0.2083
Age            1 1.094e+05    109354   0.097   0.7559
NbAnneePermis  1 2.332e+05    233230   0.206   0.6498
AncContrat     1 1.164e+06   1164022   1.030   0.3105
AncVehicule    1 2.806e+07  28060574  24.824 7.68e-07 ***
Formule        2 9.666e+06   4833019   4.276   0.0142 *
Usage          2 2.225e+06   1112421   0.984   0.3742
CSP           11 1.937e+07   1760695   1.558   0.1064
CRM            1 1.390e+05    138984   0.123   0.7259
NbAnneeCRM50  26 1.713e+07    658685   0.583   0.9529
CodeAuto    1941 2.268e+09   1168415   1.034   0.2913
NbAnneeAss     1 4.260e+05    425950   0.377   0.5395
TypeGarage     4 2.014e+06    503602   0.446   0.7757
FraisExp       1 8.129e+05    812911   0.719   0.3967
Residuals    809 9.145e+08   1130383
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

FIGURE 4.1 – AOV obtenue pour le montant payé

```
> summary(fitMR)
              Df    Sum Sq   Mean Sq  F value   Pr(>F)
NatSin         1    470910    470910    3.246 0.071958 .
DateSurv       1    979939    979939    6.755 0.009517 **
DelaiDecla     1   8011595   8011595   55.229 2.73e-13 ***
DureeOuvClo    1   4772098   4772098   32.897 1.37e-08 ***
GarPr          2  69632846  34816423  240.011  < 2e-16 ***
EtatSin        2  25145724  12572862   86.672  < 2e-16 ***
NbGarSin       1   1324937   1324937    9.134 0.002589 **
TauxResp       1 498041860 498041860 3433.307  < 2e-16 ***
TypeEval       1     76601     76601    0.528 0.467635
ConvIDA        1  10670986  10670986   73.562  < 2e-16 ***
Region        20   3110585    155529    1.072 0.374031
Age            1      1815      1815    0.013 0.910954
NbAnneePermis  1       500       500    0.003 0.953189
AncContrat     1     33798     33798    0.233 0.629446
AncVehicule    1   1928673   1928673   13.296 0.000283 ***
Formule        2   1167243    583621    4.023 0.018254 *
Usage          2     73633     36817    0.254 0.775909
CSP           11   3624147    329468    2.271 0.009944 **
CRM            1    148729    148729    1.025 0.311572
NbAnneeCRM50  26   2604919    100189    0.691 0.874931
CodeAuto    1941 307237159    158288    1.091 0.072560 .
NbAnneeAss     1    334802    334802    2.308 0.129101
TypeGarage     4   1997995    499499    3.443 0.008410 **
FraisExp       1    271412    271412    1.871 0.171739
Residuals    809 117355025    145062
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

FIGURE 4.2 – AOV obtenue pour le montant recours

Cependant, l'AOV nécessite des hypothèses sur la distribution de la variable à expliquer qui n'étaient pas vérifiées dans notre cas :

- La normalité de la distribution de la variable à expliquer pour chaque classe
- L'égalité de la variance de la variable à expliquer pour chaque classe

Il convenait donc de vérifier ces hypothèses. Les différents tests statistiques nous ont permis de réfuter ces hypothèses : les résultats de nos AOV ne pouvaient donc pas être exploités.

Nous avons alors eu recours au rapport de corrélation.

Le rapport de corrélation, noté η^2 est un indicateur statistique mesurant l'intensité de la liaison entre une variable qualitative et une variable quantitative. Il se calcule à l'aide des variances intraclasses et interclasses. Il s'agit du rapport de la variance interclasses sur la variance totale. Ce rapport est compris entre 0 et 1, et plus il est élevé, plus la variable quantitative est expliquée par la variable qualitative.

Nous avons commencé par sélectionner uniquement les variables quantitatives ayant relativement peu de classes : en effet, utiliser une variable continue ou avec beaucoup de modalités, aurait contribué à obtenir un modèle surement plus précis, mais également trop complexe.

À partir des tableaux de résultats suivants, nous avons, de ce fait, retenu neuf variables : **NatSin**, **GarPr**, **NbGarSin**, **TauxResp**, **ConvIDA**, **Formule**, **Usage**, **Energie** et **TypeGarage**. Nous avons alors utilisé le package **BioStatR**, qui permet, via la fonction ***eta2***, de calculer le rapport de corrélation.

Pour le montant payé, nous avons alors retenu les variables **GarPr**, **TauxResp**, **NbGarSin** et **Formule**. Nous n'avons pas retenu la variable **NatSin** car elle n'apporte rien de plus que la variable **GarPr** (le type de garantie principale est lié à la nature du sinistre).

Pour le montant recours, nous avons retenu les variables **GarPr**, **ConvIDA** et **TauxResp**.

Nous remarquerons que les variables retenues sont celles que nous aurions retenues si notre AOV avait été valable.

```
$NatSin
[1] 0.07251411

$GarPr
[1] 0.1302889

$NbGarSin
[1] 0.0802021

$TauxResp
[1] 0.03418409

$ConvIDA
[1] 0.01568018

$Formule
[1] 0.02028357

$Usage
[1] 0.0001332414

$Energie
[1] 0.0007935346

$TypeGarage
[1] 0.003617573
```

FIGURE 4.3 – Rapports de corrélation avec le montant payé

```
$NatSin
[1] 0.0004416246

$GarPr
[1] 0.0474727

$NbGarSin
[1] 0.005357668

$TauxResp
[1] 0.2670379

$ConvIDA
[1] 0.04573049

$Formule
[1] 0.007653484

$Usage
[1] 9.51699e-05

$Energie
[1] 0.003160792

$TypeGarage
[1] 0.0009889498
```

FIGURE 4.4 – Rapports de corrélation avec le montant recours

4.3 Regroupement des sinistres en classes

La deuxième étape dans la construction de notre modèle simple est la création de classes de sinistres à partir des variables les plus corrélées trouvées précédemment, tout en veillant à respecter le critère de Cochran. Ce critère a pour principe de faire en sorte que chaque classe formée contienne au moins 5% de la population totale contenue dans la base, afin de veiller à ce que le nombre de classes ne soit pas trop élevé, ce qui éviterait alors d'avoir des montants moyens qui ne sont présents qu'un faible nombre de fois et qui pourraient influer négativement les résultats.

Pour respecter ce critère, on vérifie la proportion de sinistres contenue dans chaque classe, et les classes ne respectant pas les 5% désirés sont rassemblées afin de créer de nouvelles classes comprenant cette fois-ci la bonne proportion de sinistres.

– Montant payé :

GarPr	BDG	→	1
	RCMAT	→	2
	RCORP	→	3
TauxResp	0%	→	1
	50%	→	2
	100%	→	3
Formule	Intermédiaire	→	1
	Tiers	→	2
	Tous Risques	→	3

TABLE 4.1 – Tableau de conversion des modalités des variables pour le montant payé[2]

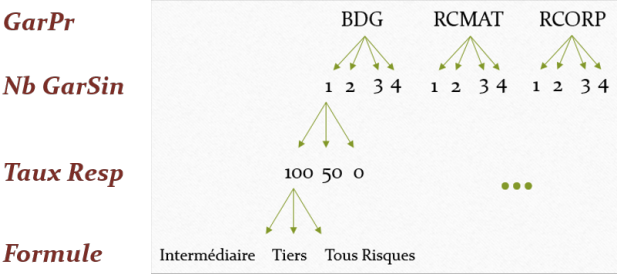


FIGURE 4.5 – Principe de formation des classes pour le montant payé

Exemple : un sinistre qui appartient à la classe 2131 a pour garantie principale une garantie matérielle, une seule garantie qui intervient, un taux de responsabilité de 100%, ainsi qu'une formule intermédiaire.

2. Pour la variable NbGarSin, on garde les modalités déjà renseignées (entiers de 1 à 4)

On obtient de la sorte sept classes de sinistres pour les montants payés :

1111	2111	2112	1131	2132	4441	4442

TABLE 4.2 – Différentes classes créées pour les montants payés

Les classes 4441 et 4442 sont les classes créées à partir de celles qui avaient un effectif inférieur à 5%. On peut afficher leur composition :

1112	1113	2113	2121	2122

TABLE 4.3 – Composition de la classe 4441

2123	2133	2211	2212	2213	2221	2222	2223	2231	2232
2233	2312	2331	2332	2333	3211	3212	3213	3221	3222
3231	3232	3233	3311	3312	3313	3331	3332	3333	3432

TABLE 4.4 – Composition de la classe 4442

On rajoute donc une colonne "ClasseMP" à la fin de notre base de données qui contient les différentes classes des sinistres créées à partir des variables corrélées avec le montant payé.

– Montant recours :

GarPr	BDG	→	1
	RCMAT	→	2
	RCORP	→	3
TauxResp	0%	→	1
	50%	→	2
	100%	→	3
ConvIDA	N	→	1
	O	→	2

TABLE 4.5 – Tableau de conversion des modalités des variables pour le montant recours

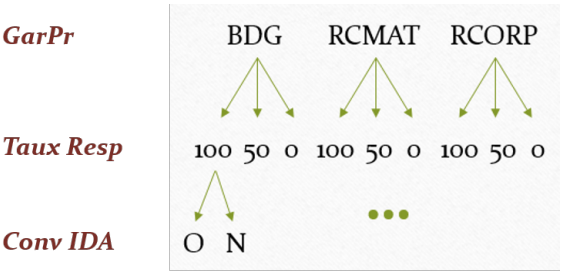


FIGURE 4.6 – Principe de formation des classes pour le montant recours

Exemple : un sinistre qui appartient à la classe 2131 a pour garantie principale une garantie matérielle, une seule garantie qui intervient, un taux de responsabilité de 100%, ainsi qu'une formule intermédiaire.

On obtient de la sorte six classes de sinistres pour les montants recours :

111	211	212	231	232	444

TABLE 4.6 – Différentes classes créées pour les montants recours

La classe 444 est créée à partir de celles qui avaient un effectif inférieur à 5%. On peut afficher leur composition :

213	221	222	311	312	321	322	331	332

TABLE 4.7 – Composition de la classe 444

On rajoute donc une colonne "ClasseMR" à la fin de notre base de données qui contient les différentes classes des sinistres créées à partir des variables corrélées avec le montant recours.

4.4 Calcul des montants à prédire moyens pour chaque classe

La prochaine étape à réaliser dans notre modèle simple est le calcul des montants moyens.

Avant toute chose, on sépare les données en une base d'apprentissage et une base de test

4.4.1 Calcul des montants payés moyens

Sur la base d'apprentissage, on crée deux vecteurs contenant respectivement les montants payés des sinistres, et les classes formées à partir du montant payé.

On calcule alors la moyenne des montants payés par rapport aux classes de sinistres à l'aide de la fonction ***tapply*** du logiciel R. Les montants ainsi calculés sont ensuite implémentés dans la base de test, de sorte à « remplacer » les montants initialement renseignés.

On affiche les montants payés moyens obtenus par classe :

V1	MoyMP
1111	319.453079470199
2111	1141.84851851852
2112	1127.5831300813
2131	1187.46482269504
2132	1232.69679471789
4441	805.367592592593
4442	2244.94254901961

FIGURE 4.7 – Montants payés moyens obtenus par classe

4.4.2 Calcul des montants recours moyens

De la même manière, on crée deux vecteurs contenant respectivement les montants recours des sinistres et les classes formées à partir du montant recours ; puis on calcule la moyenne des montants recours par rapport aux classes de sinistres, que l'on implémente également dans la base de test.

On affiche les montants recours moyens obtenus par classe :

V1	MoyMR
111	0
211	-509.568287671233
212	-1108.65588785047
231	-4.18934911242604
232	-4.02805836139169
444	-304.433555555556

FIGURE 4.8 – Montants recours moyens obtenus par classe

Avant de poursuivre les calculs, il convient de vérifier que les classes et montants de la base de test sont tous compris dans la base d'apprentissage. C'est bien notre cas ici (dans le cas contraire, nous aurions eu à traiter les classes absentes de la base d'apprentissage).

4.4.3 Calcul des montants totaux moyens des sinistres

A partir des montants payés et recours moyens obtenus lors des deux étapes précédentes, on calcule les montants totaux moyens des sinistres sur les deux bases (apprentissage et test).

– Base d'apprentissage :

X	MontantPaye	MontantRecours	MontantTotal	ClasseMP	ClasseMR
754	319.4531	0.000000	319.45308	1111	111
1056	1232.6968	-4.028058	1228.66874	2132	232
1625	1232.6968	-4.028058	1228.66874	2132	232
2574	1141.8485	-509.568288	632.28023	2111	211
572	1232.6968	-4.028058	1228.66874	2132	232
2545	1127.5831	-1108.655888	18.92724	2112	212
2675	1232.6968	-4.028058	1228.66874	2132	232
1871	2244.9425	-4.189349	2240.75320	4442	231
1780	1232.6968	-4.028058	1228.66874	2132	232
175	805.3676	0.000000	805.36759	4441	111
583	1232.6968	-4.028058	1228.66874	2132	232
499	805.3676	-304.433556	500.93404	4441	444
1941	319.4531	0.000000	319.45308	1111	111

FIGURE 4.9 – Montants moyens obtenus dans la base d'apprentissage

– Base de test :

X	MontantPaye	MontantRecours	MontantTotal	ClasseMP	ClasseMR
14	2244.9425	-4.189349	2240.75320	4442	231
15	805.3676	-304.433556	500.93404	4441	444
16	1187.4648	-4.189349	1183.27547	2131	231
22	805.3676	-304.433556	500.93404	4441	444
32	1232.6968	-4.028058	1228.66874	2132	232
37	1232.6968	-4.028058	1228.66874	2132	232
39	319.4531	0.000000	319.45308	1111	111
41	805.3676	0.000000	805.36759	4441	111
44	2244.9425	-304.433556	1940.50899	4442	444
45	1187.4648	-4.028058	1183.43676	2131	232
48	1232.6968	-4.028058	1228.66874	2132	232
51	805.3676	-304.433556	500.93404	4441	444
55	1232.6968	-4.028058	1228.66874	2132	232

FIGURE 4.10 – Montants moyens de la base de test

4.5 Evaluation de la qualité du modèle

4.5.1 Présentation des différentes métriques d'erreur

Pour chacun de nos modèles, nous allons calculer plusieurs métriques d'erreurs, afin d'avoir plusieurs moyens de comparaison.

4.5.1.1 Root Mean Square Error (RMSE)

La première métrique d'erreur que nous avons calculée est la RMSE (Root Mean Square Error), qui correspond à la racine carrée de l'erreur quadratique moyenne entre les montants réels et prédits des sinistres.

Elle se traduit par la formule suivante :

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}}$$

avec :
- n = Nombre de sinistres
- x_i = Montant prédit pour le sinistre i
- y_i = Montant réel du sinistre i

4.5.1.2 Norme Euclidienne

La seconde métrique que nous avons choisie de calculer est la norme Euclienne, qui correspond à la racine carrée de la somme du carré des distances séparant les montants réels et prédits des sinistres.

Elle se traduit par la formule suivante :

$$Err_{Euclid} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

avec :
- n = Nombre de sinistres
- x_i = Montant prédit pour le sinistre i
- y_i = Montant réel du sinistre i

4.5.1.3 Erreur absolue moyenne (MAE)

La troisième métrique que nous avons choisie de calculer est la MAE (Mean Absolute Error), qui est une quantité qui permet de mesurer la proximité des prévisions ou des prédictions avec les résultats éventuels.

Son expression est donnée par :

$$MAE = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n}$$

avec :
- n = Nombre de sinistres
- x_i = Montant prédit pour le sinistre i
- y_i = Montant réel du sinistre i

4.5.1.4 Moyenne obtenue sur les 10% pires prédictions

La dernière métrique choisie est une métrique qui consiste à calculer la moyenne des montants des sinistres sur les 10% pires prédictions, afin d'avoir un certain aperçu sur les écarts extrêmes des montants prédits par rapport aux montants réels des sinistres.

On supposant $v = |x - y|$, avec x un vecteur contenant les prédictions des montants des sinistres et y un vecteur contenant les montants réels des sinistres ; et k un vecteur contenant les composantes de v rangées dans un ordre décroissant. En considérant n le nombre de sinistres, on a donc $k = (k_1, k_2, \ldots, k_n)$.

On peut alors calculer $l = \left\lfloor \frac{n}{10} \right\rfloor$, où l représente 10% du nombre de sinistres.

On calcule alors la moyenne des 10% pires prédictions à l'aide de l'expression suivante :

$$Err = \sum_{i=1}^{l} \frac{k_i}{l}$$

4.5.2 Résultats obtenus

On obtient comme résultats pour les différentes métriques d'erreur :

- Base d'apprentissage :

	RMSE	Err Euclid	MAE	10%	Err Relative
Montant Payé	1171.4658	55801.6249	536.9652	2557.1663	47.08%
Montant Recours	423.1069	20154.2852	154.1676	968.9636	48.14%
Montant Total	1148.2070	54693.7185	529.142	2495.5684	54.83%

TABLE 4.8 – Erreurs obtenues pour la base d'apprentissage

Afin de comprendre un peu mieux ces résultats, nous les comparerons par la suite avec ceux obtenus pour les autres modèles de machine learning.

- Base de test :

	RMSE	Err Euclid	MAE	10%	Err Relative
Montant Payé	1199.0349	28576.2998	535.3188	2615.1021	46.40%
Montant Recours	334.3218	7967.8090	158.3160	907.1200	53.37%
Montant Total	1208.2866	28796.7929	547.5963	2691.0594	54.82%

TABLE 4.9 – Erreurs obtenues pour la base de test

Afin de comprendre un peu mieux ces résultats, nous les comparerons par la suite avec ceux obtenus pour les autres modèles de machine learning.

En comparant les erreurs calculées pour les montants totaux sur la base de test avec celles calculées sur la base d'apprentissage, on observe que les montants sont à peu près du même ordre de grandeur pour chacune des erreurs calculées, à l'exception de la norme euclidienne, qui est presque doublée dans le cas de la base d'apprentissage.

4.5.3 Calcul des métriques d'erreur par classe

Il nous est également paru intéressant de calculer les différentes métriques d'erreurs sur chacune des classes créées, afin d'avoir un aperçu plus détaillé de la performance de notre modèle simple.

Dans un premier temps, il est intéressant de connaître les différentes proportions de sinistres contenues dans chaque classe, ce qui permettra une meilleure interprétation des résultats :

– Pour le montant payé :

Classe	1111	2111	2112	2131	2132	4441	4442
Proportion de sinistres	13%	6%	22%	7%	36%	7%	9%

TABLE 4.10 – Proportions de sinistres par classe pour le montant payé

– Pour le montant recours :

Classe	111	211	212	231	232	444
Proportion de sinistres	15%	7%	23%	7%	40%	8%

TABLE 4.11 – Proportions de sinistres par classe pour le montant recours

Dans un second temps, il convient de calculer les erreurs par classe. Nous nous concentrons ici seulement sur les résultats obtenus pour la base de test.

– Pour le montant payé :

Classe	**RMSE**	**Err Euclid**	**MAE**	**10%**	**Err Relative**
1111	217.1465	1723.5472	162.4580	468.7669	51.12%
2111	1044.5227	5009.3547	670.8768	2944.2715	52.23%
2112	843.9768	9696.5552	664.1778	1740.8983	66.99%
2131	609.9818	4313.2229	306.8463	1492.5789	27.88%
2132	754.2304	10666.4289	235.8794	1803.6197	19.15%
4441	1176.4372	8235.0606	705.3829	3411.7999	81.90%
4442	3113.3006	22233.4131	1836.1298	7663.2655	71.35%

TABLE 4.12 – Erreurs obtenues sur les montants payés prédits pour chaque classe sur la base de test

On remarque des différences de montants assez importantes d'après les erreurs calculées sur les montants payés prédits pour chacune des classes, notamment pour les classes recomposées 4441 et 4442, pour lesquelles les RMSE, MAE et moyennes obtenus sur les 10% pires prédictions sont considérablement supérieures à celles calculées pour les autres classes de sinistres ; ceci se constate également à travers les erreurs relatives (le modèle atteint plus de 80% et 70% d'erreur sur les prédictions des montants payés pour ces deux classes). À l'inverse, les classes 1111 (Garantie principale BDG – 1 garantie – Taux de responsabilité 0% - Formule intermédiaire) et 2131 (Garantie principale RCMAT – 1 garantie – Taux de

responsabilité 100% - Formule intermédiaire) sont les classes dont les erreurs sont les plus faibles. En revanche, on constate plus de 50% d'erreur sur la classe 1111 du fait que le montant moyen prédit est assez faible (de l'ordre de 320€), mais seulement 19.15% d'erreur de prédiction pour la classe 2132, c'est donc cette dernière pour qui les prédictions sont les plus proches de la réalité.

– Pour le montant recours :

Classe	RMSE	Err Euclid	MAE	10%	Err Relative
111	150.3342	1336.2000	16.9139	190.8857	100%
211	588.8540	3770.5059	524.5833	1146.2117	156.40%
212	535.4542	6057.9725	352.8955	1231.7446	32.09%
231	244.5707	1404.9515	64.1361	663.6036	106.09%
232	4.0281	61.6174	4.0281	4.0281	∞
444	407.6421	2967.6791	355.6548	826.9664	177.33%

TABLE 4.13 – Erreurs obtenues sur les montants recours prédits pour chaque classe sur la base de test

De même que pour les montants payés, on remarque des différences de montants assez importantes d'après les erreurs calculées sur les montants recours prédits pour chacune des classes de sinistres, notamment pour les classes 211 et 212 (Garantie principale RCMAT – Taux de responsabilité 0%) qui ont des erreurs plus que doublées par rapport à celles calculées pour les autres classes de sinistres (à l'exception de la classe recomposée 444 pour qui les erreurs se rapprochent de celles des classes 211 et 212). Néanmoins au vu des erreurs relatives calculées, les prédictions pour la classe 212 restent raisonnables à côté de celles des classes 211 et 444 où le taux d'erreur dépasse largement les 150%. On remarque par ailleurs que les erreurs obtenues pour la classe 232 (Garantie principale RCMAT – Taux de responsabilité 100% - Pas de convention IDA) sont de l'ordre de grandeur 10€, ce qui est extrêmement faible. Ceci pourrait s'expliquer par le fait que cette classe regroupe presque 40% de la totalité des sinistres, et que la très grande majorité de ces sinistres a un montant recours nul (les classes 111, 231 et 232 ont près de 99.6% de leurs sinistres qui engendrent des montants recours nuls), toutefois, on a un taux d'erreur qui tend vers l'infini du fait que la totalité des sinistres de cette classe contenus dans la base de test a un montant recours nul.

4.5.4 Application de la validation croisée sur notre modèle simple

4.5.4.1 Principe de la validation croisée

La validation croisée est une méthode de test permettant de pallier au sur-apprentissage sur la base de test. En effet, lorsque l'on sépare la base en deux (80% pour la base d'apprentissage et 20% pour la base de test), le modèle peut sur-apprendre sur la base d'apprentissage et ne pas être bien calibré pour la base de test. Afin d'éviter ce sur-apprentissage, nous appliquons cette méthode sur chacun de nos modèles.

Cette technique consiste à séparer la base totale en n bases de parts égales. Nous sélectionnons $(n-1)$ bases pour former la base d'apprentissage, la dernière servira de base de test. Cette méthode est répétée n fois en changeant à chaque fois la base de test.

Nous obtenons donc n modèles calibrés sur presque toute la base, et nous calculons les erreurs globales car chaque base de test est en théorie trop petite pour la validation du modèle.

Le schéma ci-dessous est un schéma explicatif de cette technique avec $n = 4$:

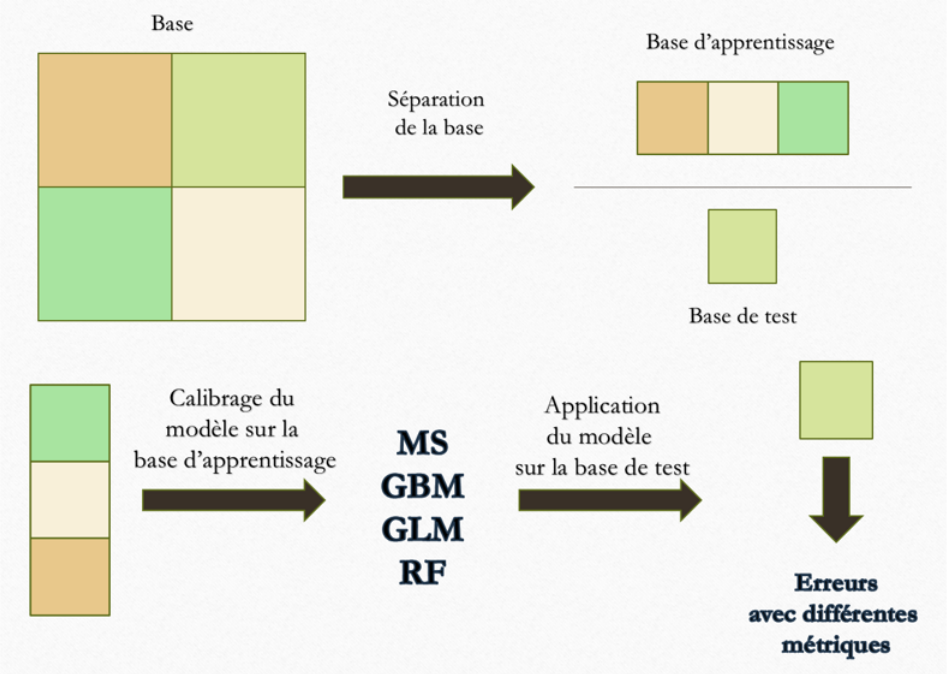


FIGURE 4.11 – Schéma explicatif du principe de la validation croisée

4.5.4.2 Application sur notre modèle simple

Pour appliquer la validation croisée sur notre modèle simple, on choisit $n = 10$.

Nous pouvons retrouver le détail des erreurs calculées pour chacun des croisements réalisés pour le montant payé et le montant recours à prédire en annexes A.1 et A.2.

Afin d'avoir une idée plus pertinente de la qualité du modèle, nous représentons au travers de diagrammes en boîte, la répartition des montants réels des sinistres en fonction des montants moyens calculés par classe.

Après exécution de la validation croisée sur notre modèle simple, il est possible d'établir le nombre moyen de sinistres ainsi que les montants payés et recours prédits moyens des sinistres par classe, à partir des résultats obtenus pour chacun des n croisements effectués :

– Pour le montant payé :

Classe	1111	2111	2112	2131	2132	4441	4442
Montant payé moyen (en €)	325	1185	1112	1158	1254	821	2300

TABLE 4.14 – Montants payés moyens par classe après application de la validation croisée

– Pour le montant recours :

Classe	111	211	212	231	232	444
Montant recours moyen (en €)	-3.81	-499	-1107	-16	-3.31	-260

TABLE 4.15 – Montants recours moyens par classe après application de la validation croisée

Concernant les classes 111, 231 et 232, il convient de préciser que près de 99.6% des sinistres qu'elles comportent ont des montant recours nuls (d'où les très faibles montants recours moyens obtenus).

Les graphiques ci-dessous représentent la répartition des montants réels en fonction des montants prédits moyens par classe :

– Pour le montant payé :

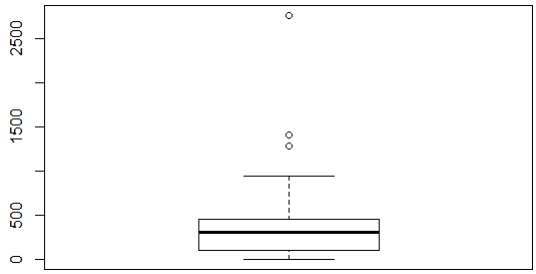


FIGURE 4.12 – Classe 1111

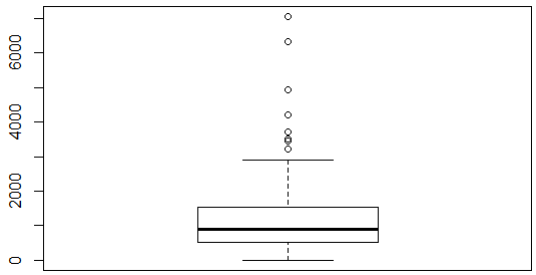


FIGURE 4.13 – Classe 2111

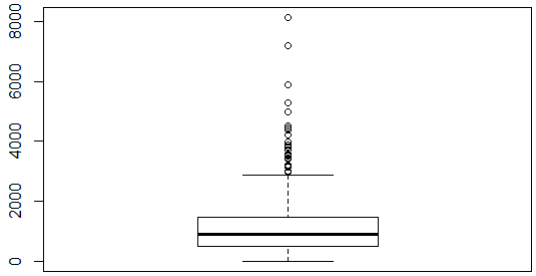


FIGURE 4.14 – Classe 2112

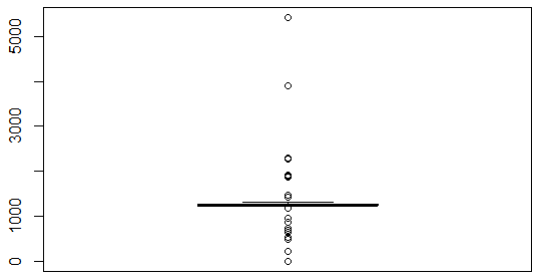


FIGURE 4.15 – Classe 2131

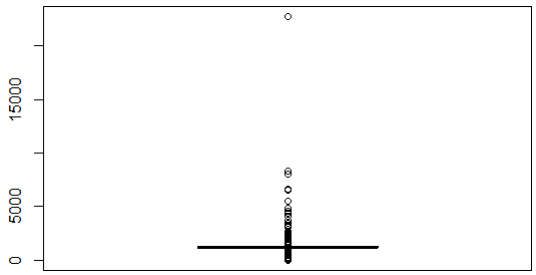


FIGURE 4.16 – Classe 2132

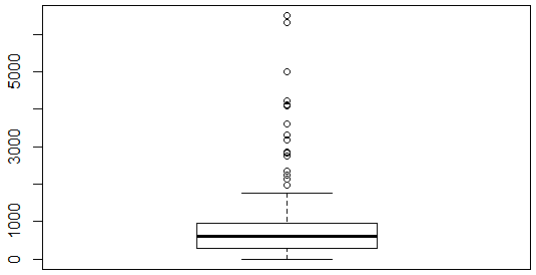


FIGURE 4.17 – Classe 4441

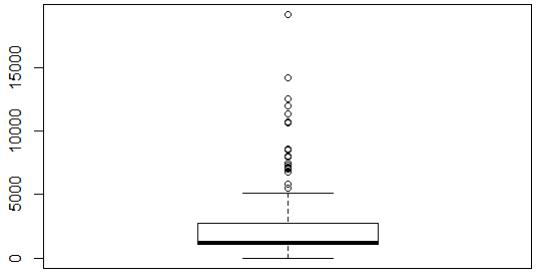


FIGURE 4.18 – Classe 4442

– Pour le montant recours :

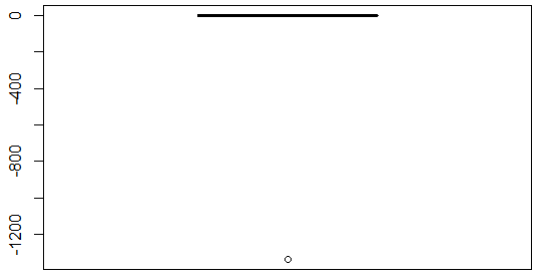


FIGURE 4.19 – Classe 111

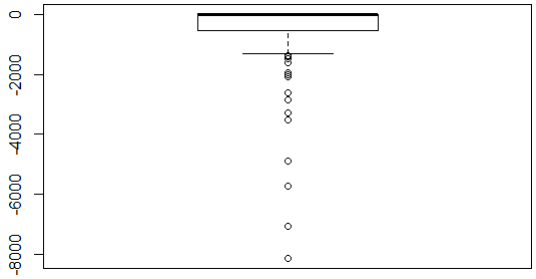


FIGURE 4.20 – Classe 211

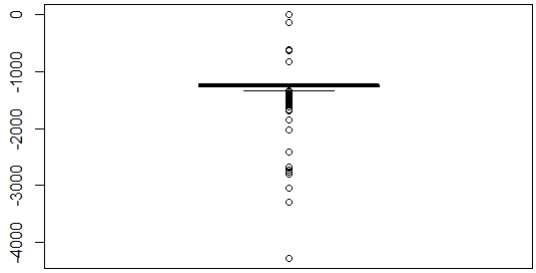


FIGURE 4.21 – Classe 212

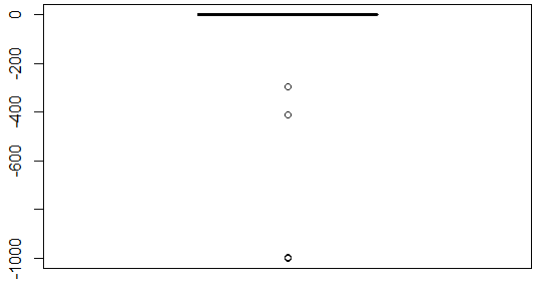


FIGURE 4.22 – Classe 231

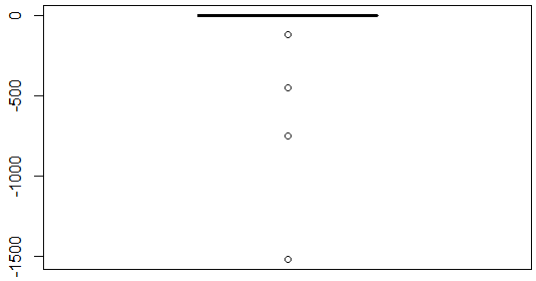


FIGURE 4.23 – Classe 232

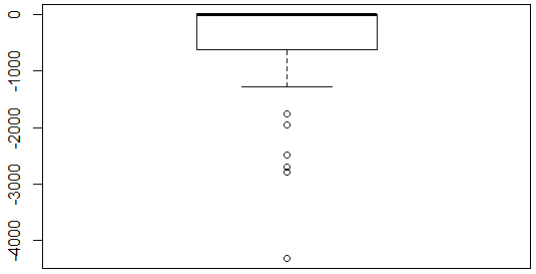


FIGURE 4.24 – Classe 444

Le modèle étant basé sur la moyenne des montants des sinistres, les valeurs extrêmes régissent nos modèles. On peut le voir sur toutes les simulations, que ce soit pour le montant recours ou pour le montant payé. Notons quand même que les valeurs simulées sont quand même bien regroupées autour de la moyenne.

En conclusion, ce modèle nous semble optimisé. Cependant nous rappelons sa simplicité. En effet la variable à estimer est une variable quantitative et le regroupement en classes discrétise le montant estimé. Par conséquent, il y a un écart irréductible entre les montants simulés et les montants réels.

Chapitre 5

Préparation à la programmation

5.1 Utilisation du package H2O

H2O est un logiciel libre conçu pour l'analyse de données massives, distribué par *H2O.ai*. Il peut être utilisé avec le logiciel R, ainsi que sur d'autres environnements (tels que Python ou Java, entre autres). Le logiciel R seul est bien souvent inefficace pour l'étude de larges bases de données : c'est dans ce contexte que l'utilisation de H2O, de par sa capacité à paralléliser les opérations, se révèle être la plus utile. Nous pourrons également mentionner la possibilité qu'offre H20 d'utiliser des clusters d'ordinateurs, afin d'augmenter encore la puissance de calcul.

H2O permet de réaliser diverses opérations sur les bases (échantillonnage) et de créer des modèles de Machine Learning, notamment :

- le Modèle Linéaire Généralisé (GLM)
- les forêts aléatoires
- le Gradient Boosting Machine (GBM)
- le Deep Learning (Réseau de neurones avec descente stochastique du gradient)

5.2 Normalisation de la base de données

L'étape de normalisation de la base de données est fondamentale avant d'utiliser les divers algorithmes de Machine Learning. En effet, sans elle, les variables n'auraient pas toutes eu la même importance, ce qui aurait faussé les résultats.

Nous avons normalisé les variables de deux façons différentes selon leur nature : quantitative ou qualitative.

- Pour les variables quantitatives comme l'âge ou l'ancienneté du véhicule, nous avons utilisé la transformation affine :

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}$$

  Ainsi, nous obtenons des valeurs comprises entre 0 (pour la valeur minimale de la variable x_i) et 1 (pour la valeur maximale).

- Pour les variables qualitatives, nous avons utilisé le package *Caret*.

  Pour une variable à n classes, celui-ci va créer n colonnes correspondantes, prenant comme valeur 1 si l'individu appartient à cette classe, 0 sinon.

On obtient alors une base normalisée, avec dorénavant 169 variables.

NbAnneeCRM50	NbAnneeAss	NatSin.CORP	NatSin.MAT	GarPr.BDG	GarPr.RCMAT	GarPr.RCORP
0.00	0.35483871	0	1	0	1	0
0.00	0.32258065	0	1	0	1	0
0.00	0.00000000	0	1	0	1	0
0.00	0.12903226	0	1	0	1	0
0.00	0.51612903	1	0	0	0	1
0.00	0.48387097	0	1	0	1	0
0.00	0.06451613	0	1	0	1	0
0.12	0.25806452	0	1	0	1	0
0.02	0.51612903	0	1	1	0	0
0.00	0.32258065	0	1	0	1	0
0.00	0.51612903	0	1	1	0	0
0.02	0.45161290	0	1	0	1	0
0.00	0.51612903	0	1	0	1	0
0.00	0.51612903	0	1	0	1	0
0.00	0.19354839	0	1	0	1	0

FIGURE 5.1 – Aperçu d'un morceau de la base de données après normalisation

5.3 Hyperparamétrage des modèles

Pour éviter le sur-apprentissage des modèles, il convient de réaliser un travail d'hyperparamétrage sur les modèles de forêts aléatoires et gradient boosting. Pour cela, il faut rechercher les hyperparamètres optimaux.

Le logiciel H2O utilisé sous R nous propose deux méthodes pour rechercher les meilleurs hyperparamètres :

– La méthode **Grid Search** :

C'est la façon classique de rechercher les hyperparamètres optimaux. Dans ce cas, on choisit un nombre fini de valeurs à tester pour chaque paramètre. L'algorithme va alors tester chaque combinaison possible.

Par exemple, dans le cas de forêts aléatoires, on peut choisir :

- **ntrees** $\in \{10, 20, 30\}$
- **max_depth** $\in \{5, 10, 15\}$
- **mtries** $\in \{50, 70, 80\}$

L'algorithme va alors créer 27 modèles.

– La méthode **Random Search** :

Dans ce cas-là, on va renseigner, pour chacun des paramètres, un intervalle dans lequel il sera choisi aléatoirement.

Par exemple, toujours dans le cas de forêts aléatoires, on peut choisir :

- **ntrees** $\in [10, 50]$
- **max_depth** $\in [5, 20]$
- **mtries** $\in [40, 80]$

On renseigne également un critère d'arrêt, par exemple que l'algorithme s'arrête après avoir tourné pendant deux heures ou bien après avoir créé 1000 modèles.

Dans les deux cas, l'algorithme nous renvoie la liste des modèles classés selon leur taux d'erreur (nous avons choisi la RMSE) sur la base d'apprentissage, toujours en utilisant la validation croisée (nfolds=5), afin d'éviter au maximum le sur-apprentissage.

Chapitre 6

Modèle Linéaire Généralisé (Generalized Linear Model - GLM)

6.1 Présentation du modèle

6.1.1 Présentation générale

Le Modèle Linéaire Généralisé a été formulé par John Nelder et Robert Wedderburn en 1972 comme moyen d'unifier les autres modèles statistiques. Il peut être considéré comme une extension de la régression multiple linéaire pour une seule variable explicative.

Rappelons que le but général de la régression multiple est de quantifier la relation entre plusieurs variables explicatives indépendantes et une variable à expliquer dépendante.

Le modèle GLM généralise la régression linéaire en permettant au modèle linéaire d'être relié à la variable à expliquer via une fonction lien, et en autorisant l'amplitude de la variance de chaque mesure d'être une fonction de sa valeur prévue, cela en utilisant les méthodes des moindres carrés du modèle linéaire général pour estimer et tester les hypothèses.

6.1.2 Le modèle GLM

Modèle : La variable à prédire est à valeurs dans $\mathbb{R}$

Un modèle GLM suppose :

- $(Y_1, ...Y_n)$ n variables aléatoires indépendantes
- On suppose que la loi de Y_i appartient à une famille de distributions avec des paramètres qui dépendent des variables explicatives à travers une fonction lien.

  Les modèles classiques sont :
  - Le Modèle GLM Gaussien : $Y_i \sim \mathcal{N}\big(\mu(x_{i,1}, ..., x_{i,p}), \sigma^2\big)$
  - Le Modèle GLM Bernoulli : $Y_i \sim \mathcal{B}er\big(\pi(x_{i,1}, ..., x_{i,p})\big)$
  - Le Modèle GLM Poisson : $Y_i \sim \mathcal{P}\big(\lambda(x_{i,1}, ..., x_{i,p})\big)$
  - Le Modèle GLM Gamma : $Y_i \sim Gam\big(\alpha, \beta(x_{i,1}, ..., x_{i,p})\big)$
- On choisit une fonction lien pour lier les variables explicatives aux paramètres de la distribution choisie.

  Le lien se fait au travers de l'espérance :

$$g\big(\mathbb{E}(Y_i)\big) = \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} \tag{6.1}$$

Les liens par défaut proposés par R sont :

- Pour le modèle GLM Gaussien : $g(x) = x$

On a alors $Y_i \sim \mathcal{N}\big(\mu(x_{i,1},...,x_{i,p}),\sigma^2\big)$ avec :

$$\begin{aligned} g\big(\mathbb{E}(Y_i)\big) &= \mathbb{E}(Y_i) && (6.2)\\ &= \mu(x_{i,1},...,x_{i,p}) && (6.3)\\ &= \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} && (6.4) \end{aligned}$$

- Pour le modèle GLM Bernoulli : $g(x) = logit(x) = ln\left(\frac{x}{1-x}\right)$

On a alors $Y_i \sim \mathcal{B}er\big(\pi(x_{i,1},...,x_{i,p})\big)$ avec :

$$\begin{aligned} g\big(\mathbb{E}(Y_i)\big) &= g\big(\pi(x_{i,1},...,x_{i,p})\big) && (6.5)\\ &= ln\left(\frac{\pi(x_{i,1},...,x_{i,p})}{1-\pi(x_{i,1},...,x_{i,p})}\right) && (6.6)\\ &= \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} && (6.7) \end{aligned}$$

Le modèle GLM Bernoulli avec lien $logit$ est appelé *modèle de régression logistique.*

- Pour le modèle GLM Poisson : $g(x) = ln(x)$

On a alors $Y_i \sim \mathcal{P}\big(\lambda(x_{i,1},...,x_{i,p})\big)$ avec :

$$\begin{aligned} g\big(\mathbb{E}(Y_i)\big) &= ln\big(\lambda(x_{i,1},...,x_{i,p})\big) && (6.8)\\ &= \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} && (6.9) \end{aligned}$$

On a alors $Y_i \sim \mathcal{P}\big(exp(\beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p})\big)$, on parle de *modèle log-poisson.*

- Pour le modèle GLM Gamma : $g(x) = \frac{1}{x}$

En tarification, on utilise plutôt : $g(x) = ln(x)$

On a alors $Y_i \sim Gam\big(\alpha,\beta(x_{i,1},...,x_{i,p})\big)$ avec :

$$\begin{aligned} g\big(\mathbb{E}(Y_i)\big) &= ln\big(\alpha\beta(x_{i,1},...,x_{i,p})\big) && (6.10)\\ &= \beta_0 + \beta_1 x_{i,1} + ... + \beta_p x_{i,p} && (6.11) \end{aligned}$$

Le modèle GLM Gamma avec lien ln est appelé *modèle log-gamma.*

6.1.3 Paramétrage du modèle

Comme pour chacun des autres modèles, l'utilisateur dispose d'un large choix de paramètres à renseigner pour le modèle GLM sous H2O. Nous pouvons par exemple citer :

- une distribution **family** : Gaussienne, Gamma, Poisson ou encore Binomiale. Dans notre cas, de par la présence de montants nuls ou négatifs, seule la distribution Gaussienne pouvait être utilisée et adaptée à notre problème.
- une fonction **link**
- **alpha** : paramètre permettant de pénaliser les variables explicatives supplémentaires.
- **lambda** : paramètre de régularisation permettant d'éviter le sur-apprentissage.

Ces différents paramètres peuvent être optimisés avec une méthode Grid Search ou une méthode Random Search. Nous utiliserons ici les paramètres par défaut du logiciel pour **alpha** et **lambda**.

6.2 Application à notre problème de tarification

6.2.1 Prévision du montant total

Afin de nous familiariser avec le modèle GLM implémenté dans H2O, nous avons réalisé un premier essai visant à prédire directement le montant total, avec les paramètres par défaut et en utilisant l'intégralité des variables. On obtient alors les résultats ci-dessous :

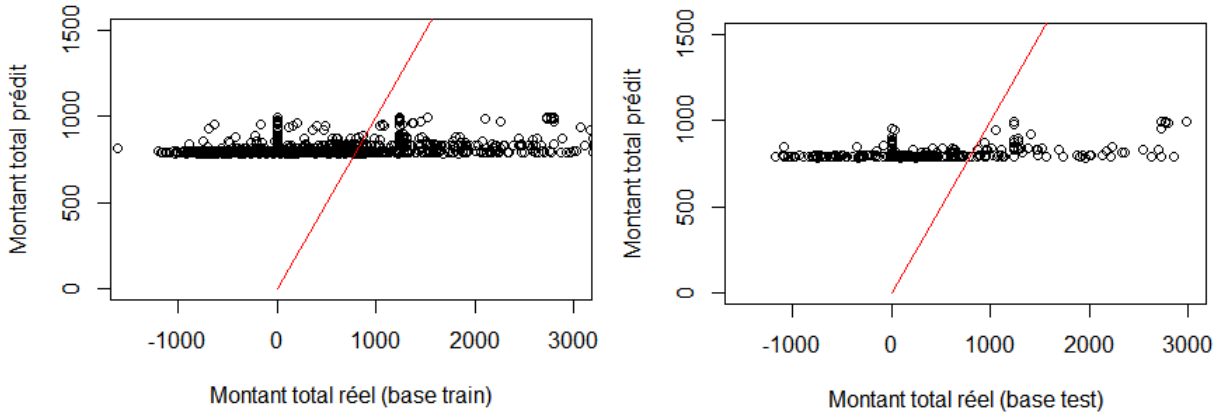


FIGURE 6.1 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases

Comme le montrent les deux graphiques ci-dessus, l'algorithme semble prédire une valeur sensiblement identique pour tous les sinistres (entre 785€ et 1063€).

Par ailleurs, on peut afficher le tableau des erreurs de prédiction de ce modèle :

	RMSE	**Err Euclid**	**MAE**	**10%**	**Err Relative**
Base d'apprentissage	1274.6519	60716.7976	761.7279	2601.8189	78.96%
Base de test	1330.7566	31715.5891	789.9932	2795.5739	79.09%

TABLE 6.1 – Erreurs obtenues pour le montant total

On comprend alors qu'un tel modèle n'est pas très efficace : ses performances sont en-dessous de celles du modèle simple (79% contre 55% pour le modèle simple).

6.2.2 Prédiction par sélection de variables

Le modèle précédent a pu être amélioré en ne retenant que les variables les plus importantes (grâce à l'utilisation de la fonction ***h2o.varimp***). Par exemple, en retenant 44 variables, on obtient les résultats ci-dessous :

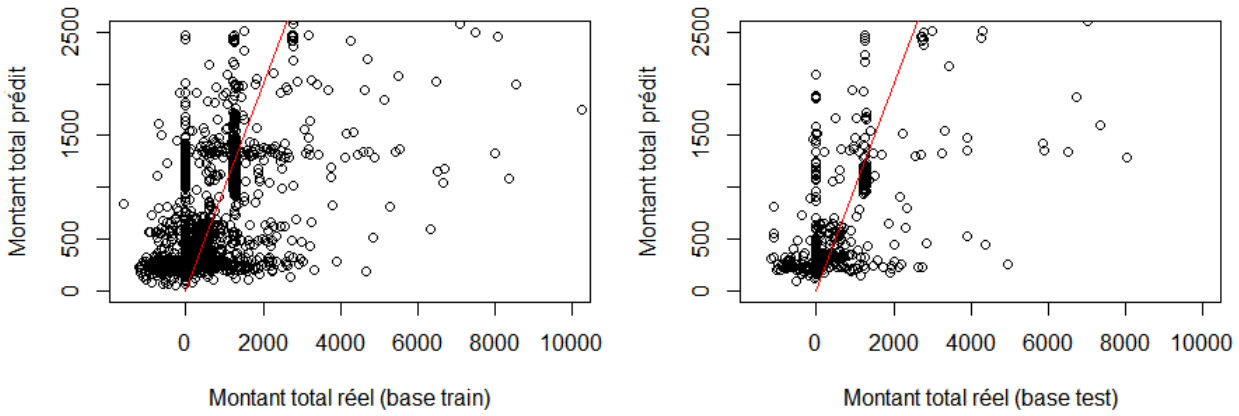


FIGURE 6.2 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases après amélioration

Graphiquement, les prédictions semblent meilleures ; cette impression est confirmée par le calcul des différentes erreurs :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	1117.0785	53210.9400	550.2983	2388.2405	57.04%
Base de test	1161.0666	27671.4103	561.5694	2505.125	56.22%

TABLE 6.2 – Erreurs obtenues pour le montant total après amélioration

Ces prédictions sont légèrement moins bonnes que celles du modèle simple (à 2% d'erreur près).

Par ailleurs, avec l'augmentation de la complexité du modèle, nous sommes alors en droit de nous demander s'il faut retenir ce modèle.

Chapitre 7

Arbres de décision et Forêts aléatoires

7.1 Les arbres de régression et de classification

7.1.1 Présentation du modèle

L'idée de base de ce modèle est simple : disposant de n variables explicatives X_1, X_2,..., X_n et d'une variable à expliquer Y (un réel dans le cas de la régression, une classe dans le cas de la classification), on cherche à prédire Y connaissant X_1, X_2,...,X_n. L'algorithme va alors créer un arbre binaire, qui va séparer la population en deux sous-populations à chaque nœud selon divers critères. Par exemple, l'algorithme va chercher à maximiser la variance interclasses dans le cas d'un arbre de régression.

A chaque nœud, l'algorithme va tester si l'une des conditions d'arrêt est vérifiée (profondeur maximale, nombre d'individus minimal, etc.). Si ce n'est pas le cas, l'algorithme sépare à nouveau la population en deux sous-populations. En revanche, si la condition est vérifiée, l'algorithme calcule la moyenne de la variable Y (ou la classe majoritaire dans le cas de la classification) pour les individus présents dans cette feuille. Ainsi, lorsque l'on voudra utiliser cet arbre pour prédire Y, les individus qui seront classés dans cette feuille se verront attribuer cette moyenne comme prédiction.

Les arbres de régression présentent de nombreux avantages : entre autres, ils sont simples à interpréter, rapides à mettre en place et assez économiques en temps de calcul.

Toutefois, ils sont très sensibles à l'échantillon d'apprentissage et peuvent avoir tendance à surapprendre. Ils ont alors un faible biais, mais une variance très grande. C'est pour réduire cette dernière que l'on utilise des forêts aléatoires. L'utilisation de forêts aléatoires s'accompagne d'une perte de lisibilité du modèle, mais d'une amélioration des performances.

7.1.2 Quelques exemples d'arbres de régression

En utilisant les paramètres par défaut du package *rpart* du logiciel R, on applique ce modèle à nos variables à expliquer (montant payé et montant recours), ce qui nous permet d'obtenir les deux arbres de régression suivants :

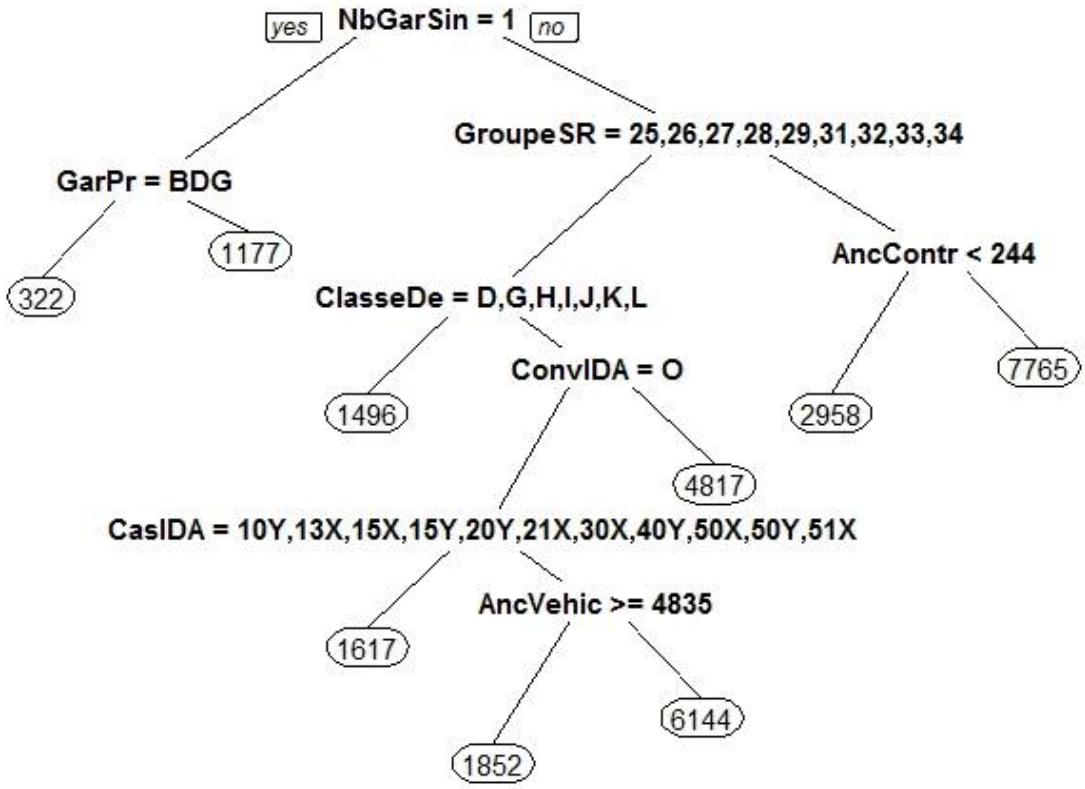


FIGURE 7.1 – Arbre de régression obtenu pour le montant payé

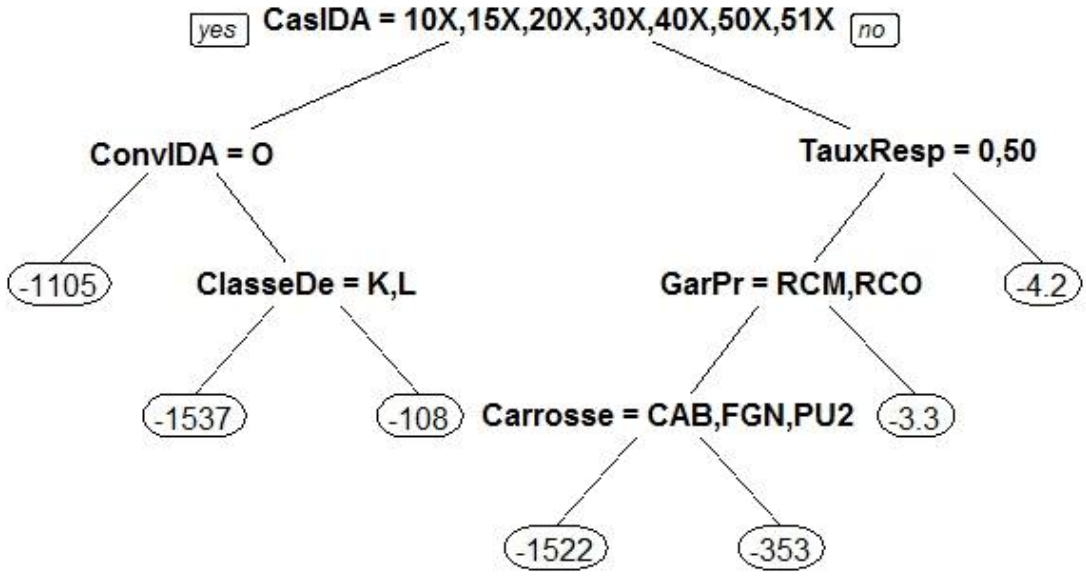


FIGURE 7.2 – Arbre de régression obtenu pour le montant recours

7.2 Les forêts aléatoires

7.2.1 Présentation du modèle

7.2.1.1 Bagging ou Boostrap Aggregating

Les forêts aléatoires utilisent la technique générale du **bagging**. Le bagging consiste à choisir K sous-ensembles de la base apprentissage avec remise, et à construire un arbre de décision sur chacun d'entre eux. Lors de la prédiction, on prendra ainsi la moyenne des prédictions retournées par chacun des K arbres.

Le **boostrap** permet de réduire la variance du modèle : ainsi, il est beaucoup moins sensible à l'échantillon d'apprentissage.

7.2.1.2 Forêts aléatoires

L'algorithme des forêts diffère de celui du bagging sur un seul point : le nombre de variables prises en compte à chaque nœud. Dans le bagging, à chaque nœud l'algorithme choisit parmi les p variables, celle qui permettra de réaliser la meilleure séparation selon le critère choisi (ex : maximisation de la variance interclasses). Avec une forêt, à chaque nœud, l'algorithme sélectionne aléatoirement q ($< p$) variables et choisit la meilleure d'entre elles.

Cela évite (notamment dans le cas où certaines variables sont très prédictives) d'obtenir des arbres trop corrélés, et donc une variance très importante.

Le choix de q est généralement de $\sqrt{q}$ pour la classification, et de $\frac{p}{3}$ (arrondis à l'entier inférieur) pour la régression.

7.2.2 Choix de l'hyperparamétrage

Afin de tenter d'optimiser l'hyperparamétrage, nous avons joué sur trois hyperparamètres :

- **ntrees** : le nombre d'arbres créés par l'algorithme
- **max_depth** : la profondeur maximale de chaque arbre
- **mtries** : le nombre de variables utilisées pour créer chaque arbre. L'algorithme sélectionnera alors q variables parmi les p variables totales. Dans le cas de la régression, on a en général $q \approx \frac{p}{3}$

Afin d'éviter au maximum un éventuel sur-apprentissage, on s'assure d'utiliser la validation croisée lors de nos Grid Search et Random Search.

7.2.3 Application à notre problème de tarification

7.2.3.1 Prévision du montant total

Là encore, nous avons commencé par un premier modèle simple, c'est-à-dire sans optimisation des hyperparamètres, visant à prédire le montant total des sinistres.

Les résultats obtenus sont les suivants :

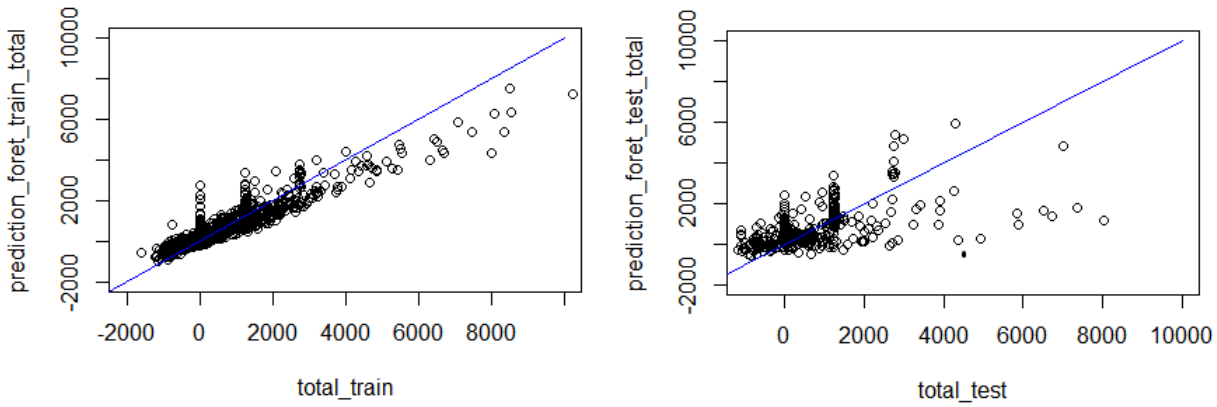


FIGURE 7.3 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases

On peut également afficher le tableau des erreurs de prédiction de ce modèle :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	445.5422	21699.3058	232.8688	1058.8727	24.14%
Base de test	1149.9449	27406.3509	553.4463	2551.3667	55.41%

TABLE 7.1 – Erreurs obtenues pour le montant total

Ce premier modèle nous donne des résultats bien meilleurs sur la base d'apprentissage, mais équivalents à ceux donnés par le modèle GLM et le modèle simple sur la base de test : il s'agit d'un problème de sur-apprentissage, malgré l'utilisation d'une validation croisée dans l'élaboration du modèle.

7.2.3.2 Utilisation de la méthode Random Search

Nous avons fait le choix d'utiliser la méthode Random Search pour trouver des hyperparamètres améliorant les prédictions. En laissant tourner l'algorithme plusieurs heures, nous avons obtenu près d'un millier de modèles. H2O les retourne selon leur RMSE calculée sur la base d'apprentissage.

Les hyperparamètres du meilleur modèle obtenu sont alors :

- ntrees = 20
- mtries = 50
- max_depth = 20

Les erreurs calculées sur ce modèle sont alors les suivantes :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	428.9296	20431.6431	203.6287	960.2615	21.10%
Base de test	1147.3456	27344.4020	551.1239	2518.1141	55.18%

TABLE 7.2 – Erreurs obtenues pour le montant total après la Random Search

Il est possible de remarquer que les résultats obtenus sont très légèrement meilleurs suite à la Random Search. On constate également que les performances de ce modèle sont meilleures que celles du GLM, mais moins bien que celles du modèle simple.

Chapitre 8

Modèle Gradient Boosting (GBM)

8.1 Présentation du modèle

8.1.1 Présentation générale

Comme tous les modèles de boosting, le modèle GBM s'appuie sur la combinaison de plusieurs modèles simples (en l'occurrence sur la combinaison de plusieurs arbres de décision), afin d'obtenir un modèle unique plus performant.

En règle générale, l'utilisation du modèle GBM permet d'obtenir de meilleurs résultats que l'utilisation des forêts aléatoires.

8.1.2 Schématisation du modèle

L'algorithme du modèle GBM (dans le cas d'une régression) peut se schématiser de la manière suivante :

1. On commence par choisir une fonction de coût différentiable. En général, on choit la fonction

$$G(x,y) = \frac{1}{2}(x-y)^2$$

   qui aura pour but d'estimer l'erreur entre les valeurs prédites par notre modèle et la variable à expliquer.
2. On initialise ensuite le modèle avec une valeur constante

$$\gamma_0 = arg\min_{\gamma\in\mathbb{R}}\Big(\sum_{i=1}^{n} G\big(y_i,\gamma_0\big)\Big)$$

   avec y_i la i-ème composante de la variable à expliquer, et n le nombre d'observations.
3. Pour b allant de 1 à B (ntrees) :
   - On note f_b le modèle obtenu à l'étape b
   - On calcule le résidu $y - f_{b-1}(x)$ qui s'apparente au gradient négatif de la fonction de coût :

$$y_i - f_b(x_i) = -\left[\frac{dG(y_i, f_{b-1}(x_i))}{df_{b-1}(x_i)}\right]$$

   - On crée un nouvel arbre M_b afin d'estimer $y_i - f_{b-1}(x_i)$

• On résout le problème d'optimisation :

$$\gamma_m = arg\min_{\gamma\in\mathbb{R}}\Big(\sum_{i=1}^{n} G\big(y_i, f_{b-1}(x_i)+\gamma M_b(x_i)\big)\Big)$$

Le nouveau modèle f_b s'écrit alors

$$f_b(x) = f_{b-1}(x) + \gamma_m M_b(x)$$

4. Finalement, l'algorithme renvoie le modèle

$$F(x) = \gamma_0 + \gamma_1 M_(x) + \ldots + M_B(x)$$

8.2 Choix de l'hyperparamétrage

Afin de tenter d'optimiser l'hyperparamétrage, nous avons ici joué sur trois hyperparamètres :

- **ntrees** : le nombre d'arbres créés par l'algorithme
- **max_depth** : la profondeur maximale de chaque arbre
- **learn_rate** : $(0 < LR < 1)$ le learn_rate représente le pourcentage de correction que va apporter un arbre supplémentaire à la prédiction.
  Par exemple, avec deux arbres, si le premier prédit 100€ et le second 150€, la correction est alors de 50€.
  Avec un learn_rate à 1, le montant prédit sera de $100 + 1 * 50 = 50$€.
  En revanche, avec un learn_rate à 0.1, le montant prédit sera de $100 + 0.1 * 50 = 105$€.

  Dans l'algorithme du modèle GBM donné ci-dessus, il était écrit :

$$f_b(x) = f_{b-1}(x) + \gamma_m M_b(x)$$

  Cela correspondait au cas où $LR = 1$.
  En règle générale, l'expression est en fait :

$$f_b(x) = f_{b-1}(x) + LR\gamma_m M_b(x)$$

  Empiriquement, il a été remarqué que des faibles valeurs $(LR < 0.1)$ permettent d'obtenir les meilleurs résultats.

Par ailleurs, nous avons utilisé la version stochastique du modèle GBM, dans l'objectif d'obtenir de meilleurs résultats. Pour cela, nous avons renseigné trois paramètres supplémentaires :

Ces trois paramètres doivent appartenir à l'intervalle $[0, 1]$: il s'agit de taux d'échantillonnage.

- **sample_rate** : le pourcentage d'observations utilisées pour créer chaque arbre
- **col_sample_rate_per_tree** : le pourcentage de variables retenues pour chaque arbre
- **col_sample_rate** : le pourcentage de variables retenues à chaque nouveau nœud

Il est recommandé de choisir ces paramètres entre 0.7 et 0.8 afin d'obtenir des résultats optimaux.

8.3 Application à notre problème de tarification

8.3.1 Modèle GBM simple

Une première implémentation simple du modèle GBM (sans optimisation des paramètres) nous retourne les résultats suivants :

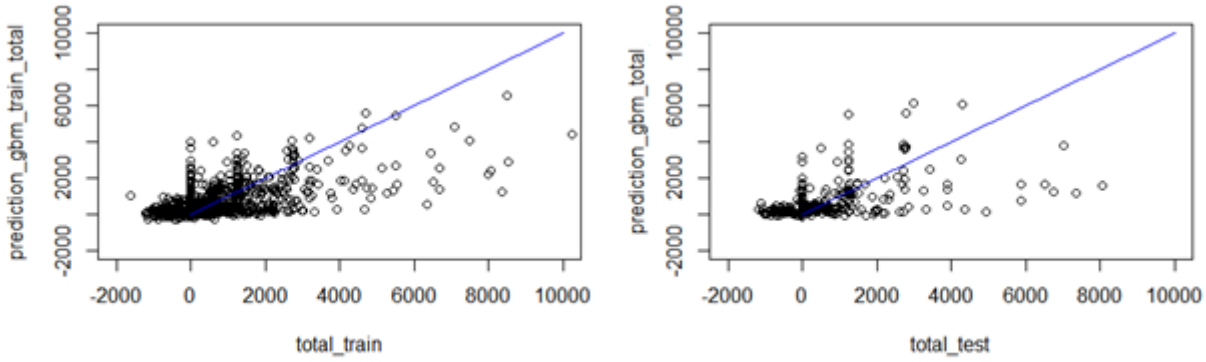


FIGURE 8.1 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases

On peut également afficher le tableau des erreurs de prédiction de ce modèle :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	990.7777	47194.7286	482.8517	2223.7383	50.05%
Base de test	1184.7259	28235.2767	557.0133	2756.9344	55.77%

TABLE 8.1 – Erreurs obtenues pour le montant total

Ce modèle GBM, un peu « naïf », retourne une prédiction moins bonne que les forêts aléatoires.

8.3.2 Optimisation du modèle GBM

8.3.2.1 Hyperparamétrage optimisé

De même que pour le modèle de forêts aléatoires, nous avons utilisé la méthode Random Search pour trouver des paramètres plus efficaces que ceux définis par défaut.

Les hyperparamètres du meilleur modèle obtenu sont alors :

- max_depth = 20
- learn_rate = 0.03
- On renseigne un nombre maximal d'arbres à réaliser (ntrees = 10000), et l'algorithme s'arrête lorsque la RMSE ne diminue plus significativement sur la base d'apprentissage.

Les erreurs calculées sur ce modèle sont alors les suivantes :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	867.8862	41340.9117	452.7496	1983.3898	46.93%
Base de test	1179.9446	28121.3245	552.2548	2610.1045	55.29%

TABLE 8.2 – Erreurs obtenues pour le montant total après la Random Search

On remarque que l'optimisation de l'hyperparamétrage du modèle permet d'obtenir des erreurs un peu plus faibles, et donc d'améliorer légèrement le modèle ; cependant, il reste moins performant que le modèle de forêts aléatoires.

8.3.2.2 Modèle GBM stochastique

Le modèle GBM peut, par ailleurs, être optimisé au travers de l'utilisation de sa variante stochastique. Pour cela, on introduit trois paramètres qui permettent à l'algorithme d'échantillonner aléatoirement des variables et des individus à chaque arbre.

Les paramètres optimaux pour le taux d'échantillonnage sont de l'ordre de 70% - 80% pour chacun des trois paramètres.

Les résultats obtenus sont alors :

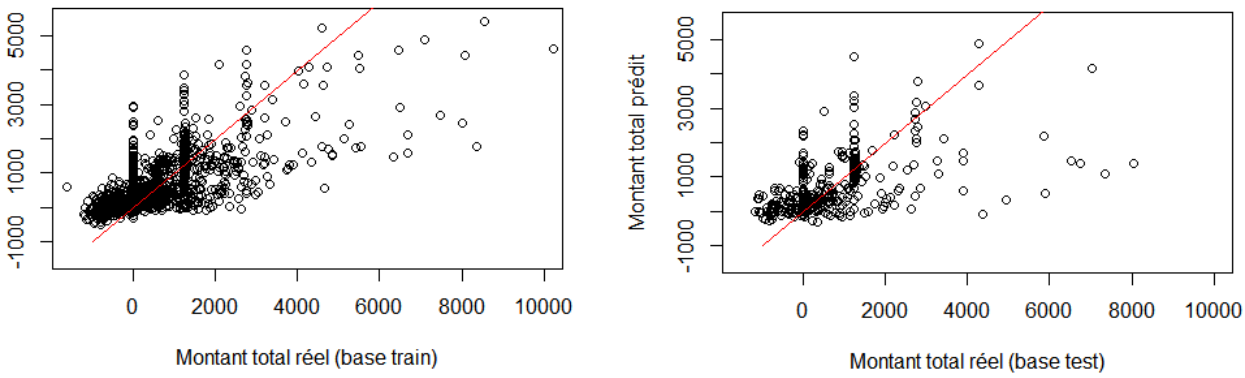


FIGURE 8.2 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases après optimisation stochastique

On peut également afficher le tableau des erreurs de prédiction de ce modèle :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	852.1083	40589.3448	450.1465	1944.0066	46.66%
Base de test	1171.9018	27929.6429	551.0315	2587.1616	55.17%

TABLE 8.3 – Erreurs obtenues pour le montant total après optimisation stochastique

On remarque que les résultats sur la base de test sont légèrement améliorés par l'utilisation du GBM stochastique, mais restent néanmoins équivalents ceux donnés par le modèle des forêts aléatoires.

Chapitre 9

Recherche des meilleurs modèles

La prédiction directe du montant total était la plus simple, elle nous a ainsi permis de nous familiariser avec l'environnement du logiciel H2O, ainsi que les différentes possibilités qu'il pouvait nous offrir.

Toutefois, au vu des résultats obtenus, il semble que cette méthode ne soit pas la plus efficace. Nous avons alors pensé à deux autres façons de prédire le montant total :

- Prédire le montant payé séparément du montant recours, puis faire la somme des deux afin d'obtenir une prédiction du montant total
- Comme alternative au modèle précédent, devant le nombre important de montants recours et payés nuls, nous avons pensé qu'il serait judicieux de prédire tout d'abord si la valeur du montant recours ou payé était différente de 0, puis, dans le cas échéant, de prédire la valeur du montant en question.
  Pour cela, nous avons dû créer deux nouvelles variables à expliquer, qui prennent comme valeurs 0 ou 1 selon que les montants payé et recours valent 0.

9.1 Prédiction des montants payé et recours séparément

Pour ce modèle, nous avons prédit séparément les deux montants (payé et recours) par chacun des trois algorithmes de machine learning ; puis nous avons fait la somme des deux pour obtenir le montant total.

Nous obtenons alors 9 combinaisons de modèles possibles :

Montant payé	Montant recours
GLM	GLM
GLM	RF
GLM	GBM
RF	GLM
RF	RF
RF	GBM
GBM	GLM
GBM	RF
GBM	GBM

TABLE 9.1 – Différentes combinaisons de modèles possibles

Avec : RF le modèle de forêts aléatoires (Random Forests).

Les modèles de forêts aléatoires et GBM sont optimisés par hyperparamétrage, le modèle GLM a pour paramètres ceux par défaut.

Nous avons alors calculé les différentes erreurs sur la base de test pour chacun des modèles :

	RMSE	Err Euclid	MAE	10%	Err Relative
GLM - GLM	1318.392	31420.90	776.4423	2762.047	77.73%
GLM - RF	1281.955	30552.51	647.3001	2709.365	64.81%
GLM - GBM	1276.502	30422.55	653.1126	2689.720	65.39%
RF - GLM	1178.498	28086.84	704.9351	2603.721	70.58%
RF - RF	1109.344	26438.72	548.2315	2487.452	54.89%
RF - GBM	1110.399	26463.87	554.3048	2492.426	55.50%
GBM - GLM	1232.754	29379.91	701.4173	2639.948	70.22%
GBM - RF	1173.066	27957.39	548.5974	2594.469	54.92%
GBM - GBM	1172.484	27943.52	558.0607	2580.864	55.87%

TABLE 9.2 – Erreurs calculées sur le montant total pour chacune des combinaisons modèles

Le meilleur de ces modèle est alors celui prédisant chacun des deux montants grâce aux forêts aléatoires.

Il est également plus performant que les différents modèles testés jusqu'à présent (à l'exception du modèle simple, qui est très légèrement plus performant : 54.89% d'erreur contre 54.82% pour le modèle simple).

On peut avoir un aperçu plus visuel à l'aide des graphiques suivants :

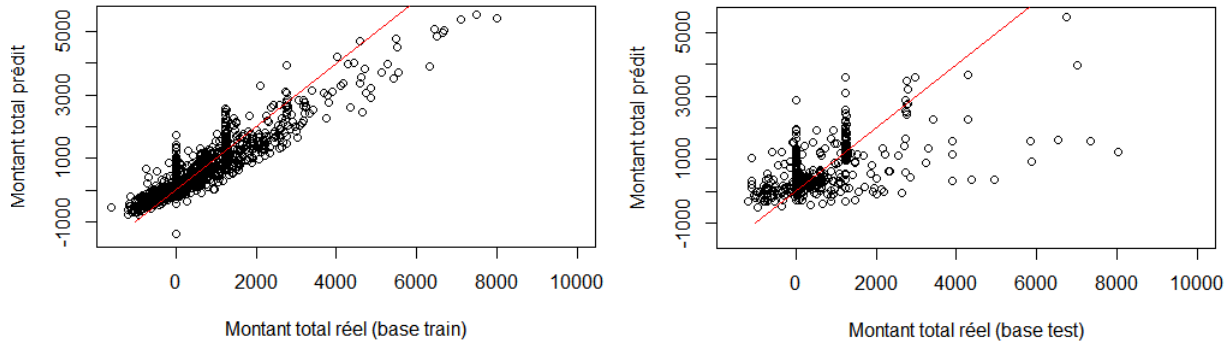


FIGURE 9.1 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases pour la combinaison de modèles RF et RF

9.2 Modèle par prédiction des valeurs nulles séparément des autres

9.2.1 Prévision du montant recours en deux étapes

9.2.1.1 Classification 0 ou 1 selon la présence ou non d'un montant recours

Nous avons testé trois algorithmes pour classifier les montants recours selon qu'ils soient nuls ou non. Notre base de données contient 2121 observations dont le montant recours est nul, soit 716 pour lesquelles il est différent de 0.

Les trois algorithmes de classification testés sont :
- Modèle GLM binomial
- Modèle de forêts aléatoires
- Modèle GBM

Pour chacun des trois modèles, on obtient les matrices de confusion suivantes :

	FALSE	**TRUE**
FALSE	401	29
TRUE	24	114

TABLE 9.3 – Matrice de confusion pour le montant recours sur la base de test pour le modèle GLM binomial

Le taux d'erreur obtenu pour cette matrice est de 9.33%.

	FALSE	**TRUE**
FALSE	378	52
TRUE	12	126

TABLE 9.4 – Matrice de confusion pour le montant recours sur la base de test pour le modèle RF

Le taux d'erreur obtenu pour cette matrice est de 11.26%.

	FALSE	**TRUE**
FALSE	390	40
TRUE	20	118

TABLE 9.5 – Matrice de confusion pour le montant recours sur la base de test pour le modèle GBM

Le taux d'erreur obtenu pour cette matrice est de 10.56%.

Le modèle retenu pour prédire la présence ou non d'un montant recours sera donc le modèle GLM binomial.

9.2.1.2 Prédiction du montant recours

Pour prédire le montant recours, nous avons sélectionné les observations de la base d'apprentissage dont le montant était non nul, afin d'apprendre un nouveau modèle de prédiction du montant sur celles-ci.

Pour cela, nous avons testé quatre modèles différents :
- Modèle de forêts aléatoires
- Modèle GBM
- Modèle GLM Gaussien
- Modèle GLM Gamma (nous avons, pour ce modèle, pris l'opposé des montants recours, car les montants doivent être positifs)

Nous avons par la suite testé l'erreur de chacun de ces modèles en le combinant au GLM binomial pour classifier.

Le meilleur modèle dans ce cas est celui qui combine le modèle GLM binomial avec un modèle de forêts aléatoires pour calculer le montant prédit.

On obtient, pour ce modèle, les résultats suivants :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	199.1760	9487.5525	40.3032	386.5247	4.18%
Base de test	414.7266	9887.0751	157.5487	1214.7510	15.77%

TABLE 9.6 – Erreurs obtenues pour le montant recours prédit avec la combinaison d'un modèle GLM binomial avec un modèle de forêts aléatoires

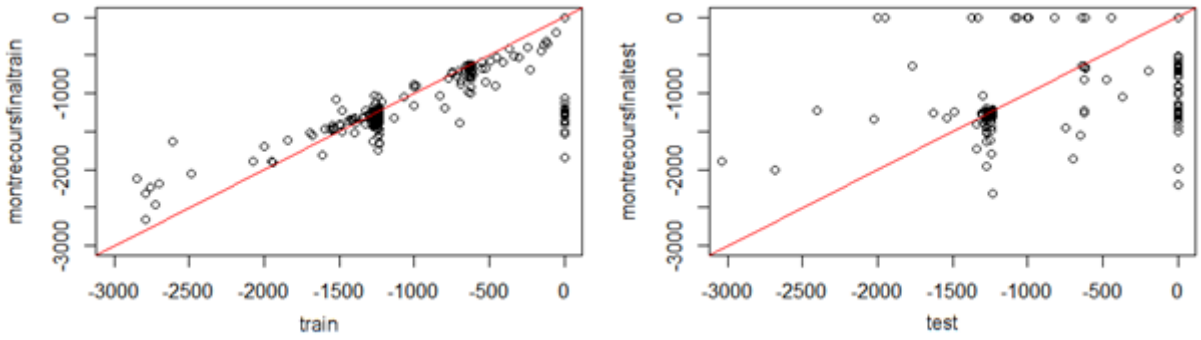


FIGURE 9.2 – Montants recours prédits par rapport aux montants recours réels sur les différentes bases pour la combinaison d'un modèle GLM binomial avec un modèle de forêts aléatoires

9.2.2 Prédiction du montant payé en deux étapes

9.2.2.1 Classification 0 ou 1 selon la présence ou non d'un montant payé

Là encore, nous avons certains sinistres dont le montant payé est nul. Toutefois, ces sinistres ne représentent que 10% des sinistres. Nous avons donc décidé d'utiliser une technique d'oversampling sur la base d'apprentissage afin d'obtenir un pourcentage de 50%.

Nous avons ensuite procédé de la même façon que pour le montant recours.

Une fois encore, le meilleur modèle pour la classification est un modèle GLM binomial. Nous obtenons alors la matrice de confusion suivante :

	FALSE	TRUE
FALSE	1	62
TRUE	8	497

TABLE 9.7 – Matrice de confusion pour le montant payé sur la base de test pour le modèle GLM binomial

Le taux d'erreur obtenu pour cette matrice est de 12.32%.

9.2.2.2 Prédiction du montant payé

Pour prédire le montant payé, nous avons sélectionné les sinistres de la base d'apprentissage dont le montant payé est différent de 0, afin de construire un nouveau modèle.

Là encore, les quatre mêmes modèles que pour prédire le montant recours ont été testés (RF, GBM, GLM gaussien et GLM gamma) ; et une fois encore, c'est le modèle de forêts aléatoires qui obtient les meilleurs résultats :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	685.1983	32638.7527	314.9824	1752.8274	32.65%
Base de test	1266.8108	30191.5864	621.4854	2954.5746	62.22%

TABLE 9.8 – Erreurs obtenues pour le montant payé prédit avec la combinaison d'un modèle GLM binomial avec un modèle de forêts aléatoires

9.2.3 Prédiction du montant total

Nous obtenons donc un modèle optimal pour la prévision du montant payé et du montant recours. En faisant la somme des deux, nous obtenons un modèle final :

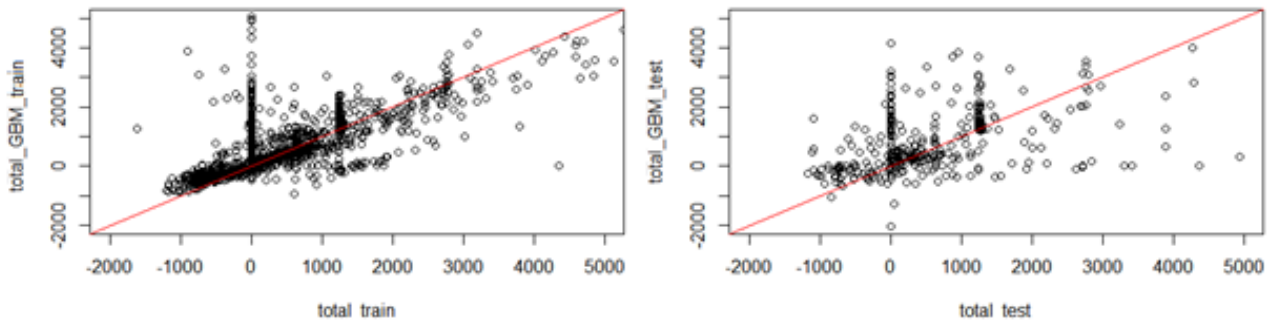


FIGURE 9.3 – Montants totaux prédits par rapport aux montants totaux réels sur les différentes bases pour la combinaison d'un modèle GLM binomial avec un modèle de forêts aléatoires

On obtient également le tableau d'erreur suivant :

	RMSE	Err Euclid	MAE	10%	Err Relative
Base d'apprentissage	739.5754	35228.9522	345.0542	1890.7507	35.77%
Base de test	1276.9468	30433.1550	639.8022	2995.4343	64.05%

TABLE 9.9 – Erreurs obtenues pour le montant total prédit avec la combinaison d'un modèle GLM binomial avec un modèle de forêts aléatoires

On remarque que les résultats obtenus par ce modèle sont bien moins bons que ceux obtenus par chacun des autres modèles, ce modèle est donc bien moins performant.

Chapitre 10

Conclusion

Un bon provisionnement des sinistres est un élément clé de l'activité d'un assureur. C'est en effet l'un des éléments qui intervient dans le calcul des Provisions Pour Sinistres À Payer (PSAP). Ces provisions figureront dans le bilan Solvabilité II de l'assureur : dans ce contexte, on comprend l'importance de les estimer au mieux.

Les méthodes classiques (forfaitaire ou par coût moyen) ont tendance à les surévaluer. Le développement du Machine Learning, qui s'opère à l'heure actuelle, ouvre alors de nouvelles possibilités pour les calculer.

L'objectif de notre bureau d'étude était alors de comparer les résultats d'un modèle classique de provisionnement à ceux pouvant être obtenus avec les différents algorithmes de Machine Learning implémentés dans le package *h2o*.

Notre bureau d'étude fut alors l'occasion pour nous de nous familiariser avec la tarification automobile et la DataScience. En effet, nous avons eu la possibilité de réaliser différentes missions telles que :

- Prendre connaissance des données, rechercher les valeurs aberrantes, ainsi que construire de nouvelles variables
- Construire un modèle classique de provisionnement
- Préparer la base pour le Machine Learning (normalisation, oversampling, etc.)
- Implémenter les différents modèles, les optimiser et les comparer

Notre meilleur modèle de Machine Learning, obtenu en prédisant séparément le montant recours du montant payé, via des forêts aléatoires pour chacun d'entre eux, nous permet d'obtenir des résultats équivalents à ceux du modèle simple, et une petite amélioration de quelques pourcents sur les autres modèles de Machine Learning.

Toutefois, le Machine Learning présente un inconvénient majeur qu'il est nécessaire de prendre en compte dans le choix du modèle : il est très complexe, et par conséquent difficilement interprétable. En effet, en concurrence avec un modèle simple qui ne prend en compte qu'un faible nombre de variables et qui est alors facilement compréhensible, le côté « boîte noire » des algorithmes de Machine Learning est pénalisant : dans un contexte professionnel, l'actuaire chargé de provisionner les sinistres aura à expliquer et à justifier la fiabilité de son modèle.

Il convient cependant de préciser deux points qui ont pu affecter les performances de nos modèles de Machine Learning :

- Le nombre d'observations : pour notre bureau d'étude, celui-ci était relativement faible, un peu moins de 3000 observations dans la base de données étudiée. Cela nous a permis de ne pas perdre de temps au niveau du calcul, néanmoins, il serait intéressant de tester ces modèles sur des bases contenant plus de données, ce qui permettrait sans doute d'améliorer encore les performances.
- La répartition des valeurs des sinistres : en effet, nous observons des « pics » de fréquence à certains endroits (nous pouvons retrouver les représentations graphiques de ces "pics" en annexes B.1 et B.2.

C'est notamment pour cette raison que le modèle simple fonctionne relativement bien. En revanche, il est fortement possible que cette répartition par "pics" entrave les performances des algorithmes de Machine Learning.

Il serait alors intéressant de reprendre ces algorithmes sur une base de données plus conséquente, ce qui permettrait sûrement d'obtenir des meilleurs modèles. Dans ce contexte, notre code R (disponible en annexe C) a été conçu pour être en grande partie réutilisable sur une autre base de données, sans trop avoir à être modifié.

Malgré l'absence de résultats probants, notre bureau d'étude fut très enrichissant pour nous trois. Ce fut l'occasion d'acquérir une première expérience pratique du domaine de l'actuariat, en l'occurrence en provisionnement IARD. Nous avons également pu enrichir nos connaissances et nos capacités dans le Machine Learning, domaine en pleine expansion et de plus en plus utilisé en actuariat.

Annexe A

Validation croisée appliquée sur notre modèle simple

– Pour le montant payé :

Croisement	RMSE	Err Euclid	MAE	10%
1	1145.3619	19267.97	525.9702	527.661
2	1154.0642	19448.64	592.5382	2693.413
3	1149.2869	19368.13	584.12	2868.388
4	911.2719	15357.03	525.3196	2179.209
5	1046.0151	17627.76	492.4953	2475.984
6	885.0733	14915.52	467.3386	2057.742
7	1625.3420	27390.75	674.8267	3079.069
8	1202.0908	20257.99	560.8381	2663.659
9	1326.5994	22356.25	582.344	2901.615
10	1016.1792	17334.72	501.2834	2376.301

TABLE A.1 – Erreurs obtenues sur les montants payés prédits avec la validation croisée

– Pour le montant recours :

Croisement	RMSE	Err Euclid	MAE	10%
1	594.6298	10003.221	189.4328	1311.5958
2	396.946	6689.453	144.4006	892.9432
3	426.2646	7183.538	152.7907	1069.7237
4	359.8651	6064.554	165.7336	936.7738
5	363.2866	6122.214	121.025	920.4183
6	486.6184	8200.639	164.3849	1144.47
7	301.3714	5078.802	134.6404	784.3149
8	292.3275	4926.390	116.1361	702.8593
9	345.6553	5825.087	171.1021	946.7435
10	310.7398	5300.824	133.128	869.3892

TABLE A.2 – Erreurs obtenues sur les montants recours prédits avec la validation croisée

Annexe B

Répartition des valeurs des sinistres

– Pour le montant payé :

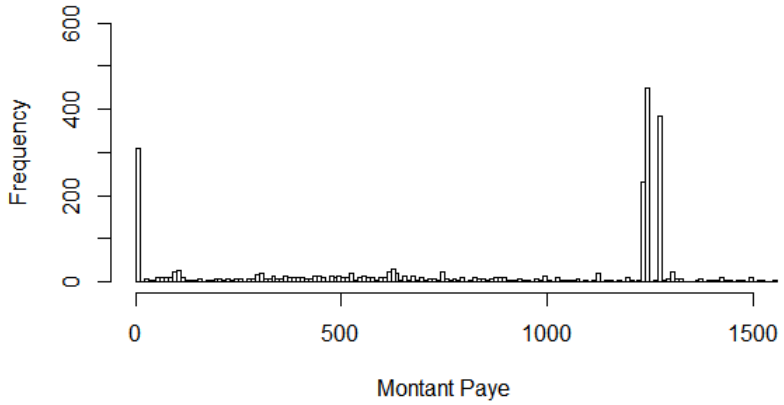


FIGURE B.1 – Répartition des sinistres en "pics" selon leur montant payé

– Pour le montant recours :

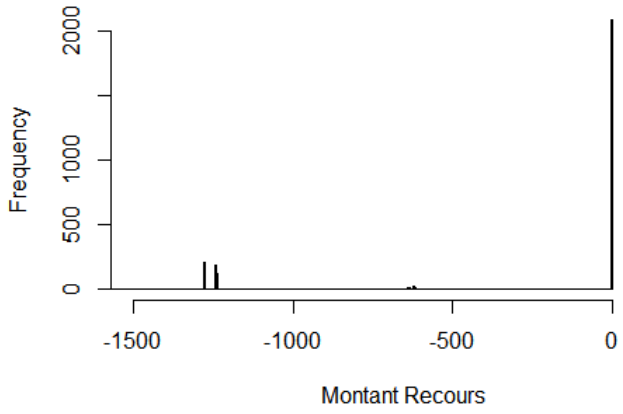


FIGURE B.2 – Répartition des sinistres en "pics" selon leur montant recours

Annexe C

Codes informatiques R

```
## Importation de la base
library(readr)
base <- read_delim(
  "C:/Users/Yoan/Desktop/BE/base.csv",
  ";",
  escape_double = FALSE,
  trim_ws = TRUE
)

x = base$CRM
## si le CRM est different de 50 on ne peut pas avoir une annee de CRM a 50%

for (i in 1:2837) {
  if (x[i] != 50) {
    base$NbAnneeCRM50[i] = 0
  }
}
# View(base)

x = base$NbAnneeCRM50
y = as.factor(x)
z = levels(y)
z

## on remplace le "3ans" par 3 pour qu'il soit un nombre comme les autres
a = base$NbAnneeCRM50
a[base$NbAnneeCRM50 == "3ans"] = "3.00"

x = base$NbAnneeCRM50
base$NbAnneeCRM50 = a
y = as.factor(x)
z = levels(y)
z

t = NULL
U = NULL
A = base$MontantTotal
for (i in 1:length(z)) {
  F = A[y == z[i]]
```

```
  t = rbind(t, floor(c(mean(F), var(F), min(F), max(F))))
  c = cbind((z), t)

  e = rbind(c("valeur", "moyenne", "ecart type", "min", "max"), c)
  hist(F, xlab = z[i])
}


## On renvoie une table avec l'ecart type,le max le min pour chaque classe ##
f = rbind(e, (c("Total", mean(A), var(A), min(A), max(A))))
table(x)   ## le nombre d'elements dans chaque classe
pie(table(x))   ##representation graphique de ce nombre


### code qui renvoie les regions (ancienne repartition) ###
library(readxl)
departements <-
  read_delim(
    "C:/Users/Yoan/Desktop/BE/departements.csv",
    ";",
    escape_double = FALSE,
    trim_ws = TRUE,
    col_names = FALSE
  )
##dans le document importe, nous avons la region en fonction du departement
##nous allons donc remplacer le code postal en la region d'appartenance du departement

reg = departements$X3
reg = c(c(reg[1:19], 0), reg[20:94])
### le code 20 est la corse qui n'est pas representee dans notre portefeuille

reg[50] = "Basse-Normandie"
x = base$CodePostal

## on remplace le code postal par la valeur du departement correspondant au code postal:
y = floor(x / 1000)
region = cbind(1:95, reg)
## on obtient donc une table des regions en fonction du departement

z = NULL
for (i in 1:length(x)) {
  for (j in 1:95) {
    if (y[i] == region[j, 1])
      z[i] = region[j, 2]

  }
}
table(z) ## nombre de personnes pour chaque region
z
pie(table(z), rad = 1) ## representation graphique de la table
table(z) / length(x) * 100 ##pourcentage de chaque region dans le portefeuille
```

```
## arbre de decision

library(rpart)
ad.base <-
  rpart(
    MontantTotal ~ NatSin + GarPr + TauxResp + NbGarSin + TypeEval + AnneePermis +
      Formule + Usage + CSP + NbAnneeCRM50 + ClasseDePrix + Financement + Energie,
    base
  )
base$ClasseDePrix = as.factor(base$ClasseDePrix)
ad.base

plot (ad.base,
      uniform = T,
      margin = 0.1)
text (ad.base, use.n = T, all = T)
levels(base$CSP)
## on obtient notre premier arbre de regression

## Pour retrouver les valeurs de l'arbre de decision avec n=68 et 3318 de moyenne avec
# le taux de responsabilite: taux de resp >75 et nature sinistre materiel

#View(base)
y = which(base$TauxResp > 75)
l = length(y)
M = 0
for (i in 1:l) {
  M = M + base$MontantTotal[y[i]]
}
M / l ##1377.95
l   ##1297

Z1 = NULL
Z2 = NULL
for (i in 1:l) {
  Z1[i] = base$NatSin[y[i]]
  Z2[i] = y[i]
}
Z1
Z2
length(Z1) ##1397
Z1[Z1 == 0] = 2

#on obtient donc 2=MAT 1=CORP

z1 = which(Z1 > 1)
length(z1) ## on a 68 sinistres

Y2 = NULL
for (j in 1:length(z1)) {
  Y2[j] = Z2[z1[j]]
}
```

```
Y2

p = length(Y2)
N = 0
for (j in 1:p) {
  N = N + base$MontantTotal[Y2[j]]
}
## Moyenne des montants de cette classe de sinistres 3317.98 ce qu'on avait dans notre
#  arbre de decision:
N / p
N   ##Motant total de la classe



## Statistiques descriptives pour Montant Total

which(is.na(BonneBase$TauxResp))
BonneBase$TauxResp[which(is.na(BonneBase$TauxResp))] = 0

x = BonneBase$TauxResp
y = as.factor(x)
z = levels(y)
z
t = NULL
U = NULL
A = BonneBase$MontantTotal
for (i in 1:length(z)) {
  F = A[y == z[i]]

  t = rbind(t, floor(c(mean(F), var(F), min(F), max(F))))
  c = cbind((z), t)

  e = rbind(c("valeur", "moyenne", "ecart type", "min", "max"), c)
  hist(F, xlab = z[i])
}
f = rbind(e, (c("Total", mean(A), var(A), min(A), max(A))))
table(x)
pie(table(x))
as.data.frame(f)

table(x)



## Statistiques descriptives pour Montant Paye

x = BonneBase$TauxResp
y = as.factor(x)
z = levels(y)
z
t = NULL
U = NULL
A = BonneBase$MontantPaye
```

```
for (i in 1:length(z)) {
  F = A[y == z[i]]

  t = rbind(t, floor(c(mean(F), var(F), min(F), max(F))))
  c = cbind((z), t)

  e = rbind(c("valeur", "moyenne", "ecart type", "min", "max"), c)
  hist(F, xlab = z[i])
}
f = rbind(e, (c("Total", mean(A), var(A), min(A), max(A))))
table(x)
as.data.frame(f)
pie(table(x))



## Statistiques descriptives pour Montant Recours

x = BonneBase$TauxResp
y = as.factor(x)
z = levels(y)
z
t = NULL
U = NULL
A = BonneBase$MontantRecours
for (i in 1:length(z)) {
  F = A[y == z[i]]

  t = rbind(t, floor(c(mean(F), var(F), min(F), max(F))))
  c = cbind((z), t)

  e = rbind(c("valeur", "moyenne", "ecart type", "min", "max"), c)
  hist(F, xlab = z[i])
}
f = rbind(e, (c("Total", mean(A), var(A), min(A), max(A))))
table(x)
as.data.frame(f)
pie(table(x))



### Age des personnes sinistrees ###

x = base$DateSurv
## x correspond au nombre de jours ecoules entre 1er janvier 1900 et la survenance du
#  sinistre
y = base$AnneeNai
u = cbind(x, y)

## code qui transforme le nombre de jours ecoules entre 1er janvier 1900 et la survenance
#  du sinistre en l'annee du sinistre
x1 = x[1:79] #sinistres ayant eu lieu en 2011
x2 = x[80:681] #sinistres ayant eu lieu en 2012
```

```
x3 = x[682:1817] #sinistres 2013
x4 = x[1818:2806] #sinistre 2014
x5 = x[2807:2837] #sinistre 2015

A = c(
  rep(2011, 79),
  rep(2012, 681 - 79),
  rep(2013, 1817 - 681),
  rep(2014, 2806 - 1817),
  rep(2015, 2837 - 2806)
)
## A remplace le nombre de jours en annee de la matrice x
Age = A - y #matrice d'age de l'assure
table(Age) ##nombre de personnes dans chaque classe d'age
pie(table(Age)) ##representation graphique du nombre de personnes dans chaque classe
hist(Age)


## Regroupement en categories ##

sum(table(Age[Age < 20]))
sum(table(Age[Age >= 56]))

## Calcul de la moyenne selon l'age
Y = base$MontantTotal
z = levels(as.factor(Age))
U = NULL
for (i in 1:length(z)) {
  A = Y[Age == z[i]]
  U = cbind(U, c(floor(mean(A)), floor(var(A))))
  P = rbind(z, U)
}
## matrice avec la moyenne et la variance  des sinistres pour chaque age
P
x = base[, -34:-36]
head(x)
dev.off()
split.screen(c(1, 2))
screen(1)
hist(x$MontantTotal)
logx = log(x$MontantTotal)
screen(2)
hist(logx)

dev.off()
split.screen(c(1, 2))
screen(1)
plot(P[1,], P[2,]) ## graphe de la moyenne en fonction de l'age
screen(2)
plot(P[1,], P[3,]) ## graphe de la variance en fonction de l'age


base2 = base
```

```
base2$AnneeNai = Age  ## on remplace l'annee de naissance par l'age de l'assure

setwd("C:/Users/Yoan/Desktop/BE")
getwd()#pour savoir dans quel dossier on se trouve
data(iris)
write.csv2(base2, file = 'base2.csv')

## Calcul du nombre d'annees de permis

Nbanneepermis = 2012 - base2$AnneePermis
base3 = base2
base3$AnneePermis = Nbanneepermis

setwd("C:/Users/Yoan/Desktop/BE")
getwd()#pour savoir dans quel dossier on se trouve
data(iris)
write.csv2(base3, file = 'base3.csv')


library(readr)

BonneBaseClasse <-
  read.csv2("C:/Users/Yoan/Desktop/BE/BonneBaseClasse.csv")

### Classes selon le critere de Cochran ###

#Pour les montants payes
"str(BonneBaseClasse)
BonneBaseClasse$ClasseMP=as.numeric(BonneBaseClasse$ClasseMP)
BonneBaseClasse$ClasseMR=as.numeric(BonneBaseClasse$ClasseMR)"

#affiche les effectifs de chaque classe:
T = table(BonneBaseClasse$ClasseMP)
T
sum(T)
#calcule le pourcentage de population dans chaque classe:
pourcent = table(BonneBaseClasse$ClasseMP) / nrow(BonneBaseClasse)
pourcent
sum(pourcent)
#releve les classes contenant moins de 5% de la population:
which(pourcent < 0.05)
length(which(pourcent < 0.05))
sum(pourcent[which(pourcent < 0.05)])  #environ 16%

W = which(pourcent < 0.05)

#Regroupement des "petites" classes en classes de plus de 5%
U1 = 0
U2 = 0
U3 = 0
UU1 = NULL
UU2 = NULL
UU3 = NULL
```

```
i = 1
for (i in 1:length(W)) {
  if (U1 < 0.05) {
    U1 = U1 + pourcent[W[i]]
    UU1 = c(UU1, W[i])
  }
  else if (U2 < 0.05) {
    U2 = U2 + pourcent[W[i]]
    UU2 = c(UU2, W[i])
  }
  else{
    U3 = U3 + pourcent[W[i]]
    UU3 = c(UU3, W[i])
  }
}
U1  #0.074
U2  #0.051
U3  #0.387
UU1
UU2 ##0.08988
UU3
#pour avoir un pourcentage superieur a 5% dans chaque classe:
U2 = U2 + U3
U2 ### 0.1286
UU2 = c(UU2, UU3)
UU2

# Remplacement par de nouvelles classes: UU1->4441, UU2->4442 #

length(UU2) ##66
length(UU1)  ##5

n = length(BonneBaseClasse[, 1])
x = BonneBaseClasse$ClasseMP == names(UU1)
for (i in 1:n) {
  for (j in 1:length(UU1)) {
    if (BonneBaseClasse$ClasseMP[i] == names(UU1)[j]) {
      BonneBaseClasse$ClasseMP[i] = 4441
    }

  }
  for (j in 1:length(UU2)) {
    if (BonneBaseClasse$ClasseMP[i] == names(UU2[j])) {
      BonneBaseClasse$ClasseMP[i] = 4442
    }
  }
}

table(BonneBaseClasse$ClasseMP)



#########
```

```
#Pour les montants recours

#affiche les effectifs de chaque classe:
T2 = table(BonneBaseClasse$ClasseMR)
T2
sum(T2) ##2837
length(T2) # 14
#calcule le pourcentage de population dans chaque classe:
pourcent2 = table(BonneBaseClasse$ClasseMR) / nrow(BonneBaseClasse)
pourcent2
sum(pourcent2) ## on a bien egual a 1
#releve les classes contenant moins de 5% de la population:
which(pourcent2 < 0.05)
length(which(pourcent2 < 0.05)) ###9
sum(pourcent2[which(pourcent2 < 0.05)])   #environ 8%

W2 = which(pourcent2 < 0.05)
pourcent2[W2]

#On ne cree qu'une seule nouvelle classe de 8% de la population
U4 = sum(pourcent2[which(pourcent2 < 0.05)])  # 0.082
U4 ## 8.21%
UU4 = W2
UU4


# Nouvelle classe creee: UU4->444 #


length(UU4) ##9

n = length(BonneBaseClasse[, 1])
for (i in 1:n) {
  for (j in 1:length(UU4)) {
    if (BonneBaseClasse$ClasseMR[i] == names(UU4)[j]) {
      BonneBaseClasse$ClasseMR[i] = 444
    }
  }
}

table(BonneBaseClasse$ClasseMR)



## MODELE SIMPLE ##

set.seed(1) #bloque le phenomene aleatoire
A = sample(1:nrow(BonneBaseClasse), floor(nrow(BonneBaseClasse) * 0.8))

b = BonneBaseClasse
baseapp = (b[A,])
basetest = (b[-A,])
```

```
#View(baseapp)
#View(basetest)


### TRAVAIL SUR LA BASE TEST ###

#### Pour les montants payes

x = baseapp$MontantPaye
y = baseapp$ClasseMP
MoyMP = tapply(x, y, 'mean')
r = aov(x ~ y)
anova(r)

#View(MoyMP)
length(MoyMP)

R2 = cbind(rownames(MoyMP), MoyMP)
#View(R2)

#Remplacement des Montants Payes dans la base de test par les Montants Payes moyens
BTMP = basetest

for (i in 1:length(R2[, 1])) {
  BTMP[BTMP[, 41] == R2[i, 1], 37] = R2[i, 2]
}
#View(BTMP)

## Pour reperer quelles classes apparaissent dans la base de test mais pas dans la base
#  d'app
levels(as.factor(baseapp$ClasseMP))
levels(as.factor(basetest$ClasseMP))   #aucune classe dans ce cas la



#### Pour les montants recours


x1 = baseapp$MontantRecours
y1 = baseapp$ClasseMR
MoyMR = tapply(x1, y1, 'mean')
r1 = aov(x1 ~ y1)
anova(r1)

#View(MoyMR)
length(MoyMR) #6

R21 = cbind(rownames(MoyMR), MoyMR)
#View(R21)


#Remplacement des Montants Recours dans la base de test par les Montants Recours moyens
```

```
for (i in 1:length(R21[, 1])) {
  BTMP[BTMP[, 42] == R21[i, 1], 39] = R21[i, 2]
}
#View(BTMP)

## Pour reperer quelles classes apparaissent dans la base de test mais pas dans la base
#  d'app
levels(as.factor(baseapp$ClasseMR))
levels(as.factor(basetest$ClasseMR))  #Aucune classe dans ce cas la


#Calcul du montant total moyen a partir des montants paye et recours moyen
BT = BTMP
#View(BT)

BT$MontantPaye = as.numeric(BT$MontantPaye)
BT$MontantRecours = as.numeric(BT$MontantRecours)
BT$MontantTotal = BT$MontantRecours + BT$MontantPaye
#View(BT)


#Calcul des erreurs

#RMSE
rmse = function(x, y) {
  rm = sqrt(sum((x - y) ^ 2) / length(x))
  return(rm)
}

#Norme euclidienne
euclid = function(x, y) {
  eu = sqrt(sum((x - y) ^ 2))
  return(eu)
}

#MAE
mae = function(x, y) {
  mae = (sum(abs(x - y)) / length(x))
  return(mae)
}

#Erreur relative
rela = function(x, y) {
  rela = sum(abs(x - y)) / (sum(abs(y)))
  return(rela)
}


#Moyenne des 10% des plus grosses erreurs
erreur = function(x, y) {
  v = abs(x - y)
  k = sort(v, decreasing = TRUE)
  i = floor(length(v) / 10)
```

```
  l = sum(k[1:i]) / i
  return(l)
}


#Renvoi des 5 erreurs

err = function(x, y) {
  vec = c(rmse(x, y), mae(x, y), rela(x, y), euclid(x, y), erreur(x, y))
  names(vec) = c(
    'rmse',
    'mae',
    'erreur relative',
    'norme euclidienne',
    'moyenne sur les 10% des pires predictions'
  )
  return(vec)
}

#err(prediction,realite)
err(BT$MontantTotal, basetest$MontantTotal)
err(BT$MontantPaye, basetest$MontantPaye)
err(BT$MontantRecours, basetest$MontantRecours)

#Representation graphique des ecarts
plot(BT$MontantTotal, basetest$MontantTotal)
plot(BT$MontantPaye, basetest$MontantPaye)
plot(BT$MontantRecours, basetest$MontantRecours)

#calcul des erreurs par classe sur la base de test

#Montant paye
D = NULL
for (i in 1:nrow(R2)) {
  y = basetest$MontantPaye[basetest$ClasseMP == R2[i]]
  x = rep(R2[i, 2], length(y))
  x = as.numeric(x)
  y = as.numeric(y)
  D = cbind(D, err(x, y))
}
D ##affiche les erreurs du montant paye pour chaque classe sur la base de test

#Montant recours
D1 = NULL
for (i in 1:nrow(R21)) {
  y = basetest$MontantRecours[basetest$ClasseMR == R21[i]]
  x = rep(R21[i, 2], length(y))
  x = as.numeric(x)
  y = as.numeric(y)
  D1 = cbind(D1, err(x, y))
}
D1   ##affiche les erreurs du montant recours pour chaque classe sur la base de test
```

```
### TRAVAIL SUR LA BASE D'APPRENTISSAGE ###


#Sur la base d'apprentissage

#On cree une nouvelle base d'apprentissage avec les montants payes moyens
BAMP = baseapp

for (i in 1:length(R2[, 1])) {
  BAMP[BAMP[, 41] == R2[i, 1], 37] = R2[i, 2]
}
#View(BAMP)


#On cree une nouvelle base d'apprentissage avec les montants recours moyens

for (i in 1:length(R21[, 1])) {
  BAMP[BAMP[, 42] == R21[i, 1], 39] = R21[i, 2]
}
#View(BAMR)

#Remplacement par les valeurs moyennes des montants dans une meme base
BA = BAMP
#View(BA)

BA$MontantPaye = as.numeric(BA$MontantPaye)
BA$MontantRecours = as.numeric(BA$MontantRecours)
BA$MontantTotal = BA$MontantRecours + BA$MontantPaye
#View(BA)

## Calcul des erreurs (a partir des fonctions donnees pour les calculs des erreurs sur la
#  base de test)

#err(prediction,realite)
err(BA$MontantTotal, baseapp$MontantTotal)
err(BA$MontantPaye, baseapp$MontantPaye)
err(BA$MontantRecours, baseapp$MontantRecours)

#Representation graphique des ecarts
plot(BA$MontantTotal, baseapp$MontantTotal)
plot(BA$MontantPaye, baseapp$MontantPaye)
plot(BA$MontantRecours, baseapp$MontantRecours)


#calcul des erreurs par classe sur la base d'apprentissage

#Montant paye
D2 = NULL

for (i in 1:nrow(R2)) {
  y = baseapp$MontantPaye[baseapp$ClasseMP == R2[i]]
  x = rep(R2[i, 2], length(y))
```

```
  x = as.numeric(x)
  y = as.numeric(y)
  D2 = cbind(D2, err(x, y))
}
D2
## affiche les erreurs du montant paye pour chaque classe sur la base d'apprentissage



#Montant recours
D3 = NULL
for (i in 1:nrow(R21)) {
  y = baseapp$MontantRecours[baseapp$ClasseMR == R21[i]]
  x = rep(R21[i, 2], length(y))
  x = as.numeric(x)
  y = as.numeric(y)
  D3 = cbind(D3, err(x, y))
}
D3
## affiche les erreurs du montant recours pour chaque classe  sur la base
#  d'apprentissage


set.seed(1)
B = BonneBaseClasse
k = 10 #nombre de classes pour la validation croisee

###Validation croisee##

### Creation d'une liste contenant les partitions de la base en k elements
p = length(t(B[, 1]))
a = sample(1:p, p)
b = floor(p / k)
l = NULL
l[[1]] = B[a[1:b], ]
for (i in 2:(k - 1)) {
  l[[i]] = B[a[(b * (i - 1)):(b * i)], ]
}
l[[k]] = B[a[(b * (k - 1)):p], ]

## creation des differentes bases de test et d'apprentissage
Apprtot = NULL
Testtot = NULL


for (j in 1:10) {
  #j=5 ## On choisit quelle base sera la base de test
  Appr = NULL
  Test = l[[j]]  ##On prend la partition j de la base

  for (i in (1:10)[-j]) {
    ## la boucle concatene les k-1 autres partition pour former la base d'apprentissage
    Appr = rbind(Appr, l[[i]])
```

```
  }
  Apprtot[[j]] = Appr
  ## On enregistre les base d'apprentissage et de test dans une liste
  Testtot[[j]] = Test
}

## On obtient une liste de matrice d'apprentissage (Apprtot) et de test (Testtot)

## MODELE SIMPLE AVEC VALIDATION CROISEE###
MR = NULL
MP = NULL
RmodeleMP = NULL
RmodeleMR = NULL
##On applique la validation croisee au modele

for (i in 1:10) {
  set.seed(1) #bloque le phenomene aleatoire
  A = sample(1:nrow(BonneBaseClasse), floor(nrow(BonneBaseClasse) * 0.8))

  b = BonneBaseClasse
  baseapp = Testtot[[i]]
  ## On prend les bases de test et d'apprentissage obtenues dans le code de la validation
  #  croisee
  basetest = Apprtot[[i]]
  #View(baseapp)
  #View(basetest)


  ### TRAVAIL SUR LA BASE DE TEST ###

  #### Pour les montants payes

  x = baseapp$MontantPaye
  y = baseapp$ClasseMP
  R = tapply(x, y, 'mean')
  r = aov(x ~ y)
  anova(r)

  #View(R)
  length(R)

  R2 = cbind(rownames(R), R)
  #View(R2)

  RmodeleMP = rbind(RmodeleMP, R) ##enregistrement des differents modele

  #Remplacement des Montants Payes dans la base de test par les Montants Payes moyens
  BTMP = basetest

  for (i in 1:length(R2[, 1])) {
    BTMP[BTMP[, 41] == R2[i, 1], 37] = R2[i, 2]
```

```
}
#View(BTMP)

## Pour reperer quelles classes apparaissent dans la base de test mais pas dans la
#  base d'app
levels(as.factor(baseapp$ClasseMP))
levels(as.factor(basetest$ClasseMP))   #aucune classe dans ce cas la



#### Pour les montants recours


x1 = baseapp$MontantRecours
y1 = baseapp$ClasseMR
R1 = tapply(x1, y1, 'mean')
r1 = aov(x1 ~ y1)
anova(r1)

RmodeleMR = rbind(RmodeleMR, R1) ##On Garde en memoire le modele

#View(R1)
length(R1)

R21 = cbind(rownames(R1), R1)
#View(R21)


#Remplacement des Montants Recours dans la base de test par les Montants Recours moyens

for (i in 1:length(R21[, 1])) {
  BTMP[BTMP[, 42] == R21[i, 1], 39] = R21[i, 2]
}
#View(BTMP)

## Pour reperer quelles classes apparaissent dans la base de test mais pas dans la
#  base d'app
levels(as.factor(baseapp$ClasseMR))
levels(as.factor(basetest$ClasseMR))  #Aucune classe dans ce cas la


#Remplacement par les valeurs moyennes des montants dans une meme base
BT = BTMP
#View(BT)

BT$MontantPaye = as.numeric(BT$MontantPaye)
BT$MontantRecours = as.numeric(BT$MontantRecours)
BT$MontantTotal = BT$MontantRecours + BT$MontantPaye
#View(BT)


#Calcul des erreurs
```

```
#err(prediction,realite)
err(BT$MontantTotal, basetest$MontantTotal)
err(BT$MontantPaye, basetest$MontantPaye)
err(BT$MontantRecours, basetest$MontantRecours)



### TRAVAIL SUR LA BASE D'APPRENTISSAGE ###



#Sur la base d'apprentissage

#On cree une nouvelle base d'apprentissage avec les montants payes moyens
BAMP = baseapp

for (i in 1:length(R2[, 1])) {
  BAMP[BAMP[, 41] == R2[i, 1], 37] = R2[i, 2]
}
#View(BAMP)


#On cree une nouvelle base d'apprentissage avec les montants recours moyens

for (i in 1:length(R21[, 1])) {
  BAMP[BAMP[, 42] == R21[i, 1], 39] = R21[i, 2]
}
#View(BAMR)

#Remplacement par les valeurs moyennes des montants dans une meme base
BA = BAMP
#View(BA)

BA$MontantPaye = as.numeric(BA$MontantPaye)
BA$MontantRecours = as.numeric(BA$MontantRecours)
BA$MontantTotal = BA$MontantRecours + BA$MontantPaye
#View(BA)

## Calcul des erreurs (a partir des fonctions donnees pour les calculs des erreurs
#  sur la base de test)

#err(prediction,realite)
rbind(
  err(BA$MontantTotal, baseapp$MontantTotal),
  err(BA$MontantPaye, baseapp$MontantPaye),
  err(BA$MontantRecours, baseapp$MontantRecours)
)

#View(BA)
table(BonneBaseClasse$ClasseMR)
```

```
  ###Cross validation du modele simple
  ## MP et MR representent les 5 erreurs des k validations croisees respectivement des
  #  montants payes et des montants recours
  MP = rbind(MP, err(BA$MontantPaye, baseapp$MontantPaye))
  MR = rbind(MR, err(BA$MontantRecours, baseapp$MontantRecours))
}

MR
MP
RmodeleMP ## differents modele obtenus
RmodeleMR

##Modele global de l'estimation des montants payes par classe:
ModeleGlobalMP = apply(RmodeleMP, 2, FUN = 'mean')

##Modele global de l'estimation des montants recours par classe
ModeleGlobalMR = apply(RmodeleMR, 2, FUN = 'mean')

##matrice du modele global obtenu pour le montant paye:
RR = cbind(rownames(R2), ModeleGlobalMP)

##matrice du modele global obtenu pour le montant recours:
RR2 = cbind(rownames(R1), ModeleGlobalMR)


#On applique le modele global a la base totale
BAMP = BonneBaseClasse

for (i in 1:length(RR[, 1])) {
  BAMP[BAMP[, 41] == RR[i, 1], 37] = RR[i, 2]
}
#View(BAMP)


#On cree une nouvelle base d'apprentissage avec les montants recours moyens

for (i in 1:length(RR2[, 1])) {
  BAMP[BAMP[, 42] == RR2[i, 1], 39] = RR2[i, 2]
}
#View(BAMR)

#Remplacement par les valeurs moyennes des montants dans une meme base
B = BAMP
#View(BA)

B$MontantPaye = as.numeric(B$MontantPaye)
B$MontantRecours = as.numeric(B$MontantRecours)
B$MontantTotal = B$MontantRecours + B$MontantPaye
#View(BA)
dev.off()
## affichage des differents graphes de la valeur estimee en abscisse et de la valeur
#  reelle en ordonnee
plot(B$MontantTotal, BonneBaseClasse$MontantTotal)
```

```
plot(B$MontantRecours, BonneBaseClasse$MontantRecours)
plot(B$MontantPaye , BonneBaseClasse$MontantPaye)

## Analyse plus precise des resultats
#montant recours

## avec hist pour chaque classe
MP = NULL
n = length(RR2[, 1])
for (i in 1:n) {
  MP[[i]] = BonneBaseClasse$MontantRecours[BonneBaseClasse$ClasseMR == RR2[i, 1]]
  boxplot(MP[[i]], main = paste("histogram de", RR2[i, 1]))
}
## on obtient donc 6 box plot, 1 pour chaque classe

MR = NULL
m = length(RR[, 1])
for (i in 1:m) {
  MR[[i]] = BonneBaseClasse$MontantPaye[BonneBaseClasse$ClasseMP == RR[i, 1]]
  boxplot(MR[[i]], main = paste("histogram de", RR[i, 1]))
}
## on obtient donc 7 box plot, 1 pour chaque classe

## calcul du nombre de 0 pour chaque classe pour montant recours

a = NULL
for (i in 1:length(RR2[, 1])) {
  a[i] = sum(BonneBaseClasse$MontantRecours == 0 &
               BonneBaseClasse$ClasseMR == RR2[i, 1])
}
names(a) = names(RR2[, 1])
a
sum(a) / length(base$NumSin) ### on a donc 0.75% des montants recours egaux a 0.


############################################################
#CODES DES 5 METRIQUES D'ERREURS UTILISEES DANS NOTRE BE#
############################################################

#RMSE
rmse = function(x, y) {
  rm = sqrt(sum((x - y) ^ 2) / length(x))
  return(rm)
}

#Norme euclidienne
euclid = function(x, y) {
  eu = sqrt(sum((x - y) ^ 2))
  return(eu)
}

#MAE
mae = function(x, y) {
```

```
  mae = (sum(abs(x - y)) / length(x))
  return(mae)
}

#Moyenne des 10% des plus grosses erreurs
erreur = function(x, y) {
  v = abs(x - y)
  k = sort(v, decreasing = TRUE)
  i = floor(length(v) / 10)
  l = sum(k[1:i]) / i
  return(l)
}

#Erreur relative
rela = function(x, y) {
  rela = sum(abs(x - y)) / (sum(abs(y)))
  return(rela)
}



#Renvoi des 5 erreurs directement

err = function(x, y) {
  vec = c(rmse(x, y), mae(x, y), rela(x, y), euclid(x, y), erreur(x, y))
  names(vec) = c(
    'rmse',
    'mae',
    'erreur relative',
    'norme euclidienne',
    'moyenne sur les 10% des pires predictions'
  )
  return(vec)
}


#renvoi des 5 erreurs pour 3 modeles
err2 = function(mat) {
  vec = NULL
  x = mat[, 1]
  for (i in 2:4) {
    y = mat[, i]
    k = c(rmse(x, y), mae(x, y), rela(x, y), euclid(x, y), erreur(x, y))
    vec = rbind(vec, k)
  }
  colnames(vec) = c(
    'rmse',
    'mae',
    'erreur relative',
    'norme euclidienne',
    'moyenne sur les 10% des pires predictions'
  )
  rownames(vec) = c('glm', 'RF', 'GBM')
```

```
  return(vec)
}

#renvoi des 5 erreurs pour 9 modeles
err3 = function(mat) {
  vec = NULL
  x = mat[, 1]
  for (i in 2:10) {
    y = mat[, i]
    k = c(rmse(x, y), mae(x, y), euclid(x, y), erreur(x, y))
    vec = rbind(vec, k)
  }
  colnames(vec) = c(
    'rmse',
    'mae',
    'erreur relative',
    'norme euclidienne',
    'moyenne sur les 10% des pires predictions'
  )
  rownames(vec) = c(
    'glm-glm',
    'glm-RF',
    'glm-gbm',
    'rf-glm',
    'rf-rf',
    'rf-gbm',
    'gbm-glm',
    'gbm-rf',
    'gbm-gbm'
  )

  return(vec)
}
###################################
####Normalisation de la base ######
###################################

BonneBase <-
  read_delim(
    "C:/Users/Yoan/Desktop/BE/BonneBase.csv",
    ";",
    escape_double = FALSE,
    trim_ws = TRUE
  )
#View(BonneBase)

BonneBase$NbAnneeAss = Base$NbAnneeAss

#Changement des "characteres" en "factor"
i <- sapply(BonneBase, is.character)
BonneBase[i] <- lapply(BonneBase[i], as.factor)
i
#Changement de certaines variables numeriques en facteurs (impossible a automatiser)
```

```
BonneBase[3] = lapply(BonneBase[3], as.factor)
BonneBase[4] = lapply(BonneBase[4], as.factor)
BonneBase[19] = lapply(BonneBase[19], as.factor)

#On verifie la classe de chaque variable
lapply(BonneBase, levels)


#Normalisation des variables quantitatives
Base = BonneBase[, 1:27]
Montant = BonneBase[, 28:31]
#View(Montant)
#View(Base)
sapply(Base, class)
Base1 = Base[sapply(Base, class) == "integer"]
Base1$NbAnneeAss[is.na(Base1$NbAnneeAss)] = 0
BaseInt = NULL
for (i in 1:length(Base1)) {
  k = max(Base1[, i])
  l = min(Base1[, i])
  Base1[, i] = (Base1[, i] - l) / (k - l)
}

View(Base1)

#Normalisation des variables qualitatives
Base2 = Base[sapply(Base, class) == "factor"]
#install.packages('pbkrtest')
library(caret)
library(readr)
dmy <- dummyVars(" ~ .", data = Base2)
trsf <- data.frame(predict(dmy, newdata = Base2))
print(trsf)
View(trsf)
dim(trsf)### 287 162
#Reconstruction de la base
basefinale = cbind(Base1, trsf, Montant)
#View(basefinale)
dim(basefinale) ###2837 173

basefinale[is.na(basefinale)] = 0

setwd("C:/Users/Yoan/Desktop/BE/BonneBase.cs")

write.csv2(basefinale, file = 'basefinal.csv')


############################################################
#Introduction : lancement d'H2O et importation de la base#
############################################################
library(h2o)
localH2O = h2o.init(nthreads = -1)
#En indiquant "-1", on autorise H2O a utiliser tous les coeurs du processeurs afin
```

```
#de paralleliser les calculs

library(readr) #Librairie pour l'importation de la base

basefinal <- read_delim(
  "C:/Users/Yoan/Desktop/BE/basefinal.csv",
  ";",
  escape_double = FALSE,
  locale = locale(decimal_mark = ","),
  trim_ws = TRUE
)
#chemin vers la base normalisee

b = basefinal[,-1]
#View(b)
################################################
#Modele predisant directement le montant total#
################################################

############################
#Creation bases train/test#
############################
set.seed(1)

n = length(b$MontantPaye)
A = sample(1:n, floor(n * 0.8))

train = (b[A,])
test = (b[-A,])

train.h2o <- as.h2o(train)
test.h2o <- as.h2o(test)

colnames(train.h2o)
y.dep3 <- 173

x.indep <- 1:169

#####
#GLM#
#####

regression.model_total <-
  h2o.glm(
    y = y.dep3,
    x = x.indep,
    training_frame = train.h2o,
    family = "gaussian",
    nfolds = 5
  )

#Montant total
h2o.performance(regression.model_total)
```

```
summary(regression.model_total)
h2o.varimp(regression.model_total)[1:100,]
predict.train_total <-
  as.data.frame(h2o.predict(regression.model_total, train.h2o, folds = 5))
predict.reg_total <-
  as.data.frame(h2o.predict(regression.model_total, test.h2o))

prediction_test_total = predict.reg_total$predict
prediction_train_total = predict.train_total$predict

total_train = train$MontantTotal
total_test = test$MontantTotal

err(prediction_train_total, total_train)
err(prediction_test_total, total_test)

plot(
  total_train,
  prediction_train_total,
  xlim = c(-1500, 3000),
  ylim = (c(0, 1500)),
  ylab = 'Montant total predit',
  xlab = 'Montant total reel (base train)'
)
lines(c(0, 10000), c(0, 10000), col = "blue")
plot(
  total_test,
  prediction_test_total,
  xlim = c(-1500, 3000),
  ylim = (c(0, 1500)),
  ylab = 'Montant total predit',
  xlab = 'Montant total reel (base test)'
)
lines(c(0, 10000), c(0, 10000), col = "blue")

###################################
#Random Forest avec random search#
###################################

#Parmetrage du Random Search
para = list(
  mtries = seq(20, 80, by = 5),
  ntrees = 1:200,
  max_depth = 5:100
)

search_criteria2 <-
  list(strategy = "RandomDiscrete", max_runtime_secs = 600)


RF_grid2 <- h2o.grid(
  "randomForest",
  x = x.indep,
```

```
  y = y.dep3,
  grid_id = "RFrandom1",
  training_frame = train.h2o,
  nfolds = 5,
  seed = 4,
  hyper_params = para,
  search_criteria = search_criteria2
)


#Selection du meilleur modele retourne par la Random search

RFresultats = h2o.getGrid("RFrandom1", 'rmse')
RFresultats

best_RF_model_id <- RFresultats@model_ids[[1]]
best_RF <- h2o.getModel(best_RF_model_id)

predict2.train_totalRF <-
  as.data.frame(h2o.predict(best_RF, train.h2o))

predict2.test_totalRF <-
  as.data.frame(h2o.predict(best_RF, test.h2o))

prediction_RF_test_total = predict2.test_totalRF$predict
prediction_RF_train_total = predict2.train_totalRF$predict

err(prediction_RF_train_total, total_train)
err(prediction_RF_test_total, total_test)

plot(prediction_RF_train_total, total_train)
plot(prediction_RF_test_total, total_test)

########################
#GBM avec random search#
########################

gbm_params <- list(
  learn_rate = seq(0.1, 0.3, 0.01),
  max_depth = seq(2, 10, 1),
  ntrees = seq(1, 200, 1)
)
search_criteria2 <- list(strategy = "RandomDiscrete",
                         max_runtime_secs = 600)


gbm_grid2 <- h2o.grid(
  "gbm",
  x = x.indep,
  y = y.dep3,
  grid_id = "gbm_grid1",
  training_frame = train.h2o,
  nfolds = 5,
```

```
  seed = 1,
  hyper_params = gbm_params,
  search_criteria = search_criteria2
)
gbm_gridperf = h2o.getGrid(grid_id = "gbm_grid1",
                           sort_by = "rmse",
                           decreasing = FALSE)

#choix du meilleur modele
best_gbm_model_id <- gbm_gridperf@model_ids[[1]]
best_gbm <- h2o.getModel(best_gbm_model_id)


#test du meilleur modele

predict2.train_totalGBM <-
  as.data.frame(h2o.predict(best_gbm, train.h2o))
predict2.test_totalGBM <-
  as.data.frame(h2o.predict(best_gbm, test.h2o))

prediction_GBM_test_total = predict2.test_totalGBM$predict
prediction_GBM_train_total = predict2.train_totalGBM$predict

err(prediction_GBM_train_total, total_train)
err(prediction_GBM_test_total, total_test)

plot(prediction_GBM_train_total, total_train)
plot(prediction_GBM_test_total, total_test)

################################
#Comparaison des trois modeles#
################################

total_glm_train2 = prediction_train_total
total_glm_test2 = prediction_test_total

total_RF_train2 = prediction_RF_train_total
total_RF_test2 = prediction_RF_test_total

total_GBM_train2 = prediction_GBM_train_total
total_GBM_test2 = prediction_GBM_test_total

total_train = train$MontantTotal
total_test = test$MontantTotal

predictions_train2 = cbind(total_train,
                           total_glm_train2,
                           total_RF_train2,
                           total_GBM_train2)
predictions_test2 = cbind(total_test, total_glm_test2, total_RF_test2, total_GBM_test2)

err2(predictions_train2)
```

```
err2(predictions_test2)



##################################################
#Modele de prediction en separent Paye et Recours#
##################################################
############################
#Creation bases train/test#
############################
b = basefinal
View(b)
set.seed(1)
n = length(b$MontantPaye)
A = sample(1:n, floor(n * 0.8))
train = (b[A,])
test = (b[-A,])
train.h2o <- as.h2o(train)
test.h2o <- as.h2o(test)

#############################################################################
#Partie où l'on choisit les variables explicatives/ variables a expliquer#
#############################################################################
colnames(train.h2o)
y.dep <- 170 #montant paye
y.dep2 <- 172 #montant recours
x.indep <- 1:169 #variables explicatives

#####
#GLM#
#####
regression.model_paye <-
  h2o.glm(
    y = y.dep,
    x = x.indep,
    training_frame = train.h2o,
    family = "gaussian"
  )
regression.model_recours <-
  h2o.glm(
    y = y.dep2,
    x = x.indep,
    training_frame = train.h2o,
    family = "gaussian"
  )

#Montant paye
h2o.performance(regression.model_paye)
summary(regression.model_paye)

predict.train <-
  as.data.frame(h2o.predict(regression.model_paye, train.h2o))
```

```
predict.reg <-
  as.data.frame(h2o.predict(regression.model_paye, test.h2o))

prediction_test = predict.reg$predict
prediction_train = predict.train$predict

realite_train = train$MontantPaye
realite_test = test$MontantPaye

err(prediction_train, realite_train)
err(prediction_test, realite_test)

#Montant recours
h2o.performance(regression.model_recours)
summary(regression.model_recours)

predict.train_rec <-
  as.data.frame(h2o.predict(regression.model_recours, train.h2o))
predict.reg_rec <-
  as.data.frame(h2o.predict(regression.model_recours, test.h2o))

prediction_test_rec = predict.reg_rec$predict
prediction_train_rec = predict.train_rec$predict

realite_train_rec = train$MontantRecours
realite_test_rec = test$MontantRecours

err(prediction_train_rec, realite_train_rec)
err(prediction_test_rec, realite_test_rec)


####################
#Forets aleatoires#
####################

#Random Search pour le montant paye
RF_params = list(
  mtries = seq(20, 80, by = 5),
  ntrees = 1:200,
  max_depth = 5:100
)

search_criteria2 <- list(strategy = "RandomDiscrete",
                         max_runtime_secs = 600)



RFrandompaye <- h2o.grid(
  "randomForest",
  x = x.indep,
  y = y.dep,
  grid_id = "RFrandompaye",
  training_frame = train.h2o,
```

```
  nfolds = 5,

  seed = 4,
  hyper_params = RF_params,
  search_criteria = search_criteria2
)


RFpaye = h2o.getGrid("RFrandompaye", 'rmse')

best_RF_model_idpaye <- RFpaye@model_ids[[1]]
rforest.model_paye <- h2o.getModel(best_RF_model_idpaye)

#Random Search pour le montant recours
RFrandomrecours <- h2o.grid(
  "randomForest",
  x = x.indep,
  y = y.dep2,
  grid_id = "RFrandomrecours",
  training_frame = train.h2o,
  nfolds = 5,

  seed = 4,
  hyper_params = RF_params,
  search_criteria = search_criteria2
)




RFpaye = h2o.getGrid("RFrandomrecours", 'rmse')

best_RF_model_idrecours <- RFrecours@model_ids[[1]]
rforest.model_recours <- h2o.getModel(best_RF_model_idrecours)

#montant paye
h2o.performance(rforest.model_paye)
h2o.varimp(rforest.model_paye)

predict2.train_paye <-
  as.data.frame(h2o.predict(rforest.model_paye, train.h2o))

predict2.test_paye <-
  as.data.frame(h2o.predict(rforest.model_paye, test.h2o))


prediction_foret_test_paye = predict2.test_paye$predict
prediction_foret_train_paye = predict2.train_paye$predict

err(prediction_foret_train_paye, realite_train)
err(prediction_foret_test_paye, realite_test)

#montant recours
```

```
h2o.performance(rforest.model_recours)
h2o.varimp(rforest.model_recours)

predict2.train_recours <-
  as.data.frame(h2o.predict(rforest.model_recours, train.h2o))

predict2.test_recours <-
  as.data.frame(h2o.predict(rforest.model_recours, test.h2o))


prediction_foret_test_recours = predict2.test_recours$predict
prediction_foret_train_recours = predict2.train_recours$predict

err(prediction_foret_train_recours, realite_train_rec)
err(prediction_foret_test_recours, realite_test_rec)



############################
#Gradient boosting machine#
############################
#Random Search pour le montant paye

gbm_params <- list(
  learn_rate = seq(0.1, 0.3, 0.01),
  max_depth = seq(2, 10, 1),
  ntrees = seq(1, 200, 1)
)
search_criteria2 <- list(strategy = "RandomDiscrete",
                         max_runtime_secs = 600)


gbm_gridpaye <- h2o.grid(
  "gbm",
  x = x.indep,
  y = y.dep,
  grid_id = "gbm_gridpaye",
  training_frame = train.h2o,
  nfolds = 5,
  seed = 1,
  hyper_params = gbm_params,
  search_criteria = search_criteria2
)
gbm_gridperfpaye = h2o.getGrid(grid_id = "gbm_gridpaye",
                               sort_by = "rmse",
                               decreasing = FALSE)
#choix du meilleur modele
best_gbm_model_idpaye <- gbm_gridperfpaye@model_ids[[1]]
gbm.model_paye <- h2o.getModel(best_gbm_model_idpaye)


#Random search pour le montant recours
```

```
gbm_gridrecours <- h2o.grid(
  "gbm",
  x = x.indep,
  y = y.dep2,
  grid_id = "gbm_gridrecours",
  training_frame = train.h2o,
  nfolds = 5,

  seed = 3,
  hyper_params = gbm_params,
  search_criteria = search_criteria2
)
gbm_gridperfrecours = h2o.getGrid(grid_id = "gbm_gridrecours",
                                  sort_by = "rmse",
                                  decreasing = FALSE)
?h2o.grid
#choix du meilleur modele
best_gbm_model_idrecours <- gbm_gridperfrecours@model_ids[[1]]
gbm.model_recours <- h2o.getModel(best_gbm_model_idrecours)



#prediction du montant paye
h2o.performance (gbm.model_paye)

predict3.train_paye <-
  as.data.frame(h2o.predict(gbm.model_paye, train.h2o))
predict3.test_paye <-
  as.data.frame(h2o.predict(gbm.model_paye, test.h2o))

prediction_gbm_test_paye = predict3.test_paye$predict
prediction_gbm_train_paye = predict3.train_paye$predict

err(prediction_gbm_train_paye, realite_train)
err(prediction_gbm_test_paye, realite_test)

#prediction du montant recours
h2o.performance (gbm.model_recours)

predict3.train_recours <-
  as.data.frame(h2o.predict(gbm.model_recours, train.h2o))
predict3.test_recours <-
  as.data.frame(h2o.predict(gbm.model_recours, test.h2o))

prediction_gbm_test_recours = predict3.test_recours$predict
prediction_gbm_train_recours = predict3.train_recours$predict

err(prediction_gbm_train_recours, realite_train_rec)
err(prediction_gbm_test_recours, realite_test_rec)
```

```
##############################
#Calcul des 9 montants totaux#
##############################
total_train = train$MontantTotal
total_test = test$MontantTotal

total_glm_train1 = prediction_train + prediction_train_rec
total_glm_test1 = prediction_test + prediction_test_rec

total_glm_train2 = prediction_train + prediction_foret_train_recours
total_glm_test2 = prediction_test + prediction_foret_test_recours

total_glm_train3 = prediction_train + prediction_gbm_train_recours
total_glm_test3 = prediction_test + prediction_gbm_test_recours

total_RF_train1 = prediction_foret_train_paye + prediction_train_rec

total_RF_test1 = prediction_foret_test_paye + prediction_test_rec

total_RF_train2 = prediction_foret_train_paye + prediction_foret_train_recours
total_RF_test2 = prediction_foret_test_paye + prediction_foret_test_recours

total_RF_train3 = prediction_foret_train_paye + prediction_gbm_train_recours
total_RF_test3 = prediction_foret_test_paye + prediction_gbm_test_recours

total_GBM_train1 = prediction_gbm_train_paye + prediction_train_rec
total_GBM_test1 = prediction_gbm_test_paye + prediction_test_rec

total_GBM_train2 = prediction_gbm_train_paye + prediction_foret_train_recours
total_GBM_test2 = prediction_gbm_test_paye + prediction_foret_test_recours

total_GBM_train3 = prediction_gbm_train_paye + prediction_gbm_train_recours
total_GBM_test3 = prediction_gbm_test_paye + prediction_gbm_test_recours


v = cbind(
  total_test,
  total_glm_test1,
  total_glm_test2,
  total_glm_test3,
  total_RF_test1,
  total_RF_test2,
  total_RF_test3,
  total_GBM_test1,
  total_GBM_test2,
  total_GBM_test3
)

err3(v)
#Le meilleur modele est celui qui predit chacun des montants avec une forêt aleatoire
plot(
  total_train,
```

```
  total_RF_train2,
  xlim = c(-1500, 10000),
  ylim = (c(-1500, 5500)),
  ylab = 'Montant total predit',
  xlab = 'Montant total reel (base train)'
)
plot(
  total_test,
  total_RF_test2,
  xlim = c(-1500, 10000),
  ylim = (c(-1500, 5500)),
  ylab = 'Montant total predit',
  xlab = 'Montant total reel (base test)'
)
lines(c(-1000, 10000), c(-1000, 10000), col = "red")

############################################################
#Modele de prediction par classification puis regression#
############################################################

#############################################################################################
#Construction des variables indicatrices de la presence d'un montant paye/recours#
#############################################################################################
Paye0 = as.factor(b$MontantPaye != 0)
Rec0 = as.factor(b$MontantRecours != 0)
b2 = cbind(b, Paye0, Rec0)
colnames(b2)

table(Paye0)
table(Rec0)
####################
#Base de test/train#
####################
set.seed(1)
n = length(b2$MontantPaye)
A = sample(1:n, floor(n * 0.8))


train = (b2[A,])
test = (b2[-A,])

train.h2o = as.h2o(train)
test.h2o = as.h2o(test)

################################
#Modele pour le montant Recours#
################################

y.dep = 175 #variable indicatrice de la presence ou non d'un montant recours
x.indep = 1:169

############################################################
#modele de prevision de la presence ou non d'un recours #
```

```
#########################################################
model_rec <-
  h2o.glm(
    y = y.dep,
    x = x.indep,
    training_frame = train.h2o,
    seed = 1122,
    nfolds = 5,
    family = 'binomial'
  )
## On essaye aussi h2o.gbm(y=y.dep, x=x.indep, training_frame = train.h2o,  seed = 1122,
#  nfolds=5)
## et h2o.randomForest(y=y.dep, x=x.indep, training_frame = train.h2o,  seed = 1122,
#  nfolds=5)

h2o.performance(model_rec)
h2o.varimp(model_rec)

recourstrain <- as.data.frame(h2o.predict(model_rec, train.h2o))
recourstest <- as.data.frame(h2o.predict(model_rec, test.h2o))

recourstr = recourstrain$predict
recourste = recourstest$predict

a = table(test$Rec0, recourste) #matrice de confusion

#taux d'erreur
(a[2, 1] + a[1, 2]) / sum(a)
#####################################
#prevision du montant le cas echeant#
#####################################
## Selection des sinistres de la base train pour lesquels il y a recours, afin
#  d'apprendre dessus
B = train$MontantRecours != 0
train2 = train[B,]
train2
train2.h2o = as.h2o(train2)
y.dep2 <- 172 #montant reocours

model_montrec <-
  h2o.randomForest(
    y = y.dep2,
    x = x.indep,
    training_frame = train2.h2o,
    seed = 1122,
    nfolds = 5
  )

montrecourstrain <-
  as.data.frame(h2o.predict(model_montrec, train.h2o))
montrecourstest <-
  as.data.frame(h2o.predict(model_montrec, test.h2o))
```

```
montrecourstr = montrecourstrain$predict
montrecourste = montrecourstest$predict


#calcul du montant recours predit
## Pour cela on multiplie le montant recours predit par l'indicatrice de la presence
#  ou non d'un recours

montrecoursfinaltrain = montrecourstr * (as.numeric(recourstr) - 1)
montrecoursfinaltest = montrecourste * (as.numeric(recourste) - 1)

#base de train
rec_train = train$MontantRecours
err(rec_train, montrecoursfinaltrain)
plot(
  rec_train,
  montrecoursfinaltrain,
  xlim = c(-3000, 0),
  ylim = c(-3000, 0)
)
abline(0, 1, col = 'red')

#base de test
rec_test = test$MontantRecours
err(rec_test, montrecoursfinaltest)
plot(
  rec_test,
  montrecoursfinaltest,
  xlim = c(-3000, 0),
  ylim = c(-3000, 0)
)
abline(0, 1, col = 'red')


##############################
#Modele pour le montant paye#
##############################
colnames(train)
y.dep3 = 174 #variable indiquant la presence ou non d'un montant paye
x.indep = 1:169 *
#############################################################################################
#Creation d'une seconde base train oversample afin d'avoir 50% chaque type de sinistres#
#############################################################################################
n1 = length(train$MontantPaye[train$Paye0 == FALSE])
n2 = length(train$MontantPaye[train$Paye0 == TRUE])
i = which(train$Paye0 == FALSE)
over = sample(i, size = (n2 - n1), replace = TRUE)
b3 = train[over,]
b4 = rbind(train, b3)
table(b4$Paye0) #la base contient 50% de chaque type de sinistres
```

```
##########################################################
#modele de prevision de la presence ou non d'un sinistre#
##########################################################
train3.h2o = as.h2o(b4) #on apprend sur la base oversample

glm.model_paye <-
  h2o.glm(
    family = 'binomial',
    y = y.dep3,
    x = x.indep,
    training_frame = train.h2o,
    seed = 1122,
    nfolds = 5
  )
#la encore on a teste glm binomial, randomForest et GBM

h2o.performance(glm.model_paye)
h2o.varimp(glm.model_paye)

payetrain <- as.data.frame(h2o.predict(glm.model_paye, train.h2o))
payetest <- as.data.frame(h2o.predict(glm.model_paye, test.h2o))

payetr = payetrain$predict
payete = payetest$predict

a = table(test$Paye0, payete) #matrice de confusion
a
#taux d'erreur
(a[2, 1] + a[1, 2]) / sum(a)
#####################################
#prevision du montant le cas echeant#
#####################################

y.dep4 <- 170 #montant paye
train4 = train[train$MontantPaye != 0,]
## On selectionne uniquement les sinistres dont le montant paye est different de 0, afin
#  d'apprendre dessus
train4.h2o = as.h2o(train4)

rf.model_montpaye <-
  h2o.randomForest(
    y = y.dep4,
    x = x.indep,
    training_frame = train4.h2o,
    seed = 1122,
    nfolds = 5
  )
#On a teste aussi random Forest, GLM gaussien et gamma

montpayetrain <-
  as.data.frame(h2o.predict(rf.model_montpaye, train.h2o))
montpayetest <-
  as.data.frame(h2o.predict(rf.model_montpaye, test.h2o))
```

```
montpayetr = montpayetrain$predict
montpayete = montpayetest$predict

################################
#calcul du montant paye predit#
################################

montpayefinaltrain = montpayetr * (as.numeric(payetr) - 1)
montpayefinaltest = montpayete * (as.numeric(payete) - 1)

#base de train
paye_train = train$MontantPaye
err(paye_train, montpayefinaltrain)
plot(paye_train, montpayefinaltrain)
#base de test
paye_test = test$MontantPaye
err(paye_test, montpayefinaltest)
plot(paye_test, montpayefinaltest)


#########################################################################
#Calcul du montant total predit en faisant la somme des deux modeles#
#########################################################################

total_train = train$MontantTotal
total_test = test$MontantTotal

total_train = montpayefinaltrain + montrecoursfinaltrain
total_test = montpayefinaltest + montrecoursfinaltest

err(total_train, total_train)
err(total_test, total_test)
plot(total_test,
     total_test,
     xlim = c(-2000, 5000),
     ylim = c(-2000, 5000))
plot(
  total_train,
  total_train,
  xlim = c(-2000, 5000),
  ylim = c(-2000, 5000)
)
abline(0, 1, col = 'red')
```

Bibliographie

1. Pierre AILLIOT - Cours sur les modèles linéaires (2016)
2. Alexandre BOUTEMY, Antoine GODEC et Mathilde MOUDEN - *Tarification d'un produit IARD et analyses d'impacts sur un marché concurrentiel*, Bureau d'étude (2016)
3. Fédération Française de l'Assurance, site internet : `http://www.ffa-assurance.fr/`
4. Documentation sur H2O - *H2O, Sparkling Water, Steam, and Deep Water Documentation*, site internet : `http://docs.h2o.ai/h2o/latest-stable/index.html`
5. Gilbert SAPORTA - *Probabilités, Analyse des données et Statistique*, Editions TECHNIP (2011)
6. Franck VERMET - Cours sur l'apprentissage statistique (2016)