Projet data, pp. 1–8, 2025
Disponible aussi sur:
https://github.com/tguigue30/Projet_data_ML.git

Modélisation prédictive sous Python du risque de montant sinistre en assurance automobile : une approche par apprentissage automatique.

Théo Guigue[1], Marie-Léa Fouché[2]

1. *Faculté d'économie, Université de Montpellier*
2. *Faculté d'économoe, Univesité de Montpellier*

Résumé

Ce projet consiste en l'analyse et la modélisation d'un jeu de données d'assurance automobile, en utilisant des méthodes avancées de machine learning.

Mots-clés:

traitement des données, python, pandas, économétrie, statistiques, assurance

Contexte et objectifs

Le secteur de l'assurance automobile génère une quantité importante de données relatives aux contrats, aux véhicules, aux conducteurs et aux sinistres. Traditionnellement, des approches économétriques classiques étaient utilisées pour modéliser les montants des sinistres ou la fréquence des accidents, mais ces méthodes peinent à capturer des relations complexes et non linéaires présentes dans les données massives modernes. Dans ce contexte, l'objectif principal de cette étude est de mettre en œuvre des techniques modernes d'apprentissage automatique afin de prédire de manière précise le montant potentiel d'un sinistre automobile, transformé ici en un problème de classification multi-classes. À travers une approche systématique comprenant le prétraitement des données, le traitement du déséquilibre des classes, la sélection et l'optimisation des modèles, et la comparaison de différentes méthodologies de machine learning.

Features ingeneering

Construction de la variable cible *y*

Dans un premier temps, la variable cible `claim_amount_winsorized` a été discrétisée en cinq classes de largeur fixe à l'aide de la fonction `pd.cut()`, afin de transformer un problème initial de régression complexe et asymétrique en un problème de classification multi-classes plus simple à modéliser. Les classes créées ont été intitulées 'Très faible', 'Faible', 'Moyen', 'Élevé' et 'Très élevé', chacune correspondant à une tranche de montants winsorisés. Cette approche a été privilégiée en raison de la forte asymétrie observée dans la distribution de la variable continue, qui aurait conduit à une mauvaise performance du modèle en cas d'utilisation directe pour la régression. Il est à noter qu'ici la modélisation s'intéresse à connaître le coût d'une prestation pour l'assurance, certaines valeurs négatives (remboursement) furent retirés en triant le dataset original sur des valeurs positives. En effet, les classes étaient trop bruitées sans cette condition.

La suppression d'une variable indicatrice via `drop_first=True` est indispensable pour éviter les problèmes de multicolinéarité même sur les modèles d'apprentissage par arbres comme XGBoost ou LightGBM. Cela a été fait pour éviter des problèmes de non-stabilité.

Analyse de la distribution des classes et algorithme SMOTE

Une analyse de la distribution des observations par classe a révélé un déséquilibre important, notamment une surreprésentation de la classe ''Très faible'' et une sous-représentation marquée des classes ''Élevé'' et ''Très élevé''. Ce déséquilibre aurait pu biaiser l'apprentissage du modèle en le poussant à favoriser systématiquement les classes majoritaires. Afin de pallier ce problème, la technique SMOTE (Synthetic Minority Over-sampling Technique) a été utilisée. SMOTE génère de nouvelles observations synthétiques pour les classes minoritaires en interpolant linéairement entre des points existants, selon la formule $x_{\text{new}} = x_i + \lambda(x_{zi} - x_i)$, où λ est un coefficient aléatoire compris entre 0 et 1. Cette méthode permet de densifier l'espace des données sans simplement dupliquer les observations, réduisant ainsi le risque d'overfitting tout en rééquilibrant l'ensemble d'apprentissage.

Préparation des Features X

Avant l'application de SMOTE, les étapes de préparation des données ont inclus l'imputation des valeurs manquantes présentes dans les features à l'aide de la méthode 'SimpleImputer' avec la moyenne comme stratégie d'imputation, cette étape étant nécessaire car la plupart des algorithmes de machine learning, dont Random Forest et XGBoost, ne tolèrent pas la présence de valeurs manquantes. Par la suite, les labels correspondant aux classes cibles ont été encodés sous forme numérique grâce au 'LabelEncoder' de Scikit-Learn, afin d'être compatibles avec les algorithmes d'apprentissage supervisé utilisés ultérieurement. Après la phase de préparation, SMOTE a été appliqué sur l'ensemble des features imputés et sur les labels encodés, garantissant ainsi un échantillon équilibré entre les différentes classes avant toute séparation en train et test. Cela a permis d'assurer que les modèles entraînés n'aient pas un biais naturel envers la classe majoritaire.

Optimisation séquentielle bayésienne des hyperparamètres

Enfin, une phase d'optimisation des hyperparamètres a été réalisée à l'aide de la bibliothèque Optuna, en utilisant une cross-validation stratifiée à cinq plis afin de garantir la robustesse de la sélection des paramètres et de limiter l'overfitting. Les meilleurs hyperparamètres trouvés ont ensuite été utilisés pour entraîner un modèle final, qui a été évalué sur un ensemble de test indépendant à l'aide de métriques classiques telles que l'accuracy, le classification report détaillé (précision, rappel, F1-score) et la matrice de confusion. Cette méthode repose sur un cadre d'optimisation séquentielle basé sur une approche bayésienne, et plus précisément sur l'algorithme Tree-structured Parzen Estimator (TPE). Mathématiquement, le problème d'optimisation peut être formalisé de la manière suivante :

$$\theta^* = \arg\max_{\theta \in \Theta} \mathbb{E}[f(\theta)]$$

où θ représente un vecteur d'hyperparamètres à optimiser, Θ désigne l'espace de recherche défini par l'utilisateur, $f(\theta)$ est la fonction objectif associée à la performance du modèle (par exemple l'accuracy sur un ensemble de validation), et θ^* correspond aux hyperparamètres optimaux recherchés.

Pour résoudre ce problème, une phase initiale d'exploration aléatoire a été conduite, consistant à échantillonner quelques configurations de θ de manière uniformément répartie dans l'espace Θ. À la suite de ces premiers essais, une estimation probabiliste de la distribution des hyperparamètres a été construite. Deux distributions probabilistes distinctes ont été apprises : $l(x)$, représentant la distribution des hyperparamètres associés aux meilleures performances observées, et $g(x)$, représentant la distribution des hyperparamètres associés aux performances moindres.

Le choix des nouveaux hyperparamètres à tester a été ensuite guidé par la maximisation du rapport :

$$\frac{l(\theta)}{g(\theta)}$$

Cette stratégie consiste à privilégier les régions de l'espace des hyperparamètres où la probabilité d'obtenir de bonnes performances est significativement plus élevée que celle d'obtenir de mauvaises performances. Le processus d'optimisation a été mené de manière itérative : à chaque étape, un nouvel ensemble de paramètres θ a été proposé, évalué par la fonction $f(\theta)$, et les distributions $l(x)$ et $g(x)$ ont été mises à jour en conséquence.

Ce procédé a été poursuivi jusqu'à atteindre un nombre maximal d'essais prédéfini ou un temps d'exécution maximal. Par rapport aux méthodes classiques telles que Grid Search ou Random Search, qui explorent l'espace de manière systématique ou totalement aléatoire sans utiliser les résultats précédents, Optuna permet une exploration adaptative de l'espace des hyperparamètres. Il est ainsi capable de concentrer la recherche dans les régions prometteuses, ce qui aboutit à une optimisation plus rapide et plus efficace. L'application de cette approche a permis de sélectionner de manière automatique les hyperparamètres optimaux, assurant ainsi une meilleure performance générale des modèles tout en limitant les coûts computationnels liés à une recherche exhaustive.

Machine Learning

Modèles utilisés

Dans cette étude, plusieurs modèles de machine learning supervisé ont été utilisés afin de prédire la classe d'appartenance des sinistres sur la base de variables explicatives prétraitées. Tous les modèles s'inscrivent dans une problématique de classification multi-classes. Une présentation mathématique synthétique de chacun est proposée ci-dessous.

Le modèle **Random Forest** breiman2001random repose sur la construction d'un ensemble d'arbres de décision entraînés sur des sous-échantillons différents du jeu de données. Chaque arbre est construit par minimisation de l'impureté (souvent mesurée par l'indice de Gini) à chaque nœud, et l'agrégation finale des prédictions se fait par vote majoritaire. Mathématiquement, la fonction prédictive peut être écrite comme

$$\hat{y} = \text{mode}\left(\{h_t(x)\}_{t=1}^{T}\right)$$

où h_t désigne le t-ième arbre de décision et T le nombre total d'arbres.

Le modèle **XGBoost** s'inscrit dans la famille des algorithmes de boosting par gradient. Chaque nouvel arbre est entraîné pour approximer le gradient négatif de la fonction de perte par rapport aux prédictions courantes, conduisant à une amélioration itérative du modèle. Soit $\mathcal{L}(y, \hat{y})$ la fonction de perte, l'objectif à chaque itération est de minimiser

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

où f_t est le nouvel arbre ajouté à l'itération t et Ω est un terme de régularisation pénalisant la complexité du modèle.

Le modèle **HistGradientBoostingClassifier** repose sur une approche similaire au boosting de gradient mais utilise une approximation par histogrammes des valeurs des variables pour accélérer considérablement la recherche des meilleurs seuils de split. À chaque itération, un arbre est entraîné pour approximer la direction du gradient de la fonction de perte, dans un espace discrétisé, en minimisant une perte de type

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \lambda \sum_{j=1}^{J} \text{Penalty}(w_j)$$

où w_j sont les poids des feuilles.

Le modèle **CatBoost** repose également sur un principe de boosting par gradient mais améliore la stabilité du modèle en réduisant l'overfitting grâce à une technique dite d'Ordered Boosting, où l'ordre d'apparition des échantillons est pris en compte dans la construction des arbres. La fonction d'objectif est similaire à celle du boosting classique, avec en particulier une pénalisation plus forte des fluctuations prématurées des gradients, ce qui peut s'écrire comme

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \text{Regul(Ordered Stats)}$$

où la régularisation est calculée en fonction de l'ordre conditionnel des statistiques.

Le modèle **VotingClassifier** repose sur une agrégation des prédictions de plusieurs classificateurs de base. Dans le cas du vote majoritaire ("hard voting"), la prédiction finale est obtenue en prenant la classe la plus prédite par les différents modèles. Mathématiquement, pour un ensemble de modèles $\{h_m\}_{m=1}^{M}$, la prédiction est définie par

$$\hat{y} = \text{argmax}_k \sum_{m=1}^{M} \mathbb{1}_{\{h_m(x)=k\}}$$

où k désigne une classe possible et $\mathbb{1}$ est la fonction indicatrice.

Le modèle **StackingClassifier** repose sur une approche d'apprentissage en deux niveaux. Dans un premier temps, plusieurs modèles de base sont entraînés et leurs prédictions sont utilisées comme nouvelles variables d'entrée pour un modèle final, appelé méta-modèle. Formellement, si $h_m(x)$ désigne la prédiction du modèle de base m, alors le méta-modèle g apprend une fonction

$$\hat{y} = g(h_1(x), h_2(x), \ldots, h_M(x))$$

ce qui permet de capturer des interactions complexes entre les prédictions des modèles de base.

Enfin, le modèle **LightGBM** repose sur un algorithme de boosting de gradient extrêmement optimisé pour les grandes bases de données, utilisant des techniques telles que le "leaf-wise growth" et le "Gradient-based One-Side Sampling" (GOSS). À chaque itération, un arbre est construit en sélectionnant la feuille avec la perte la plus importante pour être divisée, ce qui permet une meilleure convergence. La fonction de perte est similaire à celle du boosting standard

$$\mathcal{L}(y, \hat{y}) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \lambda \sum_{\text{leaves}} \|w_j\|^2$$

avec des techniques d'échantillonnage spécifiques pour réduire le coût computationnel.

Ces modèles ont été choisis pour leur complémentarité en termes de biais, de variance et de capacité d'adaptation à des problèmes complexes de classification multi-classes, notamment dans un contexte de classes initialement déséquilibrées et de distributions très asymétriques des cibles.

Lutte contre l'overffiting

Afin de limiter les risques d'overfitting au cours de la modélisation, plusieurs techniques complémentaires ont été mises en œuvre. Tout d'abord, un suréchantillonnage contrôlé par SMOTE a été appliqué après imputation des valeurs manquantes, permettant de rééquilibrer les classes sans accroître artificiellement le risque de surapprentissage lié à une simple duplication d'observations. Ensuite, dans tous les modèles basés sur des arbres, des stratégies explicites de régularisation ont été introduites, notamment à travers des paramètres tels que la profondeur maximale des arbres (`max_depth`), le poids minimal des feuilles (`min_child_weight` ou `min_samples_leaf`), des pénalités L1 et L2 (`reg_alpha`, `reg_lambda`), ainsi que l'utilisation du sous-échantillonnage des colonnes et des observations (`subsample` et `colsample_bytree`). L'optimisation fine de ces hyperparamètres a été assurée par la méthode bayésienne d'Optuna couplée à une validation croisée stratifiée à cinq plis, permettant une estimation robuste de la performance sur des échantillons hors entraînement. De plus, les algorithmes de boosting utilisés, tels que XGBoost, CatBoost, HistGradientBoostingClassifier et LightGBM, ont naturellement intégré des procédures de pénalisation de la complexité pour prévenir l'adaptation excessive au bruit des données. Enfin, les méthodes d'assemblage comme le VotingClassifier et le StackingClassifier ont permis de stabiliser les prédictions finales en combinant plusieurs modèles de base, réduisant ainsi la variance globale des prédictions sans pour autant accroître significativement le biais.

Comparaison des performances globales

Note introductive : Il est important de noter que des résultats plus performants auraient pu être obtenus si des ressources informatiques plus puissantes, notamment des GPU, avaient été mises à disposition et si un temps d'entraînement plus long avait été alloué sur ordinateurs un peu plus puissants que nos machines personnelles. Dans ce contexte, l'optimisation des modèles de machine learning par Optuna aurait pu être approfondie, permettant ainsi une exploration plus fine de l'espace des hyperparamètres et une convergence vers des solutions de meilleure qualité. En l'absence de telles ressources, les recherches d'optimisation ont nécessairement été limitées, ce qui a pu restreindre le potentiel d'amélioration des performances des modèles évalués.

Afin d'évaluer la performance des différents modèles de classification utilisés dans cette étude, un tableau de comparaison a été établi en regroupant les principales métriques de performance : l'accuracy, la macro-précision, le macro-recall et le macro-f1-score.

Modèle	Accuracy	Macro Precision	Macro Recall	Macro F1-score
XGBoost	0.4862	0.49	0.49	0.46
Random Forest	0.7753	0.78	0.78	0.77
CatBoost	0.8171	0.82	0.82	0.82
HistGradientBoosting	0.8256	0.83	0.83	0.82
Voting Hard	0.8002	0.81	0.80	0.80
Stacking Classifier	0.7682	0.77	0.77	0.76
LightGBM	**0.8363**	**0.84**	**0.84**	**0.84**

Table 1: Comparaison des performances des modèles de classification

L'ensemble des métriques a été calculé à partir des prédictions sur l'échantillon de test après application d'un suréchantillonnage par SMOTE, garantissant ainsi une représentativité équilibrée des classes. Il a été constaté que le modèle LightGBM obtenait les meilleures performances sur l'ensemble des critères considérés, avec une accuracy de 83,63%, une macro-précision de 84%, un macro-recall de 84% et un macro-f1-score de 84%, surpassant ainsi l'ensemble des autres modèles. Le modèle HistGradientBoosting a également présenté d'excellentes performances, atteignant une accuracy de 82,56%, une macro-précision et un macro-recall de 83%, ainsi qu'un macro-f1-score de 82%, ce qui en fait le deuxième modèle le plus performant. Le modèle CatBoost a montré des résultats légèrement inférieurs mais tout de même très solides, avec une accuracy de 81,71% et des scores de 82% pour la macro-précision, le macro-recall et le macro-f1-score. Les modèles Random Forest, VotingClassifier basé sur un vote majoritaire ("hard voting") et StackingClassifier ont affiché des performances intermédiaires, avec des accuracies respectives de 77,53%, 80,02% et 76,82%, reflétant une qualité de prédiction correcte mais inférieure aux approches basées sur le boosting. Enfin, le modèle XGBoost, sans réglage fin avancé des hyperparamètres, a enregistré des résultats nettement plus modestes, avec une accuracy de 48,62% et des scores autour de 49% pour la précision, le recall et le f1-score, suggérant qu'une optimisation plus approfondie aurait été nécessaire pour en améliorer l'efficacité dans ce contexte spécifique. Globalement, il apparaît que les modèles de type boosting, notamment LightGBM et HistGradientBoosting, se sont avérés les mieux adaptés à la tâche de classification multi-classes sur les données étudiées.

Points forts et points faibles des modèles

Les modèles de type boosting, tels que LightGBM, HistGradientBoosting et CatBoost, se sont montrés particulièrement performants. Leur capacité à construire des ensembles d'arbres faibles de manière séquentielle, en corrigeant progressivement les erreurs des itérations précédentes, permet de s'adapter efficacement à des structures complexes, y compris en présence de bruit et de classes déséquilibrées. LightGBM, notamment, bénéficie d'une implémentation optimisée pour gérer de grands volumes de données et d'une stratégie de "leaf-

wise growth" qui favorise une meilleure généralisation lorsque la régularisation est correctement appliquée, ce qui explique ses résultats supérieurs.

HistGradientBoosting, qui repose sur une approximation par histogrammes, offre un excellent compromis entre rapidité d'exécution et robustesse sur des données prétraitées, comme celles issues d'un suréchantillonnage SMOTE. Cela a permis d'atteindre des performances proches de celles de LightGBM, avec une stabilité notable au regard du déséquilibre initial des classes.

Le modèle CatBoost a également démontré de bonnes performances grâce à sa capacité intrinsèque à traiter efficacement les variables catégorielles et à limiter l'overfitting via des techniques telles que l'Ordered Boosting. Toutefois, dans ce projet, le bénéfice attendu de CatBoost sur des variables catégorielles n'a pas été pleinement exploité, les variables d'entrée ayant déjà été principalement numériques ou imputées.

Le Random Forest, bien qu'efficace pour capturer des relations non linéaires et résilient aux données bruitées, montre ici des performances moindres par rapport aux modèles de boosting. Ceci peut s'expliquer par sa stratégie d'agrégation simple par vote majoritaire sans correction d'erreurs progressives, ce qui le rend moins performant en présence de classes initialement déséquilibrées, même après l'application du SMOTE.

Les méthodes d'assemblage telles que le VotingClassifier et le StackingClassifier ont permis d'améliorer légèrement les performances en combinant la diversité des modèles de base. Cependant, le vote majoritaire simple dans le VotingClassifier peut lisser les performances sans capturer pleinement les relations des données. Le StackingClassifier, plus complexe, a offert une meilleure adaptation, mais sa performance est restée limitée par la qualité des modèles de base et par la complexité supplémentaire du problème de classification multi-classes (ici au nombre de 5).

Enfin, XGBoost a obtenu des résultats significativement inférieurs aux autres méthodes. Ce résultat s'explique par un manque d'optimisation fine des hyperparamètres et par une configuration initiale moins adaptée au suréchantillonnage SMOTE.

Analyse des matrices de confusion

Table 2: Matrice de confusion du modèle XGBoost

Classe	Precision	Recall	F1-score
0 (Faible)	0.52	0.29	0.38
1 (Moyen)	0.46	0.30	0.36
2 (Très faible)	0.50	0.96	0.66
3 (Très élevé)	0.46	0.50	0.48
4 (Élevé)	0.49	0.37	0.42
Moyenne (macro)	0.49	0.49	0.46

Le modèle XGBoost a montré de fortes limitations dans cette configuration, avec une accuracy globale de seulement 48,62%. Bien que la classe "Très faible" ait été détectée avec un rappel élevé (96%), les autres classes ont été mal différenciées, avec des scores de précision et de rappel proches de 30 à 50%. Cette performance médiocre s'explique par une forte tendance du modèle à privilégier la classe dominante, entraînant un biais important dans la classification.

Table 3: Matrice de confusion du modèle Random Forest

Classe	Precision	Recall	F1-score
0 (Faible)	0.73	0.45	0.56
1 (Moyen)	0.84	0.80	0.82
2 (Très faible)	0.62	0.88	0.73
3 (Très élevé)	0.87	0.88	0.87
4 (Élevé)	0.86	0.88	0.87
Moyenne (macro)	0.78	0.78	0.77

L'analyse de la matrice de confusion du modèle Random Forest révèle des performances globalement solides, avec une accuracy de 77,53%. Le modèle montre une très bonne capacité à prédire les classes "Très élevé" et "Élevé", affichant des précisions et rappels supérieurs à 85%. Toutefois, des difficultés sont observées pour la classe "Faible", avec un rappel relativement faible de 45%, ce qui indique une tendance du modèle à sous-estimer cette catégorie. Ce comportement peut s'expliquer par la proximité des observations de cette classe avec d'autres classes voisines, rendant la séparation plus difficile. Globalement, Random Forest a su bien capturer la structure sous-jacente des données, notamment grâce à sa capacité à gérer efficacement des variables bruitées et des relations non linéaires, tout en restant légèrement perfectible sur les classes les moins représentées ou plus ambiguës.

Table 4: Matrice de confusion du modèle CatBoost

Classe	Precision	Recall	F1-score
Faible	0.75	0.59	0.66
Moyen	0.88	0.86	0.87
Très faible	0.67	0.81	0.74
Très élevé	0.91	0.91	0.91
Élevé	0.90	0.92	0.91
Moyenne (macro)	0.82	0.82	0.82

L'analyse de la matrice de confusion du modèle CatBoost indique des performances très élevées sur l'ensemble des classes, avec une accuracy de 81,71%. Les classes "Très élevé" et "Élevé" sont particulièrement bien prédites, avec des précisions et rappels dépassant 90%, ce qui témoigne de la capacité du modèle à modéliser efficacement les sinistres de montants élevés. La classe "Moyen" est également très bien capturée, avec un équilibre parfait entre précision et rappel. Une légère difficulté est néanmoins observée pour la classe "Faible", où le rappel s'élève à seulement 59%, traduisant une certaine confusion avec des classes adjacentes. Cette performance globalement excellente peut être attribuée aux mécanismes avancés d'Ordered Boosting de CatBoost, qui réduisent efficacement l'overfitting, même dans un contexte initialement déséquilibré par la distribution des sinistres.

Table 5: Matrice de confusion du modèle HistGradientBoosting

Classe	Precision	Recall	F1-score
Faible	0.76	0.61	0.68
Moyen	0.89	0.86	0.87
Très faible	0.67	0.83	0.74
Très élevé	0.93	0.92	0.92
Élevé	0.91	0.91	0.91
Moyenne (macro)	0.83	0.83	0.82

L'analyse de la matrice de confusion du modèle HistGradientBoosting révèle d'excellentes performances globales, avec une accuracy de 82,56%. Le modèle parvient à prédire très précisément les sinistres de classes "Élevé" et "Très élevé", atteignant respectivement des scores de précision et de rappel supérieurs à 91%. La classe "Moyen" est également très bien modélisée, ce qui traduit une capacité du modèle à généraliser sur des niveaux de sinistres intermédiaires. La classe "Très faible" reste correctement identifiée, bien qu'un léger déficit de précision subsiste par rapport aux classes plus élevées. Enfin, la classe "Faible" affiche une performance modérée, en particulier au niveau du rappel (61%), ce qui pourrait être dû à une certaine confusion avec la classe "Très faible". Ces résultats très solides confirment l'efficacité du boosting sur histogrammes, qui combine vitesse d'exécution et robustesse face aux déséquilibres de classes présents dans le jeu de données.

Table 6: Matrice de confusion du modèle VotingClassifier (vote majoritaire)

Classe	Precision	Recall	F1-score
Faible	0.80	0.51	0.62
Moyen	0.86	0.82	0.84
Très faible	0.65	0.90	0.76
Très élevé	0.88	0.90	0.89
Élevé	0.87	0.87	0.87
Moyenne (macro)	0.81	0.80	0.80

L'analyse de la matrice de confusion du modèle VotingClassifier basé sur un vote majoritaire ("hard voting") met en évidence une bonne robustesse globale, avec une accuracy de 80,02%. Les classes "Élevé", "Très élevé" et "Moyen" sont très bien modélisées, avec des scores de précision et de rappel dépassant largement les 85%. En revanche, la classe "Faible" reste plus difficile à prédire correctement : bien que la précision atteigne un niveau satisfaisant (80%), le rappel reste limité à 51%, ce qui signifie que près de la moitié des cas réels "Faible" ne sont pas correctement identifiés. La classe "Très faible" bénéficie d'un excellent rappel (90%) mais au prix d'une précision plus modérée (65%). Globalement, le VotingClassifier parvient à combiner efficacement la diversité des modèles de base, mais son fonctionnement par majorité simple limite légèrement sa capacité à modéliser des classes plus ambiguës comme "Faible". Ce comportement est typique du vote dur, qui privilégie la stabilité sur la finesse individuelle.

Table 7: Matrice de confusion du modèle StackingClassifier

Classe	Precision	Recall	F1-score
Faible	0.71	0.50	0.59
Moyen	0.81	0.78	0.79
Très faible	0.66	0.86	0.75
Très élevé	0.85	0.86	0.86
Élevé	0.82	0.84	0.83
Moyenne (macro)	0.77	0.77	0.76

La matrice de confusion du StackingClassifier montre une bonne capacité à identifier les classes "Très élevé" et "Élevé" avec des précisions et des rappels proches de 85%. La classe "Très faible" est bien rappelée (86%) mais avec une précision modérée (66%). Les performances sont plus limitées pour la classe "Faible", traduisant des difficultés à distinguer les sinistres de faible montant. Globalement, le modèle bénéficie de la diversité

des apprenants mais reste perfectible pour les classes minoritaires.

Table 8: Matrice de confusion résumée pour LightGBM

Classe	Précision	Rappel	F1-score
Faible	0.78	0.62	0.69
Moyen	0.91	0.88	0.90
Très faible	0.67	0.84	0.74
Très élevé	0.94	0.92	0.93
Élevé	0.92	0.92	0.92
Accuracy		0.8363	
Macro avg	0.84	0.84	0.84
Weighted avg	0.84	0.84	0.84

La matrice de confusion du modèle LightGBM indique d'excellentes performances sur toutes les classes, en particulier pour "Très élevé" et "Élevé" avec des rappels et précisions supérieurs à 90%. La classe "Moyen" est aussi très bien prédite. Les classes "Très faible" et "Faible" sont légèrement moins bien capturées mais restent correctement identifiées. Globalement, LightGBM offre la meilleure séparation entre les classes, confirmant sa supériorité sur ce jeu de données.

Pour chaque modèle il vient donc :

Modèle	Principal apport	Principal défaut
XGBoost	Haute capacité de modélisation complexe	Sensibilité forte aux hyperparamètres et au déséquilibre
Random Forest	Robustesse aux données bruitées et rapide à entraîner	Moins performant sans correction des erreurs (vs boosting)
CatBoost	Excellente gestion des données catégorielles et généralisation solide	Peut être lent sans GPU et nécessite des ajustements fins
HistGradientBoosting	Très bon compromis vitesse/précision pour grands jeux	Moins performant sans prétraitement soigné
VotingClassifier	Stabilité par agrégation des modèles diversifiés	Dilution possible des performances individuelles
StackingClassifier	Capture des interactions complexes entre modèles	Complexité accrue et besoin d'une bonne diversité de base
LightGBM	Performances élevées avec un entraînement rapide	Sensibilité accrue aux surajustements sur petits échantillons

Table 9: Résumé des principaux avantages et inconvénients des modèles utilisés

Application à l'assurance automobile

L'ensemble des méthodes de classification développées a été appliqué au contexte spécifique de l'assurance automobile, avec pour objectif de prédire la classe de montant d'un sinistre à partir de caractéristiques relatives au véhicule, au conducteur et à la police d'assurance. La transformation initiale du problème de régression asymétrique en une tâche de classification multi-classes a permis d'améliorer la stabilité et l'interprétabilité des modèles. Grâce au prétraitement minutieux des données, au rééquilibrage des classes par SMOTE et à l'optimisation fine des hyperparamètres, des performances de prédiction élevées ont été atteintes, en particulier avec les modèles de type boosting comme LightGBM et HistGradientBoostingClassifier. La capacité à prédire correctement la classe de gravité d'un sinistre offre des perspectives opérationnelles importantes en assurance, notamment en matière de tarification plus précise, de gestion proactive des risques et d'allocation efficiente des réserves. Cette approche souligne l'intérêt de l'apprentissage automatique pour traiter des problématiques complexes où les montants sont soumis à de fortes hétérogénéités et asymétries, caractéristiques typiques du secteur assurantiel.

Conclusion

Le projet a permis de démontrer la pertinence de l'utilisation des méthodes d'apprentissage automatique pour la prédiction de classes de montants de sinistres dans le secteur de l'assurance automobile. En transformant une problématique initiale de régression sur une variable fortement asymétrique en une tâche de classification multi-classes équilibrée grâce à l'application de la technique SMOTE, il a été possible d'améliorer considérablement la capacité des modèles à généraliser. Les phases successives de nettoyage des données, d'ingénierie des variables, d'imputation, d'optimisation bayésienne des hyperparamètres et de comparaison rigoureuse des performances ont permis de sélectionner des modèles particulièrement adaptés au problème posé. LightGBM s'est révélé être le modèle le plus performant, atteignant une accuracy de plus de 83%, suivi de près par HistGradientBoosting et CatBoost. Ces résultats confirment la supériorité des méthodes de boosting adaptées à des jeux de données déséquilibrés et complexes. La démarche adoptée, fondée sur une méthodologie complète de traitement de données et d'optimisation des modèles, a permis d'obtenir des prédictions fiables et robustes, tout en mettant en évidence l'importance du prétraitement dans la réussite des modèles d'apprentissage supervisé.

Discussion

Malgré les performances satisfaisantes obtenues, certaines limites doivent être soulignées. Le choix de transformer le problème en classification a permis de contourner les difficultés liées à l'asymétrie de la variable continue mais a conduit à une perte d'information fine sur les montants exacts des sinistres. De plus, bien que SMOTE ait permis de rééquilibrer les classes, il a potentiellement introduit des observations synthétiques qui ne reflètent pas exactement la distribution naturelle des données. Par ailleurs, la configuration initiale des modèles, notamment pour XGBoost, aurait pu être affinée davantage afin d'obtenir de meilleures performances, suggérant que le tuning d'hyperparamètres est particulièrement crucial pour les modèles sensibles au surapprentissage. La mise en œuvre du stacking et du voting a montré un potentiel intéressant mais n'a pas surpassé les meilleurs modèles de boosting, ce qui met en évidence la difficulté d'agréger efficacement des modèles lorsque ceux-ci ne présentent pas une complémentarité parfaite. Enfin, l'absence d'une analyse plus fine de l'importance des variables, par exemple via des méthodes de type SHAP ou permutation feature importance, constitue une piste d'amélioration pour des travaux futurs, permettant une meilleure interprétabilité des résultats obtenus. Ces éléments suggèrent que si l'approche globale adoptée est

adaptée et performante, des extensions vers des modèles explicatifs plus sophistiqués ou des traitements spécifiques aux données déséquilibrées pourraient encore renforcer la qualité de la modélisation.

References

1. Breiman L. Random forests. Machine learning 2001; 45:5–32
2. Chen T and Guestrin C. XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM. 2016 :785–94
3. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, and Liu TY. LightGBM: A highly efficient gradient boosting decision tree. 2017; 30
4. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, and Gulin A. CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems 2018; 31
5. Friedman JH. Greedy function approximation: A gradient boosting machine. Annals of statistics 2001 :1189–232
6. Chawla NV, Bowyer KW, Hall LO, and Kegelmeyer WP. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*. Vol. 16. 2002 :321–57