Prédiction de la Réponse des Clients à une Offre d'Assurance Automobile

Samy Mekkaoui, Cyril Zaïmi, Keryann Massin, José Richa

ENSAE Paris

19 mars 2024

Table: Illustration de notre dataset

Gender	Age	Previously Insured	Annual Premium	..	Response
Male	44	1	40454.0	..	0
Male	76	1	33536.0	..	1
...	..	..	..	..	..
Female	47	0	38294.0	..	1

- Les différentes variables de notre dataset autre que **Response** vont nous servir de variables explicatives.
- La variable **Response** constitue notre target que nous noterons dorénavant Y indiquant si le client est intéressé à la souscription d'une assurance pour véhicule.

On a donc ici à faire avec un problème de **classification**

Exploratory Data Analysis

Quelques Faits Stylisés

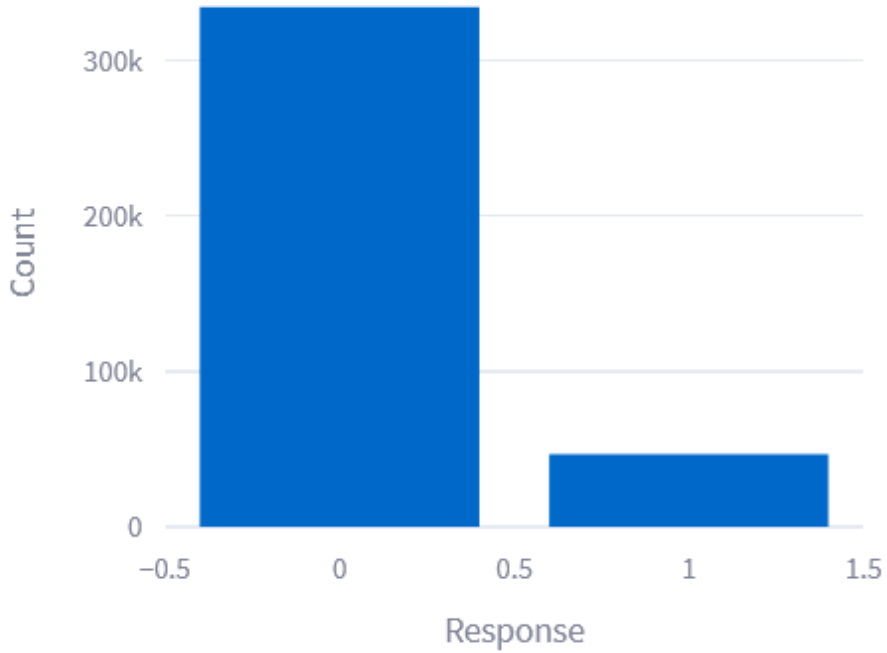

Figure: Graphique représentant la répartition de la variable Y

- On observe que notre dataset apparaît comme **déséquilibré** avec une sur représentation de la variable 0 par rapport à 1

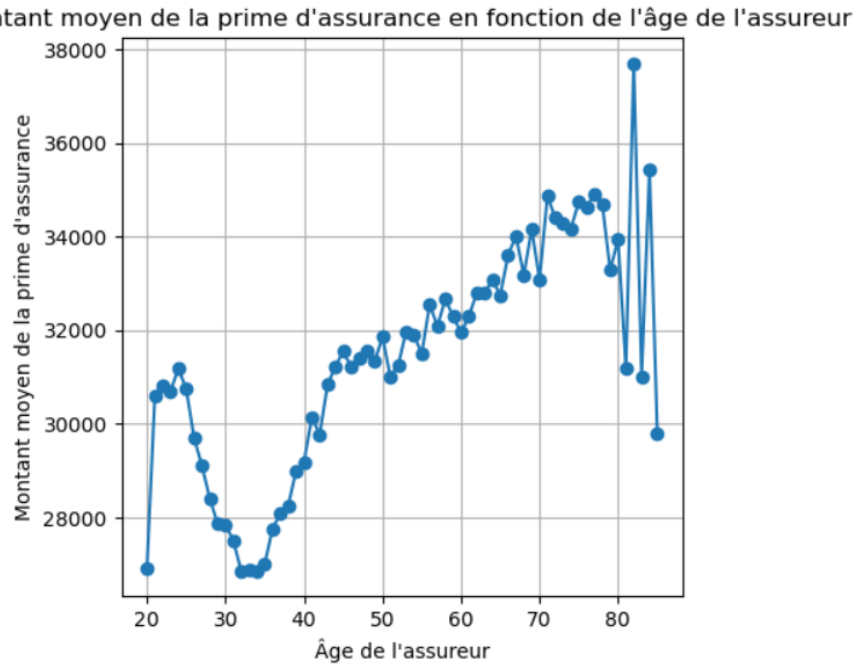

Figure: Evolution du montant moyen de la prime en fonction de l'âge de l'assuré

On observe grâce à ce graphique 3 faits stylisés :

- **La bosse des accidents** pour les assurés d'âge autour de 25 ans :
- une prime $+/-$ **linéaire** pour les assurés d'âge entre 45 et 70 ans:
- **Un manque de données** pour les assurés d'âge $>$ 70 ans

Exploratory Data Analysis

Quelques Faits Stylisés

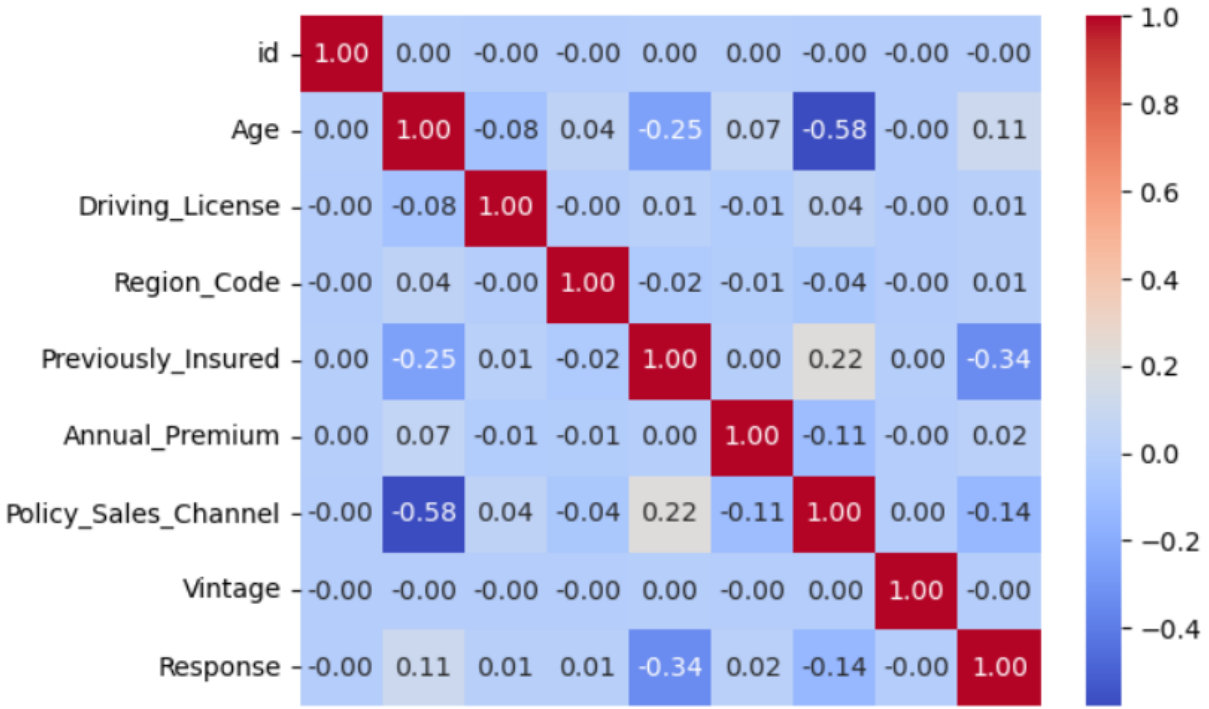

Figure: Matrice de Corrélation de notre dataset

- La variable **Previously Insured** semble être assez décorrélée de notre variable Y ($\text{Corr}(\textit{PreviouslyInsured}, Y) = -0.34$)
- Les variable **Age** et **Policy Sales Channel** semblent également avoir un pouvoir prédicteur avec Y.

Feature Engineering

Un peu de modification du dataset

- On modife les variables catégorielles **Gender** , **Vehicle Damage** et **Vehicle Age** en numériques.
- On observe de nombreux outliers dans la variable **Annual Premium** que l'on va gérer manuellement
- On gère le problème d'over/undersampling en utilisant la méthode de Python **RandomUnderSampler**

Feature Engineering

Un peu de modification du dataset

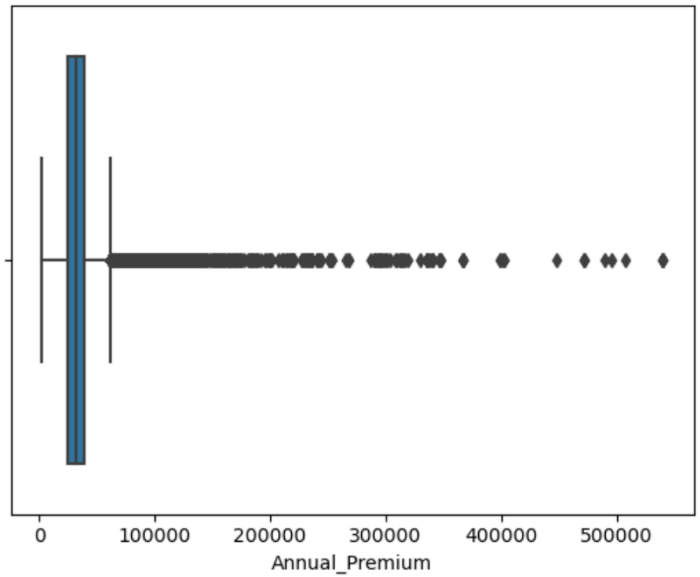

Figure: Boxplot de la variable **Annual Premium** avec les outliers

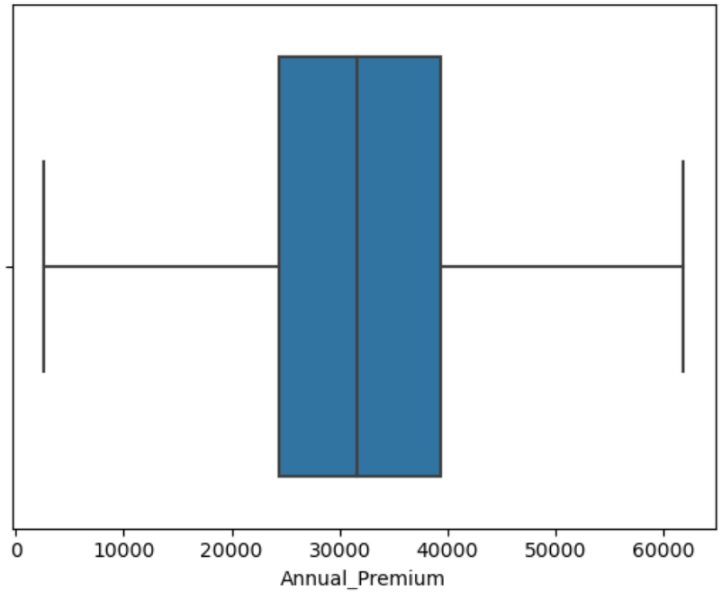

Figure: Boxplot de la variable **Annual Premium** sans les outliers

Application d'un modèle de Machine Learning :

Utilisation d'un RandomForest

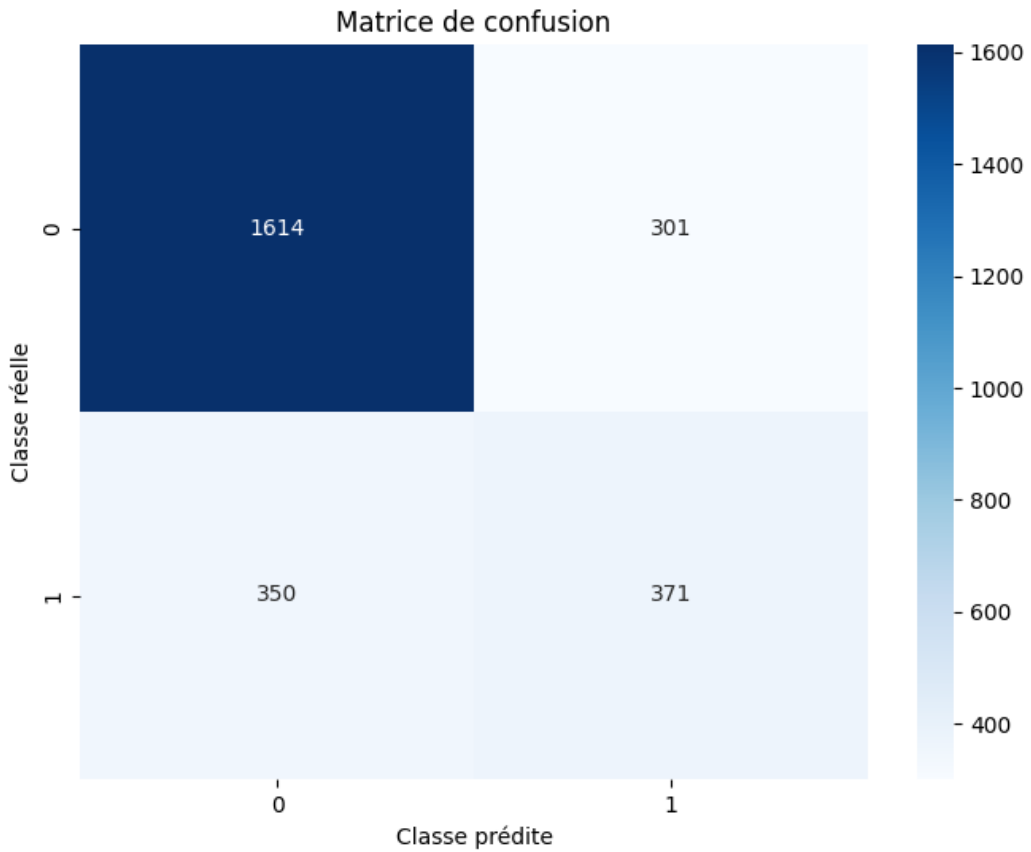

Figure: Matrice de confusion de l'algorithme Random Forest

Application d'un modèle de Machine Learning :

Utilisation d'un RandomForest

Grid search incoming

- Accuracy : $\frac{TP+TN}{TP+FP+TN+FN} = 0.75$
- Precision Score : $\frac{TP}{TP+FP} = 0.53$
- Recall Score : $\frac{TP}{TP+FN} = 0.48$
- Balanced Accuracy Score : $\frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) = 0.68$
- F1 Score : $\frac{2 \times Precision \times Recall}{Precision+Recall} = 0.50$

Métrique	Définition	Valeur
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	0.75
Precision	$\frac{TP}{TP+FP}$	0.53
Recall	$\frac{TP}{TP+FN}$	0.48
Balanced Accuracy Score	$\frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right)$	0.68
F1-Score	$\frac{2 \times Precision \times Recall}{Precision+Recall}$	0.50

Application d'un modèle de Deep Learning :

Utilisation d'un réseau de neurones

Nous avons implémenté un FFN pour la prédiction de notre variable Y dont on donne ci-dessous l'architecture avec la fonction de perte **Binary Cross Entropy** et un learning rate de **0.001**

- Input Layer de dimension 10 correspondant aux 10 features de notre dataset que nous avons sélectionnés
- 1 Couche cachée avec 32 neurones et une fonction d'activation **Relu** ainsi qu'un **Dropout** de 0.1
- 1 couche cachée avec 64 neurones, fonction d'activation **Relu** suivi d'un **Dropout** de 0.1
- L'Output Layer où on applique une fonction d'activation **sigmoïde**

Celà se formalise mathématiquement de la manière suivante :

Avec : $W_1 \in \mathbb{M}_{32\times 10}$, $W_2 \in \mathbb{M}_{64\times 32}$, $W_{out} \in \mathbb{M}_{64\times 1}$, $\varphi(x) = x^+$ et $\Psi(x) = \frac{1}{1+e^{-x}}$

Application d'un modèle de Deep Learning :

Utilisation d'un réseau de neurones

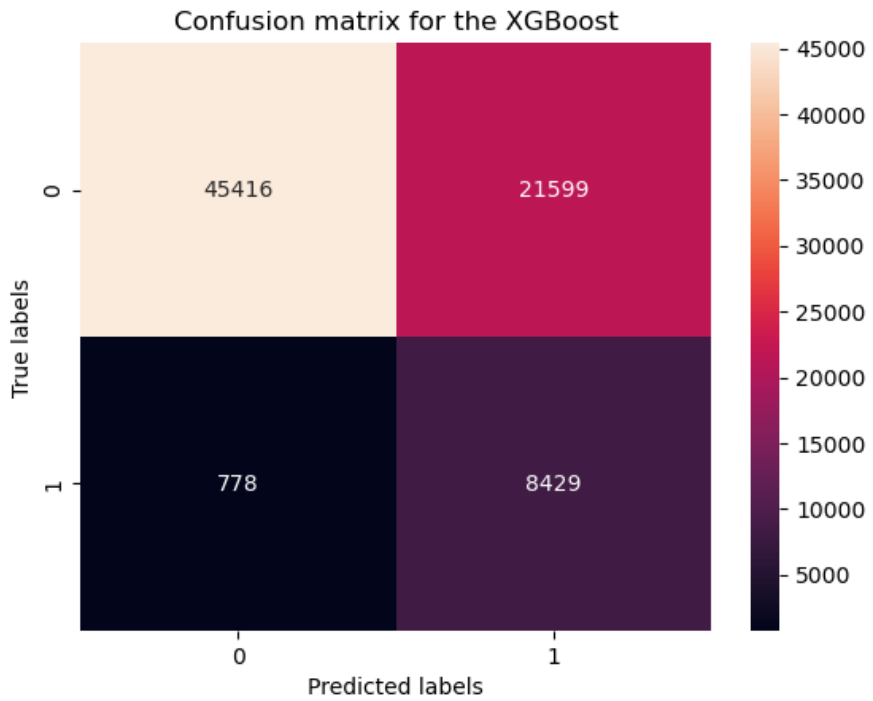

Figure: Matrice de confusion de l'algorithme de notre réseau de neurones

Application d'un modèle de Deep Learning :

Métriques obtenues avec le réseau de neurones

Métrique	Définition	Valeur
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	0.69
Precision	$\frac{TP}{TP+FP}$	0.28
Recall	$\frac{TP}{TP+FN}$	0.93
Balanced Accuracy Score	$\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$	0.68
F1-Score	$\frac{2\times Precision\times Recall}{Precision+Recall}$	0.42

Utilisation de l'application interactive Streamlit

Etude de notre dataset

Nous nous sommes également intéressés dans ce projet à l'utilisation de l'application **Streamlit** qui permet une analyse plus **user-friendly** de notre dataset.

`http://localhost:8501/`

Annexes

Matrice de confusion pour l'algorithme XGBoost

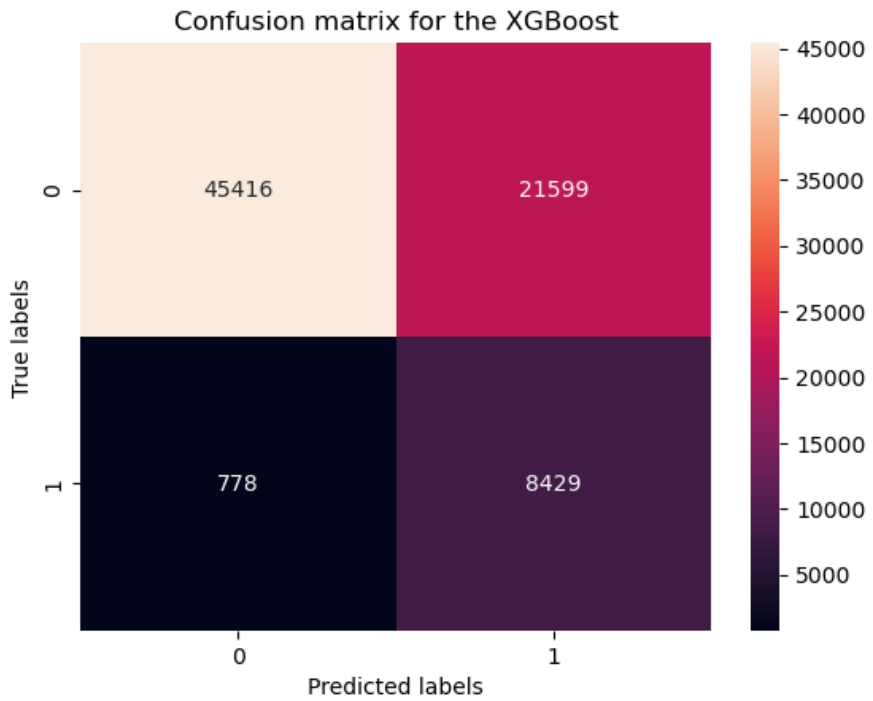

Figure: Matrice de confusion de l'algorithme de notre réseau de neurones

Annexes

Métriques obtenus avec l'algorithme XGBoost

Métrique	Définition	Valeur
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	0.71
Precision	$\frac{TP}{TP+FP}$	0.29
Recall	$\frac{TP}{TP+FN}$	0.91
Balanced Accuracy Score	$\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$	0.80
F1-Score	$\frac{2\times Precision\times Recall}{Precision+Recall}$	0.44

Annexes

Shape Value pour l'algorithme XGBoost

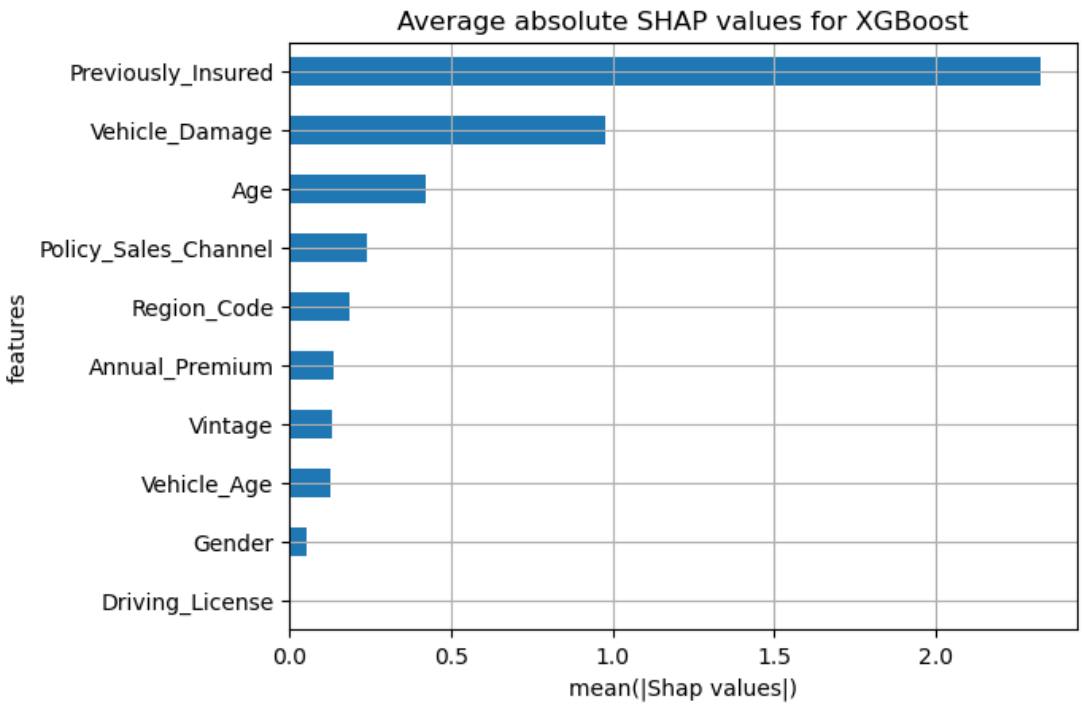

Figure: Matrice de confusion de l'algorithme XGBoost