Tarification des contrats d'assurance automobile

Léo Dutertre-Laduree, Axel Gardahaut, Guy Tsang (Groupe 2)

Résumé
Cet article présente l'intérêt des méthodes liées au Machine Learning pour affiner la tarification des contrats d'assurance automobile. Différentes familles d'algorithmes (non-ensembliste, boosting, bagging, stacking) seront confrontées en utilisant des stratégies de reéchantillonnage (SMOTE, down-sampling, up-sampling) pour l'apprentissage de ces modèles. Dans un premier temps, la démarche du projet est présentée afin de justifier les choix effectués dans le traitement et la construction des modèles de prédiction. Puis dans un second temps, le fonctionnement de certains modèles (régression logistique pénalisée et Extreme Gradient Boosting) sera explicité (intuition, mécanisme, avantages et inconvénients) afin de faire le lien entre la démarche effectuée et les performances obtenues. Une partie s'attardant sur les résultats conclura cet article et montrera qu'une hiérarchie entre les différentes familles se dessine pour ce cas de figure.
Mots-clés : *Machine Learning, Classification Binaire, Scoring, Régression Logistique Pénalisée, XGboost, LightGBM*

1 Introduction

Une société d'assurance souhaite affiner sa capacité à tarifer ses contrats automobiles avec ses clients. L'objectif est de faire payer à chaque assuré son « juste prix ». Ainsi, à partir d'un jeu de données sur ses clients, le but de cette étude sera de construire un modèle qui prédit, pour chaque assuré, sa probabilité de déposer une réclamation au cours de la prochaine année. Plus cette probabilité est élevée, plus la tarification sera élevée pour l'assuré.

Ceci constitue un problème classique de modélisation aboutissant à un score. L'évaluation des performances doit se faire à l'aide d'une métrique adaptée à un contexte où la variable cible est déséquilibrée. En effet, celle-ci présente uniquement 3.67% de labels positifs. Ceci justifie également l'intérêt de méthodes de reéchantillonnage (SMOTE [1], down-sampling, up-sampling) lors de l'apprentissage des modèles.

Dans un premier temps, les données seront présentées afin de positionner le contexte. Puis, dans un second temps, la démarche du projet sera décrite étape par étape afin de justifier les choix effectués pour améliorer le pouvoir prédictif des régresseurs. Enfin, certains modèles de prédiction utilisés seront détaillés et les résultats obtenus seront explicités.

2 Données

Les données sont fournies par la société d'assurance. Elles comportent une base d'apprentissage et de test. L'apprentissage et la validation des modèles doivent se faire sur la première base tandis que base de test ne sert uniquement à tester la performance du modèle retenu. Des valeurs manquantes sont présentes dans les deux échantillons.

Parmi les prédicteurs, on retrouve :

- des variables concernant l'assuré en personne ("ind"),
- des variables concernant la région de l'assuré ("reg"),
- des variables concernant la voiture de l'assuré ("car"),
- des variables calculées ("calc").

La variable cible ("target" dans la base) est binaire et indique si une réclamation a été déposée ("1") par l'assuré ou non ("0"). Cette variable est déséquilibrée, avec moins de 4% de labels positifs. Chaque ligne de la base de données correspond à un assuré automobile. Les intitulés des colonnes sont anonymisées.

3 Démarche

3.1 Généralités

L'objectif de ce problème de classification est de réaliser un scoring des clients. Le score correspond à la probabilité pour chaque client de déposer une réclamation dans l'année à venir. La métrique retenue pour évaluer la performance prédictive des modèles est celle du Coefficient Normalisé de Gini. Le coefficient de Gini permet de comparer la proportion cumulée des labels positifs prédits avec la proportion théorique.

Le coefficient de Gini est étroitement lié à l'aire sous la courbe ROC (AUC) :

$$\text{Gini} = 2 \times \text{AUC} - 1 \in [0, 1]$$

Ainsi, maximiser le coefficient (normalisé) de Gini revient à maximiser l'AUC.

3.2 Benchmark d'ouverture

Il est intéressant de placer un benchmark afin d'avoir une valeur de la métrique objectif (Coefficient Normalisé de Gini) à

1. Synthetic Minority Over-Sampling Technique

dépasser. Le score obtenu par un Light GBM [2] est généralement élevé et rapidement obtenu. Celui-ci est de : 0.2719 par validation croisée sur 5 blocs (5fCV). La réalisation d'un score obtenu par la méthode du Light GBM est effectuée à chaque étape du traitement : sélection de variables, regroupement par classe, normalisation des données, paramétrisation des modèles. La variation de ce score lors des ces différentes étapes permettra d'évaluer leur pertinence.

3.3 Pré-traitement des données

Le pré-traitement des données consiste essentiellement à la gestion des valeurs manquantes. Plusieurs variables ont été identifiées dans les statistiques descriptives (*Figure 1*). Selon le taux de valeurs absentes, la manipulation appliquée sera différente.

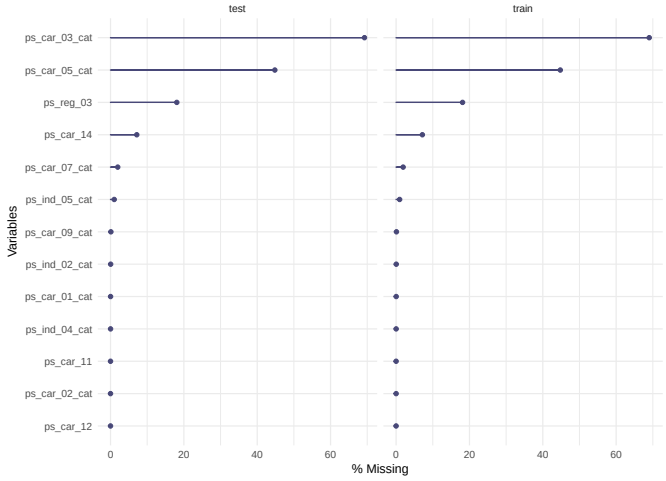

Figure 1 – Valeurs manquantes par variable

Deux variables explicatives qualitatives présentent trop de valeurs manquantes (plus de 40%) ce qui pourrait conduire à supprimer ces variables de l'étude. Cependant, la comparaison de ces variables avec la cible montre que la perte de ces valeurs manquantes serait une perte d'information. Le nombre élevé de valeurs manquantes ne permet pas une imputation sophistiquée pour ces variables. Les valeurs manquantes de ces deux variables qualitatives sont remplacées par une valeur spéciale (-999 par exemple), permettant ainsi de conserver l'information issue de l'absence de valeur.

Pour le reste des variables, une imputation par forêt aléatoire est faite. Le procédé d'imputation est le suivant :

- Utiliser uniquement l'échantillon d'apprentissage pour construire la forêt aléatoire.
- Modéliser la variable étudiée en s'assurant qu'il s'agisse du bon type d'arbres (classification ou régression) en utilisant les autres variables comme régresseurs.
- Prédire les valeurs de la variable étudiée pour l'ensemble des observations manquantes des deux échantillons (apprentissage et test).
- Remplacer les valeurs manquantes et s'assurer qu'il n'y a plus de valeurs manquantes pour la colonne traitée.

Suite aux imputations faites, le Light GBM de référence renvoie un score de Gini de 0.2721, toujours par validation croisée sur 5 blocs. On note une augmentation par rapport au benchmark bien que la différence soit trop faible pour en tirer des conclusions.

3.4 Sélection des variables

Le Light GBM effectué après les imputations permet également d'identifier l'importance des variables dans le pouvoir prédictif du modèle. L'importance d'une variable peut être mesurée de différentes manières. Celle adoptée ici est le « Gain [3] » [4], i.e. la contribution de la variable au modèle. Ainsi, plus une variable contribue au pouvoir prédictif d'un modèle, plus son « Gain » associé sera élevé. Arbitrairement, on garde la première moitié des variables les plus contributives au modèle.

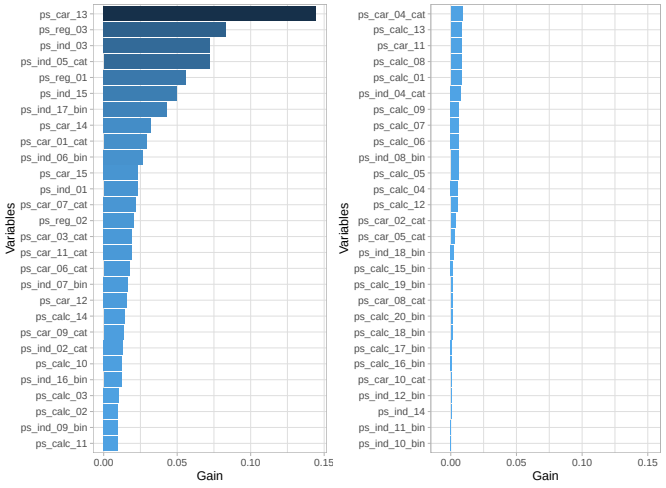

Figure 2 – Importance des variables

En plus des importances issues du LightGBM, une forêt aléatoire (sans optimisation des hyper-paramètres) est construite pour réaliser une seconde liste d'importance en termes de « Permutation » [5]. L'idée de base de cette notion est de considérer une variable importante si elle a un effet positif sur la précision de la prédiction. Ainsi, une seconde liste de variables est identifiée en sélectionnant la moitié des variables les plus importantes au modèle au sens de la notion de « Permutation ». L'union de ces deux listes constitue la liste finale des variables retenues pour la suite. Au total, ce sont 35 régresseurs sur les 57 initiaux qui sont retenus.

3.5 Traitement des données

Plusieurs variables présentent des modalités rares (effectif inférieur à 5%). Il est difficile d'obtenir des résultats d'estimation robustes pour ces modalités. Il est donc nécessaire de fu-

2. Gradient Boosting Machine
3. COMPLÉTER LA DÉFINITION
4. [1] Abu-Rmileh, A. « The multiple faces of Feature importance in XGboost », 2019
5. [11] R-bloggers, *Rstats on Pi : Predict/Infer*, « Be aware of bias in RF variable importance metrics », 2018

sionner certaines modalités. Cependant, les modalités de ces variables ont été anonymisées, il n'est donc pas possible de les regrouper par l'intermédiaire d'une certaine expertise. Dans le but de fusionner ces modalités rares, deux stratégies ont été testées : l'Analyse Factorielle de Données Mixtes (AFDM) et les tableaux de contingence.

La stratégie de fusion peut passer par une projection des modalités d'une variable sur le premier plan factoriel d'une AFDM. Chaque modalité rare sera alors associée à la modalité fréquente la plus proche d'elle (et si possible projetée dans la même direction), ou un ensemble de modalités rares peuvent se regrouper pour former un groupe supplémentaire (clusters).

La stratégie de fusion peut également passer par des tableaux de contingence (*Figure 3*) qui croisent les variables avec la cible. Les modalités rares sont fusionnées soit entre elles si elles se comportent de façon similaire, soit avec une modalité fréquente si le comportement s'en rapproche.

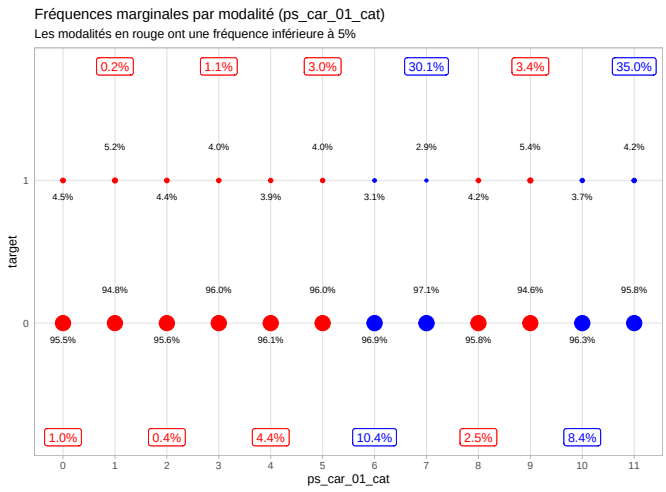

Figure 3 – Exemple de projection de modalités par croisement

* * *

Une variable présente un grand nombre de modalités (104). Plutôt que de fusionner ces modalités, on peut s'en servir comme régresseur unique d'une régression logistique avec la variable cible comme variable à expliquer. Les modalités seront alors remplacées par les coefficients estimés et la variable deviendra alors numérique et continue.

Le Light GBM fournit un score de 0.2729 (5fCV) pour la stratégie par AFDM et de 0.2757 (5fCV) pour la stratégie par tableaux croisés. Ces deux scores sont supérieurs à celui du benchmark. Cependant, nous retiendrons la deuxième stratégie car elle permet l'obtention d'un meilleur score par rapport à la première.

3.5.1 Normalisation des données

Certaines variables continues ont une distribution qui ont un fort kurtosis (i.e. avec un haut sommet) et qui n'est pas centrée. Il serait donc intéressant de tenter de les transformer à travers différentes méthodes (box-cox, logarithme, racine carré).

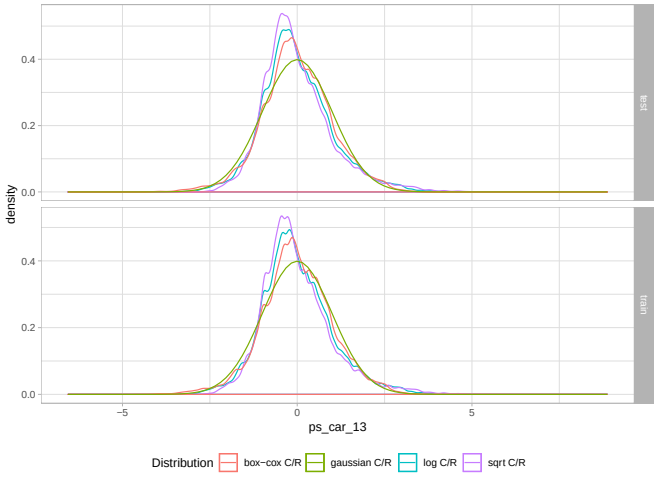

Figure 4 – Exemple de transformation

Il est notable que la transformation de box-cox est la meilleure pour normaliser cette distribution (*Figure 4*). Cette étude est menée pour les variables continues.

En réutilisant un Light GBM pour mesurer les conséquences des transformations réalisées, il est noté que la métrique diminue, suite à chacune des deux stratégies de fusion de modalités. On passe respectivement à un score de Gini de 0.2729 pour l'AFDM et de 0.2752 pour les tableaux croisés. Ces diminutions sont minimes mais la transformation des variables n'est pas obligatoire. Par conséquent, cette étape est laissée de côté pour la suite.

À la fin de cette étape, le traitement se résume à la fusion de modalités en utilisant des tableaux de contingence.

3.6 Étude des corrélations et dépendances

L'objectif de cette étape est d'étudier les liens entre variables. On évalue les corrélations et les dépendances entre variables puis avec la cible. Ceci permet de détecter des problèmes de colinéarité et ou dépendance entre prédicteurs :

- les liens entre variables continues se font grâce aux corrélations,
- les liens entre les variables continues et la cible se font grâce à un test d'ANOVA [6],
- les liens entre variables catégorielles (et la cible) se font grâce aux V de Cramer [7].

Les variables qui dépassent les seuils arbitrairement fixés ou qui ne passent pas le test d'ANOVA sont laissées de côté. L'étude n'a pas abouti à de suppression de variables.

3.7 Paramétrisation des modèles

Une dizaine d'algorithmes sont testés pour tenter d'avoir le meilleur pouvoir prédictif tels que : l'Analyse Discriminante Linéaire (LDA), les K plus proches voisins (KNN), la régression

6. Analysis of Variance
7. Le V de Cramer est la statistique du χ^2 standardisée de façon à ce qu'elle varie entre 0 et 1

logistique (pénalisée ou non), la Machine à Vecteurs de Support (SVM)), puis des modèles de bagging (Forêt Aléatoire) et de boosting (ADAboost, GBM, XGBM, LGBM).

L'ensemble des modèles ensemblistes et la régression logistique pénalisée sont optimisés selon leurs paramètres de hyper-paramètres [8]. La méthode choisie pour effectuer cette hyper-paramétrisation est celle du « grid search ». L'idée étant d'établir au préalable une liste arbitraire non exhaustive de valeurs des hyper-paramètres puis de retenir la combinaison qui maximise le score de Gini pour chaque modèle.

Le fait que les données soient déséquilibrées pose la question du ré-échantillonnage. Ainsi, pour l'ensemble des modèles, les stratégies de down-sampling, up-sampling et SMOTE sont testées en plus du jeu de données d'apprentissage initial. L'hyper-paramétrisation via la méthode du « grid search »est donc effectuée pour ces quatre cas.

4 Méthodes utilisées

L'intuition et le fonctionnement de trois méthodes sont détaillés dans cette section. La régression logistique pénalisée sera présentée en première, suivie de deux méthodes de boosting : le XGboost et le LightGBM. La régression logistique pénalisée a des principes radicalement différents des deux méthodes de boosting.

4.1 Régression logistique pénalisée

La pénalisation de la métrique optimisée pour estimer les paramètres est un principe applicable à toutes les méthodes d'estimation où l'on a des combinaisons de variables avec des coefficients à estimer (réseaux de neurones, SVM linéaire, etc). L'idée étant que les hypothèses classiques de la régression linéaire permettent d'obtenir le meilleur estimateur non biaisé i.e. l'estimateur sans biais de variance minimale. Dans les faits on obtient un estimateur de faible biais mais de variance plus élevée car ces hypothèses ne sont jamais totalement respectées.

Or, en considérant l'écart quadratique moyen (EQM) qui est la fonction de perte la plus largement utilisée on a :

$$\mathbb{E}\left[(y^* - \hat{y}^*)^2\right] = \sigma^2 + \left(\mathbb{E}\left[\hat{y}^*\right] - y^*\right)^2 + \mathbb{E}\left[\left(\hat{y}^* - \mathbb{E}\left[\hat{y}^*\right]\right)^2\right]$$

On a respectivement σ^2 la variance incompressible de la cible Y, le biais au carré et la variance de la prévision. Ainsi si on s'autorise un certain biais, on peut imaginer qu'il est possible de réduire plus que proportionnellement la variance et ainsi gagner en pouvoir prédictif. En pratique, on cherche le meilleur compromis entre le biais et la variance.
Dans le cadre classique de la discrimination binaire par la régression logistique, on suppose (x_i, y_i) *iid* d'une même distribution inconnue.
On modélise $\mathbb{P}(Y = 1|X = x_i) = \frac{1}{1+e^{-(\beta' x)}} = p_i$.

Une approche consiste à résoudre le problème suivant :

$$\left|\begin{array}{l} \min\limits_{\beta} -\dfrac{1}{n}\sum\limits_{i=1}^{n}\left[y_i \ln p_i + (1-y_i)\ln(1-p_i)\right] + R(\beta) \\ \text{sous contraintes : } R(\beta) \leq \tau \end{array}\right.$$

qui s'écrit également :

$$\min_{\beta} -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln p_i + (1-y_i)\ln(1-p_i)\right] + \lambda R(\beta)$$

où l'on aura pris soin de centrer et réduire les variables afin de leur accorder la même importance.

Il y a 3 types de pénalisation utilisés régulièrement :

1. en norme L^2 on parle alors de régression ridge : $R(\beta) = \sum_{j=1}^{p} \beta_j^2$
2. en norme L^1 on parle alors de régression LASSO : $R(\beta) = \sum_{j=1}^{p} |\beta_j|$
3. ElasticNet qui est une combinaison linéaire des 2 : $R(\beta) = \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha)\sum_{j=1}^{p} \beta_j^2$

À noter que λ est un paramètre du modèle qu'il faut calibrer également.
On a une relation inverse entre τ et λ. Lorsque $\lambda = 0$ on retrouve les MCO et plus λ est grand, plus les coefficients sont contraints (plus de biais, moins de variance). On peut voir les coefficients se déformer en fonction de λ.
La régularisation dans la régression ridge est plus souple que dans la régression LASSO qui fait converger rapidement les coefficients des variables moins explicatives vers 0.
Il peut s'agir d'une procédure de selection de variables. En revanche, la regression LASSO choisit arbitrairement une variable parmi un groupe de variables corrélées tandis que la régression ridge permet de pondérer les influences.
L'Elastic Net permet de combiner les avantages des deux méthodes en contrepartie d'une complexité accrue.

Pour choisir λ, on procède généralement par apprentissage-test lorsque c'est possible ou par validation croisée K-folds lorsque la base est petite, en minimisant un critère d'erreur. Des formules basées sur des hypothèses simplificatrices existent également.

Pour la régression LASSO, il n'y a pas de formulation explicite de β (car la fonction valeur absolue n'est pas différentiable) ainsi on procède de manière itérative (LARS, Forward Stagewise). On fait varier de manière sélective les coefficients des variables. En grande dimension, on privilégiera la descente de gradient ou ses variantes pour l'estimation de β.

4.1.1 Performance

Resampling	down	up	SMOTE	aucun
Gini Normalisé[a]	0.2566	0.2576	0.2483	0.2573

a. obtenu par validation croisée 5 blocs

8. Les algorithmes KNN et SVM ne sont pas optimisés faute du temps d'exécution nécessaire au calcul de matrices immenses de distance

4.2 Gradient Boosting Machine

4.2.1 XGboost

L'algorithme XGboost est l'abréviation de e**X**treme **G**radient **boost**ing, approche introduite par Friedman.
Comme la plupart des méthodes basées sur des arbres de décisions, il peut être utilisé en classication comme en régression.
L'approche combine 2 mécanismes : le boosting et la descente de gradient.
Le boosting est une méthode ensembliste qui consiste à agréger des classifieurs élaborés séquentiellement sur un échantillon d'apprentissage dont les poids des individus sont corrigés au fur et à mesure, les individus mal classés se voyant affecter un poids plus important. Les classifieurs sont pondérés selon leurs performances.
D'autre part, la descente du gradient est une technique itérative qui permet d'approcher la solution d'un problème d'optimisation.
Enfin, outre les deux mécanismes précédents, l'approche possède aussi des paramètres liés à l'utilisation d'arbres comme classifieurs.
Ainsi, l'algorithme XGboost dépend d'un grand nombre d' hyperparamètres :

1. Caractéristiques des arbres individuels : Profondeur T, effectifs minimums de coupe
2. Constante d'apprentissage η
3. Nombre d'arbres K
4. Taux d'échantillonnage des individus β
5. Échantillonnage des variables

On optimise les hyperparamètres par grid search.
L'algorithme propose également une gestion efficace des valeurs manquantes : « sparcity-aware split finding »(une petite référence ?).
La prévision se base sur K arbres construits de façon itérative. À chaque étape on ajoute celui qui minimise une fonction objectif régularisée $\mathcal{L}$ qui est la somme d'une fonction de perte et d'une fonction de régularisation.

$$\mathcal{L} = \sum_i l\left(\hat{y}_i, y_i\right) + \sum_k \Omega\left(f_k\right)$$

avec $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$, γ étant le coefficient de pénalisation par rapport à la profondeur T, λ le coefficient par rapport qui pondère l'impact du poids w.
La fonction de perte est généralement la déviance binomiale en classification binaire :

$$-\frac{1}{n}\sum_{i=1}^{n}\left[y_i \ln p_i + (1-y_i)\ln\left(1-p_i\right)\right]$$

On utilise la descente de gradient pour optimiser cet objectif à chaque étape.
L'optimisation est effectuée sans les valeurs manquantes pour le feature considéré, les individus ayant le feature manquant pour le noeud optimal sont affectés à la direction majoritaire.
L'algorithme réalise un one-hot encoding [9] des variables catégorielles pour pouvoir calculer le gain.
Pour accélérer l'optimisation, une approximation d'ordre 2 de la fonction objectif est faite.

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}\left[l\left(y_i, \hat{y}^{(t-1)}\right) + g_i f_t\left(\mathbf{x}_i\right) + \frac{1}{2}h_i f_t^2\left(\mathbf{x}_i\right)\right] + \Omega\left(f_t\right)$$

avec

$$g_i = \partial_{\hat{y}(t-1)} l\left(y_i, \hat{y}^{(t-1)}\right) \quad \text{et} \quad h_i = \partial^2_{\hat{y}(t-1)} l\left(y_i, \hat{y}^{(t-1)}\right)$$

Considérer l'ensemble des noeuds possibles est impossible, une heuristique gloutone [10] basée sur un ensemble de noeuds éligibles est couramment utilisée.
L'algorithme XGboost est dit "Leaf-wise", il est construit en profondeur.
On peut determiner le poids optimal de chaque noeud en différenciant la fonction objectif que l'on utilise également comme une mesure d'impureté.

$$\mathcal{L}_{\text{split}} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

où I_R et I_L sont les noeuds droit et gauche issus du noeud I.
Une optimisation de l'algorithme d'évaluation des coupes basée sur les quantiles et le nombre de noeuds voulus pour chaque feature (basé sur un scoring via les gradiants de premier ordre) est utilisée en grande dimension.
On peut ajouter de l'aléa dans l'échantillon test en utilisant seulement un échantillon des données pour construire les arbres. Ceci a l'avantage d'accélérer les calculs et de se prémunir contre le surapprentissage.
Il est évident que plus le nombre d'itérations est grand, plus on s'approche de la solution optimale, cependant celà s'effectue au détriment du temps de calcul.
Le compromis pour $\eta \in]0, 1[$ est réalisé entre vitesse de convergence et surapprentissage.
Si l'algorithme est très efficace, souple avec le choix des fonctions de coûts, adaptables à différents problèmes, permettant de prendre efficacement en compte des interactions non linéaires, cela conduit à une procédure difficilement interprétable (bien qu'on puisse calculer des mesures d'importance des variables), lourde en mémoire et intensive avec beaucoup de paramètres à optimiser et un risque de surapprentissage qui peuvent rendre sa mise en œuvre moins aisée.

4.2.2 LightGBM

L'algorithme LightGBM est une autre implématation du Gradient Boosting Tree, très efficace en mémoire et en temps de calcul.
Il s'agit également d'un algorithme « Leaf-wise ».
Les deux principales différences avec XGboost sont des approches heuristiques pour diminuer drastiquement le temps

9. consistant à construire autant de variables binaires que de valeurs possibles prises par la variable catégorielle (binarisation)
10. algorithme qui visite itérativement son voisinage pour optimiser l'objectif

de calcul tout en restant très performant.
Une première approche pour reduire le nombre d'observations à prendre en compte pour calculer le split est appelée **GOSS** : « Gradient-based One-Side Sampling ».
L'idée étant que les observations avec un faible gradient sont peu informatives, les a% premières observations triées par gradient decroissants sont gardées (ensemble A) puis on prend aléatoirement b% du reste (ensemble B).
Afin de ne pas trop modifier la distribution des observations, on multiplie les gradients des observations échantillonnées par un facteur $\frac{1-a}{b}$ dans la mesure de gain de variance $\tilde{V}_j$.

$$\tilde{V}_j(d) = \frac{1}{n}\left(\frac{\left(\sum_{x_i \in A_l} g_i + \frac{1-a}{b}\sum_{x_i \in B_l} g_i\right)^2}{n_l^j(d)} + \frac{\left(\sum_{x_i \in A_r} g_i + \frac{1-a}{b}\sum_{x_i \in B_r} g_i\right)^2}{n_r^j(d)}\right)$$

avec A_l et B_l correspondent au subset des parties A et B au point de coupure d pour le feature j dans la feuille gauche, de même pour A_r et B_r pour le côté droit ; n_l^j et n_r^j sont les nombres d'observations respectifs.

Une borne supérieure de l'erreur commise est connue théoriquement, on peut déduire de celle-ci que si le nombre d'observations est grand et que le seuil selectionné n'induit pas un déséquilibre des observations important alors l'approximation est excellente.
L'erreur de généralisation via GOSS (comparaison avec le gain de variance sur toutes les données ("vraie variance de la distribution")) est très bonne si l'erreur du GOSS est déja bonne sinon un sampling classique des observations peut permettre de mieux apprendre.
On comprend pourquoi on n'a pas intérêt à échantillonner le training set pour cet algorithme, ce qui transparait bien dans nos résultats.

La deuxième approche est appélée **Exclusive Feature Bundling** (EFB).
Le but est d'aggréger intelligement les variables qui ne sont pas ou peu mutuellement nulles simultanement.
Il s'agit d'un problème NP-difficile donc une heurtistique gloutone est utilisée.
On aggrège les variables de manière itérative dans des "bundles" jusqu'à obtenir un seuil de conflit prédéfini.
On parvient à diminuer drastiquement le nombre de features.
C'est la réduction à la fois du nombre d'observations et du nombres de features à scanner qui rend cet algorithme si rapide.
Cette approche est plus efficace que le Stochastic Gradient Boosting.

4.2.3 Performance

Resampling	down	up	SMOTE	aucun
Gini Normalisé[a]	0.2678	0.2728	0.2687	0.2778

a. obtenu par validation croisée 5 blocs

Le modèle sans resampling se classe premier en validation sans compter les modèles de stacking construits par la suite.

5 Résultats

Plusieurs modèles ont été paramétrés par « grid search » pour chaque stratégie de resampling puis évalués en validation croisée 5 blocs. La *Figure 5* répertorie les résultats de performance obtenus.

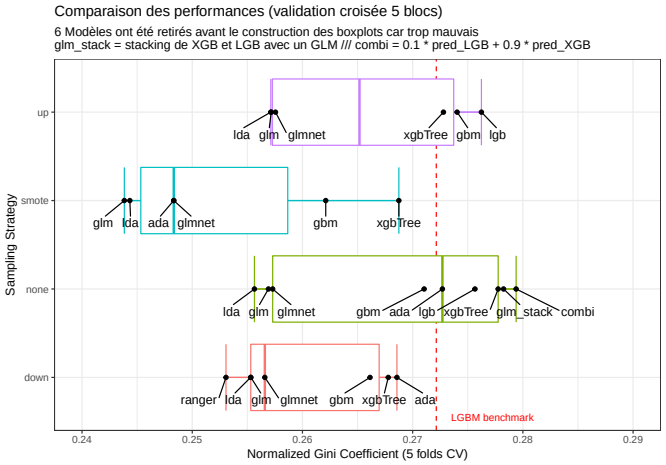

Figure 5 – Performance de validation des modèles

Certains modèles construits tels que les arbres de décision ne figurent pas dans la figure parce que leur performance est trop faible. Les modèles de stacking (glm_stack et combi) ont été construits en choisissant les deux modèles les plus performants. Le méta-classifieur du stacking est une régression logistique et le modèle "combi" correspond à un classifieur qui combine linéairement les scores issus de la validation croisée des modèles. Pour les deux modèles de stacking, seules les prédictions issues du Light GBM et du XGboost (sans reéchantillonnage) sont utilisées.

Bien que les performances des modèles ayant appris sur des données sur-échantillonnées sont équivalentes à celles des modèles sans reéchantillonnage, deux raisons vont en faveur de la seconde solution :

- le sur-échantillonnage engendre une augmentation des temps de calcul (duplication des individus, apprentissage sur quasiment deux fois plus de données)
- le meilleur modèle en validation croisée est obtenu sans reéchantillonnage.

Par conséquent, seule les modèles sans reéchantillonnage seront retenus afin de tracer leur courbe ROC (*Figure 6*).

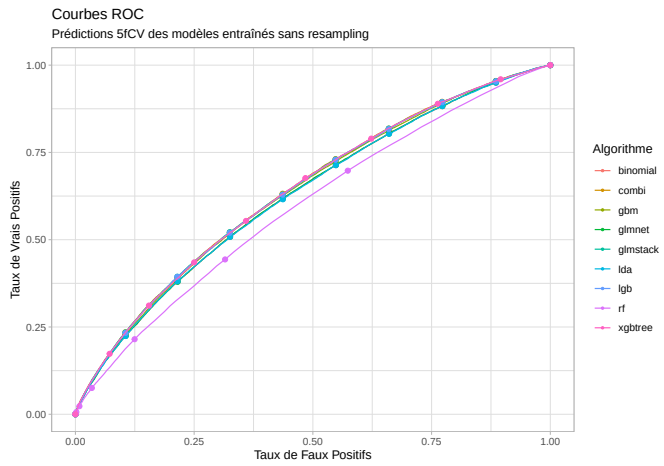

Figure 6 – Courbes ROC des modèles sans reéchantillonnage

Finalement, le modèle combiné dépasse le reste des modèles avec un coefficient normalisé de Gini en validation croisée de 0.2794. L'application à l'échantillon test renvoie une valeur de 0.2881 [11].

Pour passer de la prédiction à la classification [12], il est nécessaire de fixer un seuil pour lequel les scores en-dessous de celui-ci correspondent à un label "0" pour les individus concernés et "1" sinon. L'obtention de ce seuil dépend de l'objectif à maximiser (le score F1 semble être le plus approprié dans ce contexte (données déséquilibrées)) et doit se faire sur les prédictions issues de la validation croisée. Le seuil retenu est ensuite utilisé sur l'échantillon test afin de vérifier la stabilité entre la validation et le test.
On obtient les résultats suivants :

Table 1 – Résultats pour un seuil défini

	F1	Recall
Validation	0.1176	0.2482
Test	0.1158	0.2488

La matrice de confusion associée à l'échantillon test est :

Table 2 – Matrice de confusion

		Observé	
		0	1
Prédiction	0	152 542	4 827
	1	19 596	1 599

blabla

Références

[1] Abu-Rmileh, A. The multiple faces of 'feature importance' in xgboost. Towards data science, 8 Février 2019.

[2] Chen T., Guestrin C. Xgboost : A scalable tree boosting system, university of washington.

[3] Friedman, J. Greedy function approximation : A gradient boosting machine, 2001.

[4] Gandhi, R. Boosting algorithms : Adaboost, gradient boosting and xgboost. Hacknernoon, 5 Mai 2018.

[5] Guolin Ke et al., Microsoft Research Peking University. Lightgbm : A highly efficient gradient boosting decision tree.

[6] Hale, J. Smarter ways to encode categorical data for machine learning. Towards data science, 11 Septembre 2018.

[7] Harrell, F. Classification vs. prediction. Statistical Thinking, 2019.

[8] Hastie T., Tibshirani R., and Friedman J. The elements of statistical learning, 2001.

[9] Juhi. Gini coefficient and lorenz curve explained. Towards data science, 6 Mars 2019.

[10] Rakotomalala.R. Université de Lyon 2, http ://eric.univ-lyon2.fr/~ricco/cours/slides/regularized_regression.pdf.

[11] Rstats on pi : predict/infer. Be aware of bias in rf variable importance metrics. R-bloggers, 19 Juin 2018.

11. À titre informatif, l'AUROC (Area Under the Receiver Operating Characteristic curve) du modèle sur l'échantillon test est de 0.6440.
12. [7] Harrell, F. Classification vs. prediction. Statistical Thinking, 2019