AssurPrime : Saurez-vous prédire la prime d'assurance ? par Crédit Agricole Assurances

Charles Brunet

March 18, 2025

Abstract

Ce document a pour but de présenter la démarche scientifique que j'ai suivi durant ce challenge en présentant les méthodes, les fondations théoriques et les algortihmes que j'ai utilisés.

1 Introduction

Crédit Agricole Assurances est une filiale du Groupe Crédit Agricole dédiée à l'assurance, faisant de celui-ci un acteur multi-expert de la bancassurance et le 1er bancassureur en Europe. Crédit Agricole Assurances regroupe plusieurs entités, dont Predica et Pacifica, qui proposent une large gamme d'assurances aux particuliers, aux exploitants agricoles, aux professionnels et aux entreprises. Pour ce challenge, nous nous intéressons au produit Multirisque agricole. Ce produit d'assurance est souscrit par un agriculteur afin de sécuriser son exploitation. Elle couvre aussi bien l'exercice de son activité professionnelle, les dommages subis par ses bâtiments d'exploitations, les matériels stockés ainsi que sa protection financière et juridique. L'assuré est ainsi couvert efficacement et dans la durée pour qu'en cas de coup dur, la continuité de son activité soit assurée d'un point de vue matériel comme financier. La multirisque agricole a généré près d'1 milliard d'euros de chiffre d'affaires en 2023 soit 36% du marché agricole en France.
Notre tâche consiste à modéliser le risque incendie à partir des données fournies. C'est un enjeu majeur de ce produit d'assurance étant donné qu'il représente une part très importante de la charge globale sinistre, tout assureur confondu (environ 35% en 2023, source FFA).

Notre but est de modéliser deux données cibles liées au risque incendie, ce qui permet d'obtenir la charge annuelle pour un contrat :

. Fréquence = (Nombre de sinistres sur une année civile)/(Nombre d'années d'assurances)

. Coût moyen = (Somme des charges sur une année civile)/(Nombre de sinistres sur une année civile)

Enfin, ces deux mesures nous permettent de déterminer la charge annuelle :

Charge annuelle = (Fréquence) * (Coût moyen) * (Nombre d'années d'assurance)

2 Méthodologie & Sommaire

2.1 Méthode et organisation du travail

Pour la réalisation de ce challenge, j'ai organisé mon travail en me basant sur la présentation de Pierre Courtiol, ayant eu lieu au Collège de France le 21 février 2021. Data scientist, il a participé de nombreuses compétitions de Machine Learning déployées par des entreprises ou des laboratoires de recherches sur la plateforme Kaggle. Parmi les seize compétitions auxquelles il a participé sur Kaggle, il a obtenu la médaille d'or pour sept d'entre elles, soit l'équivalent du top 10.
Son expertise et ses conseils m'ont semblé très judicieux et adéquat au challenge AssurPrime auquel je participe.

Selon lui, la méthodologie à adopter pour garantir une étude complète et performante du problème

est la suivante (en considérant que les pourcentages correspondent au temps investi sur l'ensemble de notre travail) :

- Passage en revue de l'état de l'art, comparer notre problème aux challenges passés et analyser leurs résultats s'ils ont été postés (5%)

- Exploration des données, se familiariser avec le data set fourni, les colonnes, leur composition, les données manquantes, etc. (20%)

- Features engineering, c'est-à-dire transformer la donnée pour être exploitée au mieux par les algorithmes, extraire les informations pertinentes et cachées dans les données, sélectionner de variables explicatives (50%)

- Mise en place d'une procédure d'évaluation (10%)

- Création du modèle (15%)

J'ai donc essayé de suivre ce procédé en l'adaptant aux problématiques du challenge AssurPrime.

Avant de me plonger dans l'étude du dataset, j'ai donc recherché parmi les précédents challenges ceux qui présentaient le plus de similarités avec celui d'AssurPrime.
Le plus pertinent que j'ai trouvé jusqu'ici est celui proposé par Generali en 2019 intitulé Prédiction de sinistres. Le problème que posait ce challenge était de savoir si un immeuble allait avoir au moins un sinistre pendant l'année. Florian Pothin et Morgan Riou, gagnants du challenge, ont présenté leur démarche au Collège de France. Bien que leurs résultats ni leur rapport ne soient accessibles, leur démarche m'a semblé intéressante avant de me lancer dans mon travail.
En effet, les similarités que présentent les deux problèmes sont :

- les données tabulaires

- beaucoup de valeurs manquantes pour certaines colonnes

- une forte corrélation entre la zone géographique et le sinistre (zones sèches et chaudes auront un risque beaucoup plus élevé d'incendies, présence de pompiers, etc.)

- une forte corrélation entre l'exploitation agricole (dans leur cas le bien immobilier) et le sinistre (vétusté, équipements, etc.)

2.2 Sommaire

De part la méthodologie que j'ai choisi d'adopter, ce rapport s'organise autour de trois parties :

1. Exploration des données
2. Feature engineering
3. Modélisation

Je tâcherai de rendre ce rapport le plus détaillé et complet possible au regard des techniques que j'ai utilisées et de leurs fondations théoriques. Je tâcherai également d'éviter les redondances.

3 Exploration des données

3.1 Présentation des données

Les données fournies par Le Crédit agricole comportent un jeu d'entrainement décomposé entre variables explicatives et variables cibles, respectivement X_train et y_train. La table des variables explicatives d'entraînement comporte 383610 lignes et 374 colonnes. La table de variables cibles contient autant de lignes mais seulement 5 colonnes. Pour le test, nous avons seulement accès à la table des variables explicatives qui contient 95582 lignes et 374 colonnes.

En somme notre jeu de données a été divisé en deux parties, 80% environ constitue l'entraînement et

20% environ constitue le test. Parmi les 374 variables explicatives, on en distingue deux types :

- 140 variables internes (propres à l'exploitation) : données de capitaux, activités, surfaces, bâtiment, sinistralité passée, présence d'un extincteur, données de surfaces, localisation, surface, etc.

- 228 variables externes (propre à la zone géographique) : données de météo France (moyenne mensuelle, donnée du vent, pluviométrie), distance par rapport à la caserne de pompier la plus proche, code INSEE, etc.

A noter qu'une ligne de la base ne correspond pas à un unique contrat mais à l'ensemble des assurés ayant les mêmes caractéristiques.

3.2 Etude des données

3.2.1 Variables explicatives

Nous disposons d'une table qui contient 383610 lignes et 374 colonnes. L'écrasante majorité de nos variables sont qualitatives, voire toutes. Une variable qualitative ne peut prendre qu'un nombre fini de valeurs. J'ai donc transformé l'ensemble des variables qualitatives (qui pouvaient être des chaînes de caractères) en variables catégoriques, soit en nombre. J'ai choisi d'être beaucoup plus exhaustif sur le traitement des colonnes dans la section *Feature engineering*.

3.2.2 Variables cibles

Parmi les variables cibles, *Fréquence* et *Coût moyen*, j'ai remarqué que l'on avait un problème d'asymétrie important au sein de nos données. En effet, on a très peu de samples pour lesquelles on a constaté un sinistre, 2358 sur un total de 383610 samples soit près de 0.6%. Tous les autres samples ont pour valeur 0. Ce déséquilibre dans nos données pourrait empêcher que notre modèle apprenne des cas sinistrés.

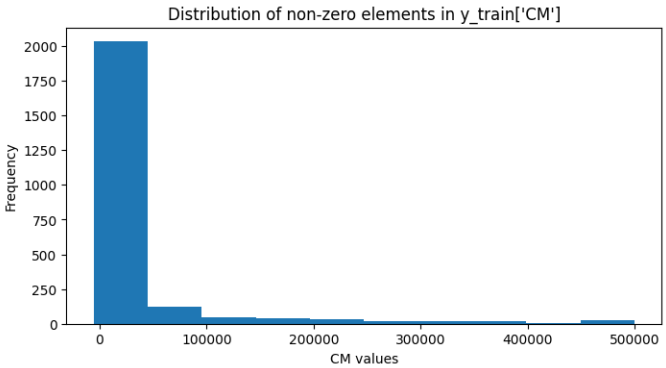


Figure 1: Histogramme présentant la distribution des valeurs sinistrées de la variable *Coût moyen* de y_{train}

4 Feature engineering

4.1 Stratégie

Parmi toutes les variables auxquelles nous avons accès, j'ai opté dans un premier temps pour une stratégie de petite échelle : j'ai sélectionné quelques variables aux noms explicites (ZONE, ZONE_VENT, etc.) que j'ai traité pour pouvoir les implémenter dans nos modèles de régressions.

Une fois cette tâche réalisée, j'ai employé les méthodes que j'avais réalisées à grande échelle, sur le reste des variables.

Compte tenu du nombres de colonnes, je fais le choix de détailler seulement quelques exemples significatifs. Les méthodes que j'ai utilisé sont très souvent analogues : attribuer des valeurs numériques aux variables qualitatives, traiter les valeurs manquantes via diverses stratégies (détaillées dans la suite

du rapport), détecter les variables qui avaient plusieurs types pour qu'elles n'en aient qu'un, analyser la répartition de chacune des catégories d'une variable au sein de la table d'entraînement grâce à des histogrammes, etc.

Chacun des procédés employés repose sur un fondement mathématique, statistique ou bien un raisonnement logique, qui se veut le plus efficient possible en termes de temps d'exécution et de mémoire.

4.2 Exemple type

La colonne NB_CASERNES est un exemple typique des modifications que j'ai pu apporter. Celle-ci contenait des données qualitatives sous quatre formes :

Variable	Type Python avant transformation	Variable après transformation	Nouveau type en Python
01. ≤ 1	Str	1.0	Float
02. ≤ 3	Str	2.0	Float
03. ≤ 14	Str	3.0	Float
04. ≥ 14	Str	4.0	Float

Table 1: Transformation des variables de la colonne NB_CASERNES

Cette variable présentait environ 4,84% de valeurs manquantes soit 18566 lignes environ. Le nombre de casernes autour d'une exploitation agricole est une donnée clef dans le contexte du risque incendie. Un nombre important de casernes augmente la probabilité que l'intervention des pompiers soit rapide et donc de minimiser les dégâts du sinistre. Pour remplacer ces valeurs manquantes, j'ai donc défini une fonction qui agit selon le fonctionnement suivant, en m'appuyant sur la colonne ZONE :

- Etape 1 : Si la donnée n'est pas manquante, on retourne la donnée

- Etape 2 : Si la donnée est manquante et que la donnée contenue dans la colonne ZONE l'est aussi, on retourne une valeur manquante

- Etape 3 : Si la valeur est manquante mais que ZONE ne l'est pas, on crée une sous table de la table d'entraînement contenant uniquement les lignes qui ont la même ZONE. Une fois cette table créée, on retourne la classe ayant le plus d'occurrence dans cette ZONE. Ici, j'ai choisi de ne pas moyenner compte tenu du fait que nos valeurs désignent des variables qualitatives.

Grâce à l'utilisation d'un dictionnaire, j'ai pu accélérer l'étape 3 et améliorer drastiquement les performances de la fonction tout en réduisant le coût en mémoire vive. En effet, si nous sommes dans la troisième situation et que la donnée a déjà été calculée et stockée dans notre dictionnaire, nous pouvons directement retourner la valeur plutôt que d'effectuer le même calcul, assez coûteux. Cette fonction nous a permis de remplacer plus de 97% des valeurs manquantes (4 valeurs manquantes après avoir appelé la fonction).

4.3 Traiter les valeurs manquantes

De nombreuses colonnes au sein du dataset comportent de nombreuses valeurs manquantes. Les données manquantes pénalisent nos modèles. De ce fait, j'ai donc essayé de les substituer en employant différentes stratégies que je présente dans cette section. Au sein de mon dataset, j'ai séparé les colonnes en plusieurs catégories :

- les colonnes qui ne contenaient qu'une seule valeur que j'ai supprimée car elles ne fournissent aucune information au modèle.

- les colonnes éligibles au test du χ^2 (voir les hypothèses dans la section sur le test du χ^2)

- les colonnes non-éligibles au test du χ^2

Parmi les colonnes les colonnes non-éligibles au test du χ^2, je les ai divisées en deux catégories, celles qui ne possédaient pas de valeurs manquantes et celles qui en présentaient.

4.3.1 Test du χ^2

Le test d'indépendance du χ^2 est couramment utilisé en machine learning pour analyser les relations entre variables catégorielles, notamment dans la gestion des valeurs manquantes. Il permet de vérifier si deux variables sont statistiquement indépendantes en comparant les fréquences observées et attendues. Dans un tableau de contingence $I \times J$, chaque cellule (i, j) contient une fréquence observée O_{ij}. Le test repose sur les hypothèses suivantes :

- **Hypothèse nulle** H_0 : les variables sont indépendantes.
- **Hypothèse alternative** H_1 : il existe une relation entre elles.

Les fréquences attendues sont calculées à partir des distributions marginales des variables. La statistique χ^2 est obtenue en comparant les écarts entre valeurs observées et attendues. Plus cet écart est important, plus la dépendance entre les variables est probable. Pour que le test soit valide, il est nécessaire que chaque catégorie ait au moins 5 occurrences. De plus, les données doivent être indépendantes et exprimées sous forme de fréquences. En pratique, si la valeur de χ^2 dépasse un seuil critique, on rejette H_0, indiquant une relation significative entre les variables[Riv22].

Mon utilisation du test : pour chacune de mes variables éligibles à ce test, j'ai effectué le test en itérant sur les colonnes du data set. Si la p-valeur du test est inférieur à 0.05 (seuil standard), alors je considère que l'hypothèse H_0 est rejetée au niveau 5%. Une fois la dépendance établie entre deux variables, j'applique un procédé similaire à celui présenté dans la section pour les variables NB_CASERNES et ZONE.

```
P-value: 0.09626544588208787
--> The variables IND_Y7_Y8 and ANCIENNETE are independent (no significant association).

 Test the independance between IND_Y7_Y8 and DUREE_REQANEUF using CHI2 test :


 Our table has the size : 365020

P-value: 1.3142746397094437e-11
--> The variables IND_Y7_Y8 and DUREE_REQANEUF are dependent (significant association).
The column to modify IND_Y7_Y8 is dependant to DUREE_REQANEUF.
```

Figure 2: Calcul de la p-valeur entre différentes colonnes pour le test d'indépendance du χ^2

4.3.2 Multiple imputation

L'imputation multiple est une technique utilisée en machine learning pour gérer les valeurs manquantes en générant plusieurs jeux de données imputés, plutôt qu'une seule estimation. Cette approche permet de mieux représenter l'incertitude liée aux valeurs manquantes et d'améliorer la robustesse des modèles prédictifs. Le processus repose sur les étapes suivantes :

- Remplir les valeurs manquantes plusieurs fois en générant m jeux de données imputés en fonction des distributions observées.
- Analyser chaque jeu de données séparément avec les modèles choisis.
- Combiner les résultats obtenus pour obtenir une estimation plus fiable des paramètres et des intervalles de confiance.

L'imputation multiple repose sur l'hypothèse que les données manquantes sont MAR (Missing At Random), c'est-à-dire que la probabilité d'absence d'une valeur dépend d'autres variables observées mais pas de la valeur elle-même.

Mon utilisation du test : j'ai donc fait le choix d'utiliser la fonction *IterativeImputer* proposée par *scikit-learn* avec la stratégie *'most_frequent'*, similairement à ce que j'ai déjà présenté via l'exemple de la section 4.2, étape 3. J'ai appliqué cette procédure uniquement aux variables qui n'étaient pas éligibles au test du χ^2 et qui présentaient des valeurs manquantes [imp24][imp25a][imp25b].

4.3.3 Filtrer les colonnes

Après avoir effectué ces deux procédures, j'ai établi un dernier filtre pour déterminer les colonnes qui avaient plus de 75%, 50% et 25% de valeurs manquantes. A partir de cette analyse, j'ai créé des tables alternatives à notre table d'entraînement pour pouvoir tester leurs performances une fois implémentées dans notre modèle. La table que j'ai principalement utilisée est celle auquelle j'ai retiré les colonnes présentant plus de 50% de valeurs manquantes.

5 Modélisation

Etant donné que j'ai réalisé ce travail sans binôme, dans un interval de temps assez limité, j'ai concenctré mon travail sur un modèle interprétable et largement documenté. En ce sens, j'ai choisi d'implémenter principalement des modèles de forêts aléatoires que je présenterais dans cette section.

A noter que j'ai implémenté deux modèles, un pour prédire la *Fréquence* et l'autre pour prédire le *Coût Moyen*. Les procédures que j'ai suivies étant très similaires, je ne ferais pas de distinction entre les modèles dans mes explications.

5.1 Le modèle

On rappelle que nos variables cibles sont :

. Fréquence = (Nombre de sinistres sur une année civile)/(Nombre d'années d'assurances)

. Coût moyen = (Somme des charges sur une année civile)/(Nombre de sinistres sur une année civile)

Enfin, ces deux mesures nous permettent de déterminer la charge annuelle :

Charge annuelle = (Fréquence) * (Coût moyen) * (Nombre d'années d'assurance)

De ce fait, modéliser les données de Fréquence et Coût moyen rentrent dans le cadre des régressions supervisées.

En apprentissage supervisé, la régression est une tâche qui consiste à prédire une variable continue Y à partir d'un ensemble de variables d'entrée X. Soit X une matrice de variables d'entrée de dimension $n \times p$, et Y un vecteur de dimension $n \times 1$ contenant les valeurs cibles. L'objectif est de trouver une fonction f telle que :

$$Y = f(X) + \epsilon$$

où ϵ représente le terme d'erreur. L'objectif est de minimiser une fonction de coût, souvent la racine de l'erreur quadratique moyenne (RMSE) :

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

5.2 Forêts Aléatoires

Une **Forêts Aléatoires ou Random Forest** est un ensemble d'arbres de décision utilisé pour la régression et la classification. Elle repose sur le principe du **bagging (Bootstrap Aggregating)**, une méthode d'ensemble qui consiste à entraîner plusieurs modèles indépendants sur des échantillons aléatoires avec remise des données d'entraînement. Ensuite, les prédictions individuelles sont agrégées pour améliorer la performance globale du modèle.

Dans une Random Forest, plusieurs arbres de décision sont construits en parallèle. Chaque arbre est entraîné sur un sous-ensemble de données obtenu par tirage aléatoire avec remise (bootstrap). À chaque nœud, un sous-ensemble aléatoire des variables explicatives est sélectionné pour déterminer la meilleure séparation.

En régression, la prédiction finale est obtenue en moyennant les prédictions de tous les arbres, ce qui réduit la variance et améliore la robustesse du modèle. La Random Forest gère bien la non-linéarité

et les interactions entre variables, est peu sensible aux valeurs aberrantes et limite le surapprentissage grâce à l'agrégation des prédictions. Toutefois, elle peut être coûteuse en calcul et perdre en interprétabilité par rapport à un simple arbre de décision.

5.3 Optimisation des Forêts Aléatoires

5.3.1 Hyperparamètres : théorie

Dans notre implémentation, nous avons utilisé le modèle `RandomForestRegressor` de la librairie `cuML`, avec les hyperparamètres suivants :

- `n_estimators` (N) : Nombre d'arbres dans la forêt. Un nombre élevé d'arbres améliore généralement la robustesse et réduit la variance, mais augmente le temps de calcul. Un choix trop faible peut mener à un sous-apprentissage (*underfitting*), tandis qu'un nombre trop grand peut ralentir l'entraînement sans gain significatif en précision.
- `min_samples_leaf` (L) : Nombre minimal d'échantillons requis dans une feuille terminale. Une valeur élevée contraint la croissance des arbres, rendant le modèle plus simple et limitant l'overfitting, mais peut aussi réduire sa capacité à capturer des tendances fines (*underfitting*).
- `max_features` (F) : Nombre maximal de variables considérées pour chaque séparation de nœud. Une valeur faible réduit la corrélation entre les arbres et améliore la généralisation, limitant l'overfitting. En revanche, une valeur trop petite peut conduire à un sous-apprentissage si des variables importantes ne sont jamais sélectionnées.

L'optimisation de ces hyperparamètres permet d'atteindre un équilibre entre biais et variance, en réduisant le risque de surapprentissage tout en conservant une bonne capacité prédictive.

5.3.2 Hyperparamètres : pratique

Avant d'implémenter le modèle, j'ai étudié la documentation du livre *The Elements of Statistical Learning* [TH17] pour comprendre son fonctionnement précis. Il est notamment recommandé pour la régression de choisir `max_features` $= \lfloor p/3 \rfloor$ (où p est le nombre de colonnes dans notre table) et `min_samples_leaf` $= 5$.

Par la suite, j'ai cherché à sélectionner une valeur optimale pour le paramètre $n_{estimators}$, qui correspond au nombre d'arbres dans la forêt.

On rappelle ce qu'est la *validation croisée ou cross validation* et l'*out of bag samples (OOB)*.

La **validation croisée** est une technique d'évaluation de modèle qui consiste à diviser les données en plusieurs sous-ensembles afin d'entraîner et tester le modèle sur différentes partitions des données. Une des méthodes courantes est la validation croisée k-fold, où les données sont divisées en k sous-ensembles de taille égale. Le modèle est entraîné sur $k-1$ sous-ensembles et évalué sur le sous-ensemble restant, ce processus étant répété k fois avec une permutation des ensembles d'entraînement et de test.

Une autre technique d'évaluation des forêts aléatoires est l'utilisation des **out of bag samples (OOB)**. Lors de l'entraînement d'une forêt aléatoire, chaque arbre est construit à partir d'un échantillon bootstrap des données d'entraînement. En conséquence, certains échantillons ne sont pas utilisés dans la construction d'un arbre donné. Ces out of bag samples peuvent alors être utilisés pour estimer la performance du modèle sans avoir recours à une validation croisée explicite.

Dans un premier temps, j'ai cherché à optimiser le paramètre $n_{estimators}$ en utilisant l'erreur des out of bag samples avec le modèle `RandomForestRegressor` de la bibliothèque scikit-learn. Toutefois, cette méthode s'est révélée coûteuse en temps de calcul.

Afin d'améliorer l'efficacité, j'ai utilisé l'implémentation GPU de `RandomForestRegressor` de la bibliothèque cuML. Cependant, cette version ne prend pas en charge l'estimation de l'erreur out of bag samples. Par conséquent, j'ai opté pour la méthode de validation croisée classique avec k-fold ($k = 5$), en évaluant l'erreur quadratique moyenne (RMSE) pour différentes valeurs de $n_{estimators}$: 50, 100, 200, 300, 400 et 500.

Cette approche m'a permis de sélectionner la valeur optimale de $n_{estimators}$ en tenant compte des contraintes de temps de calcul, tout en garantissant une évaluation robuste de la performance du modèle.

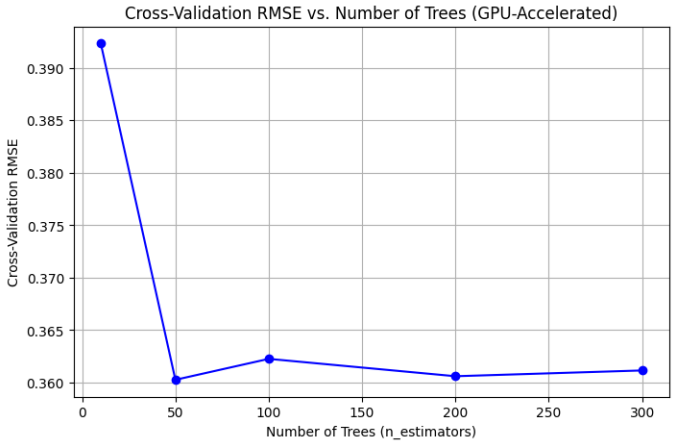


Figure 3: Sélection du meilleur $n_{estimators}$ pour le modèle de *Fréquence*

5.4 Autres contributions

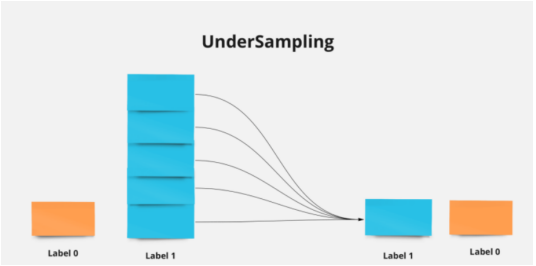
Comme je l'ai mentionné dans la section 3.2.2, nos variables cibles sont très déséquilibrées car elle présente beaucoup moins de sample pour lesquelles on a constaté de sinistres (si *Fréquence* et *Coût moyen* valent 0, alors il n'y a pas eu de sinistre. Parmi les variables sinistrées, celle-ci présentent de grandes disparités, notamment pour la variable *Coût moyen*.

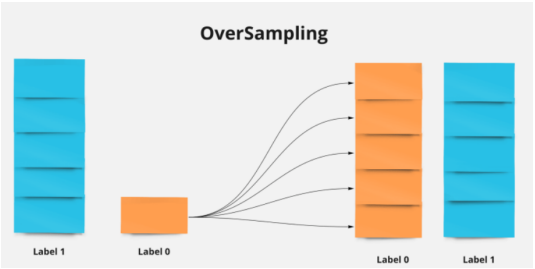
Nous pouvons constater sur notre histogramme de la section 3.2.2 (figure 1) que les données sont très disparates puisqu'elles peuvent prendre des valeurs négatives jusqu'à atteindre 500000. J'ai donc choisi d'assigner une valeur de -5000 à toutes les valeurs inférieurs à ce seuil et 50000 à toutes les valeurs au délà de ce seuil. On aurait aussi pu faire le choix de retirer les outliers ou de choisir un seuil plus large. On rappelle qu'un outlier peut être vu comme une donnée très éloignée de l'ensemble des données de notre table. L'outlier peut biaiser nos modèles de prédiction car ils vont apprendre de cette situation extrême.

Pour réduire cette disparité, j'ai également fait le choix d'appliquer une transformation logarithmique du type $x = log(x + 1)$ pour prévenir les erreurs en 0.

Le problème de cette transformation est qu'il m'oblige à exclure les cas pour lesquels la valeur du *Coût moyen* est négative. Pourtant, ce modèle a montré de bonnes performances. Une solution alternative pourrait de standardiser mes données d'entraînement. Il serait aussi intéressant d'élargir le domaine d'exclusion des outliers à $[-100000, 100000]$.

J'ai également essayé une méthode dite de *sampling*, plus utilisée pour les problèmes de classification, pour rééquilibrer mon dataset. Dans un modèle où l'on dispose d'une classe majoritaire et d'une classe minoritaire, on tâche de rétablir la balance soit en diminuant la proportion de données appartenant à la classe majoritaire, l'*undersampling*, soit en augmentant la proportion de données appartenant à la classe minoritaire, l'*oversampling*. J'ai implémenté les deux méthodes. Dans les deux cas, les prédictions de mon modèle ont **largement sous-performé**.





6 Conclusion

Ma participation a ce challenge m'a permis de découvrir de nouvelles techniques liées au machine learning extrêmement intéressantes. J'ai pu apprendre à travailler avec une base de données très importante, restituer et gérer les valeurs manquantes, optimiser une forêt aléatoire, traîter des données déséquilibrées.
Comme je l'ai mentionné dans la section 5, il me reste de nombreuses pistes à exploiter pour améliorer les prédictions de mes deux modèles, notamment la standardisation de ma variable cible ou encore ajuster la fenêtre d'exclusion des outliers.
Par ailleurs, nous pourrions nous intéresser aux techniques développées pour traiter les données déséquilibrées dans le cadre de régression. Il existe une documentation à ce sujet que je n'ai pas encore eu l'occasion d'implémenter [Yan21]. Enfin, nous pourrions considérer l'étude d'autres modèles que celui des forêts aléatoires comme les réseaux de neurones ou les méthodes de boosting. Néanmoins, le nombre de paramètres des modèles tels que les réseaux de neurones couplée à la taille de nos données entraîneraient une complexité importante, notamment pour un réseau profond. On pourrait alors considérer des techniques pour réduire la dimension de nos données comme le *Principal Component Analysis*.

References

[imp24] Impute missing data values (multiple imputation). *IBM*, 2024.

[imp25a] Iterativeimputer. *scikit-learn*, 2025.

[imp25b] Simpleimputer. *scikit-learn*, 2025.

[Riv22] Vincent Rivoirard. Statistique mathematique. 2022.

[TH17] Jerome Friedman Trevor Hastie, Robert Tibshirani. The elements of statistical learning. 2017.

[Yan21] Yuzhe Yang. Strategies and tactics for regression on imbalanced data. *Towards Data Science*, 2021.