KAGGLE: PORTO SEGURO'S SAFE DRIVE PREDICTION

Charles EVINA DJON
Paris

Contents

1 Présentation **3**
1.1 Informations sur les données . . . 3
1.2 Valeurs Nulles ou Manquantes . . . 4
1.3 Inspection de la cible . . . 6
1.4 Correlation . . . 6
1.4.1 'Float' variables . . . 7
1.4.2 'Integer' variables . . . 7

2 Modèles de Machine learning **9**
2.1 Recherche de grille (Grid Search) . . . 9
2.2 Validation croisée (Cross-Validation) . . . 9
2.3 Selection de modèles . . . 9
2.3.1 AUC (Area Under the Curve) . . . 10
2.3.2 Matrice de confusion . . . 10
2.4 La régression logistique . . . 10
2.5 Random Forest . . . 11
2.6 LightGBM . . . 12

3 Conclusion **13**

1 Présentation

Dans le cadre de ce projet, nous nous concentrerons sur la compétition Kaggle intitulée "Porto Seguro's Safe Drive Prediction". L'objectif de cette compétition est de prédire le coût d'un sinistre en fonction des caractéristiques des assurés. Le défi posé par cette compétition est de grande importance, car il vise à améliorer la capacité des compagnies d'assurance à évaluer et prévoir les risques liés à la conduite automobile. En utilisant des techniques avancées de science des données et de modélisation statistique, nous tenterons de créer un modèle précis qui permettra de prédire les coûts des sinistres en se basant sur un ensemble de variables pertinentes.

Pour mener à bien ce projet, nous suivrons une méthodologie rigoureuse qui comprendra les étapes suivantes :

- Exploration des données : Nous allons examiner attentivement les données fournies par Kaggle, comprendre les caractéristiques disponibles, identifier les éventuelles valeurs manquantes ou aberrantes, et explorer les relations entre les différentes variables.
- Prétraitement des données : Avant de construire notre modèle, nous allons nettoyer et préparer les données. Cela impliquera la gestion des valeurs manquantes, l'encodage des variables catégorielles, la normalisation des données numériques, etc.
- Sélection des caractéristiques : Nous évaluerons l'importance de chaque caractéristique pour la prédiction du coût du sinistre et sélectionnerons les variables les plus pertinentes pour construire notre modèle.
- Construction du modèle : Nous utiliserons différentes techniques de modélisation telles que la régression, les forêts aléatoires, ou les réseaux de neurones, pour construire un modèle capable de prédire avec précision le coût du sinistre.
- Optimisation du modèle : Une fois que nous aurons un premier modèle, nous chercherons à l'optimiser en ajustant les hyperparamètres et en utilisant des techniques d'optimisation pour améliorer ses performances.
- Validation croisée : Pour évaluer la robustesse de notre modèle, nous utiliserons des méthodes de validation croisée pour mesurer sa capacité à généraliser sur de nouvelles données.
- Prédiction des sinistres : Enfin, nous utiliserons notre modèle optimisé pour faire des prédictions sur les données de test fournies par Kaggle, afin de déterminer sa précision dans un environnement réel.

Ce projet offre une excellente opportunité d'apprendre et de mettre en pratique les compétences en science des données et en apprentissage automatique. En travaillant ensemble et en suivant une approche méthodique, nous pouvons espérer obtenir des résultats significatifs dans la compétition Kaggle "Porto Seguro's Safe Drive Prediction".

1.1 Informations sur les données

Les données sont présentées sous la forme traditionnelle de Kaggle, avec un fichier d'entraînement et un fichier de test. Chaque ligne correspond à un assuré spécifique et les colonnes décrivent

leurs caractéristiques. La variable cible est nommée commodément "target" ici et indique si cet assuré a fait une réclamation d'assurance dans le passé.

Dans les données d'entraînement et de test, les caractéristiques appartenant à des groupes similaires sont étiquetées en conséquence dans les noms des caractéristiques (par exemple, ind, reg, car, calc). De plus, les noms des caractéristiques incluent le suffixe bin pour indiquer les caractéristiques binaires et cat pour indiquer les caractéristiques catégorielles. Les caractéristiques sans ces désignations sont soit continues, soit ordinales. Les valeurs de -1 indiquent que la caractéristique était absente lors de l'observation. La colonne cible indique si une réclamation a été déposée ou non pour cet assuré.

Chaque ligne dans ces fichiers représente un assuré spécifique, et les colonnes décrivent différentes caractéristiques associées à ces assurés. Voici un aperçu des principales caractéristiques :

- target : La variable cible indiquant si l'assuré a déposé une réclamation d'assurance (1) ou non (0).
- ind : Caractéristiques commençant par "ind" qui sont regroupées ensemble.
- reg : Caractéristiques commençant par "reg" qui sont regroupées ensemble.
- car : Caractéristiques commençant par "car" qui sont regroupées ensemble.
- calc : Caractéristiques commençant par "calc" qui sont regroupées ensemble.
- bin : Caractéristiques binaires qui prennent des valeurs 0 ou 1.
- cat : Caractéristiques catégorielles qui peuvent avoir plusieurs catégories distinctes.
- Valeurs de -1 : Ces valeurs indiquent que la caractéristique est manquante pour l'observation correspondante.

L'objectif de ce projet est de développer un modèle d'apprentissage automatique en utilisant les données d'entraînement afin de prédire avec précision si un assuré déposera ou non une réclamation d'assurance à partir des caractéristiques fournies.

Nous allons explorer plus en détail les données, effectuer une analyse approfondie, préparer les données pour la modélisation et utiliser divers algorithmes pour créer un modèle prédictif performant. Enfin, nous évaluerons les performances du modèle sur les données de test pour voir à quel point il est capable de généraliser sur de nouvelles observations.

1.2 Valeurs Nulles ou Manquantes

```
train.isnull().any().any() —>> False
```

Malgré le résultat affichant "False" pour les valeurs nulles, nous savons d'après l'explication de Porto Seguro que les valeurs de -1 indiquent que la caractéristique était manquante lors de l'observation. Voyons quelles colonnes contiennent des valeurs de -1 :

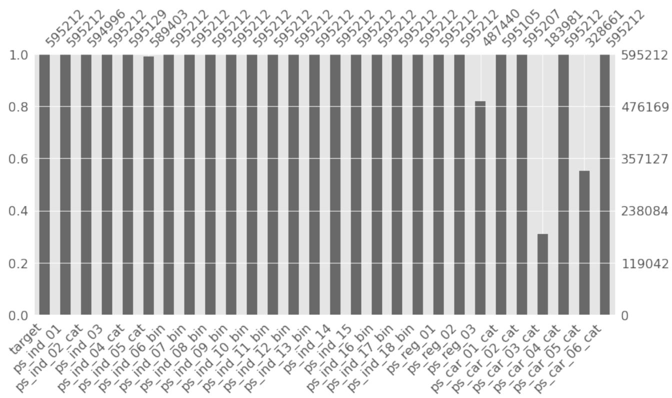

Figure 1: Mon image

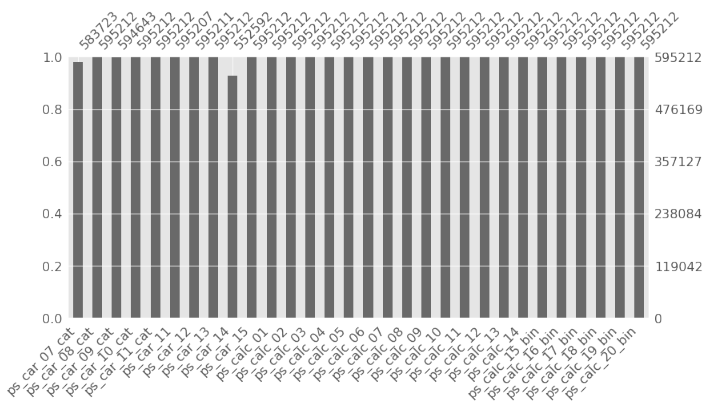

Figure 2: Mon image

Les cartes que 7 variables sur 59 de l'ensemble de données d'entraînement contiennent des valeurs nulles :

- *ps_ind_*05*_cat*
- *ps_reg_*03
- *ps_car_*03*_cat*
- *ps_car_*05*_cat*
- *ps_car_*07*_cat*
- *ps_car_*14

La majorité des valeurs manquantes se trouvent dans les caractéristiques *ps_reg_*03, *ps_car_*03*_cat* et *ps_car_*05*_cat*, où plus de la moitié des données sont probablement manquantes.

Les variables *ps_car_*03*_cat* et *ps_car_*05*_cat* ont respectivement 69% et 45% de valeurs manquantes : en raison d'une proportion élevée de valeurs manquantes, elles seront supprimées. Les autres données catégorielles seront conservées avec des valeurs manquantes -1.

- *ps_reg_03* a 18% de valeurs manquantes : la moyenne sera utilisée pour combler les lacunes, car il s'agit de données continues.
- *ps_car_14* a 7% de valeurs manquantes : la moyenne sera utilisée pour combler les lacunes, car il s'agit de données continues.
- *ps_car_12* a 1 enregistrement manquant : la moyenne sera utilisée pour combler les lacunes, car il s'agit de données continues.
- *ps_car_11* a 5 enregistrements avec des valeurs manquantes : le mode sera utilisé car il s'agit de données ordinales.

1.3 Inspection de la cible

Comme observé précédemment, la cible ne contient que deux valeurs, 0 et 1. Visualisons-les :

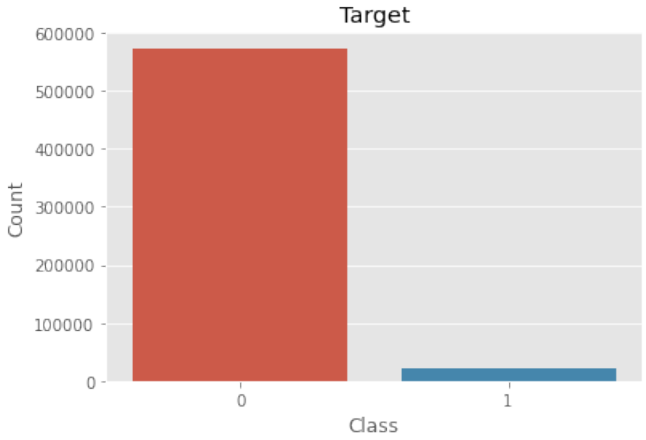

Figure 3: Mon image

On peut constater que notre variable cible est très déséquilibrée, avec une grande majorité de zéros et très peu de uns. Cette situation est courante dans le monde de l'assurance, où la plupart des individus n'ont pas de sinistres tandis que seulement quelques-uns en ont.On peut constater que nous avons un variable civ-ble qui est tres désiquilibré avec beaucoup de 0 et tres peu de un. cette situation est très présente dans le monde de l'assurance car nous avons beaucoup d'individus sans sinsitre et quelques individue ont des sinsitres.

1.4 Correlation

Comme indiqué précédemment, l'ensemble de données est composé de données de type entier (integers) et réel (float). Séparons-les pour obtenir la corrélation entre les variables :

1.4.1 'Float' variables

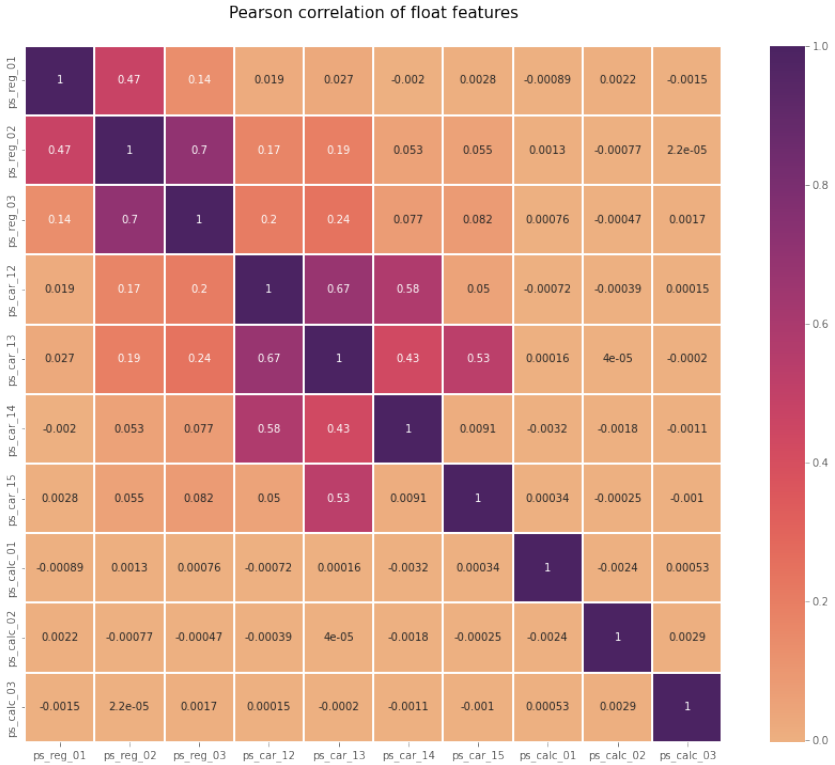

Figure 4: Mon image

La plupart des variables présentent une corrélation proche de zéro ou aucune corrélation entre elles. Les seules qui montrent une corrélation linéaire positive sont les suivantes :

- ps_reg_01 et ps_reg_02
- ps_reg_01 et ps_reg_03
- ps_reg_02 et ps_reg_03
- ps_car_12 et ps_car_13
- ps_car_13 et ps_car_15

1.4.2 'Integer' variables

Étant donné que le nombre de caractéristiques entières est beaucoup plus élevé que celui des caractéristiques réelles, la heatmap rendra impossible la lecture des corrélations à l'intérieur de chaque carré. C'est pourquoi la heatmap interactive a été choisie.

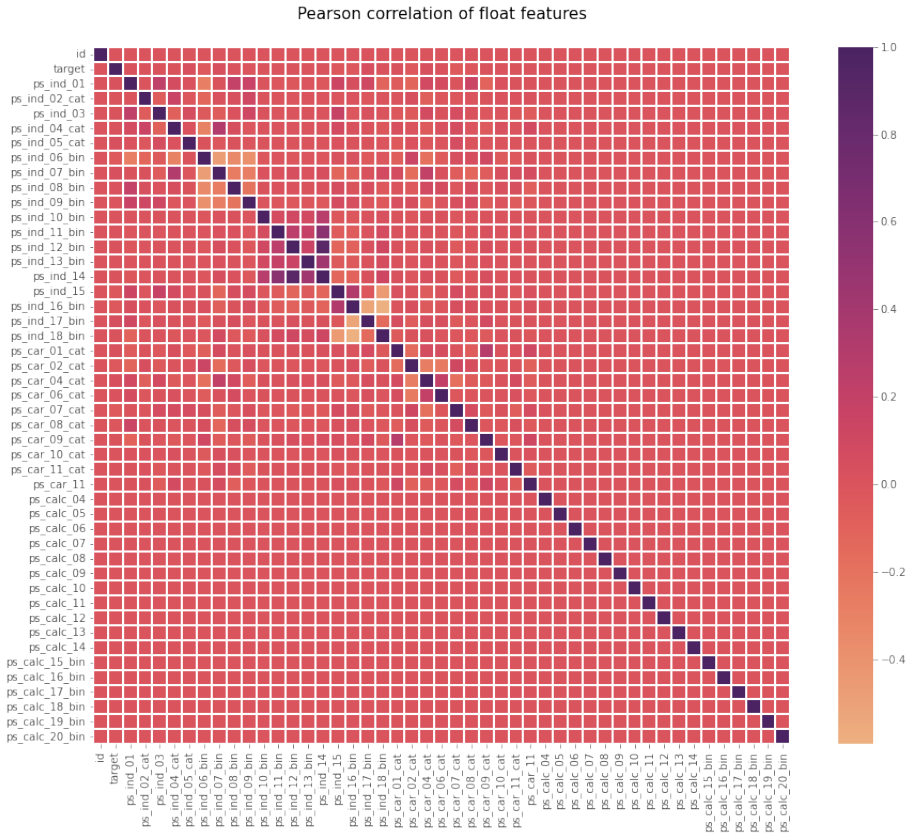

Figure 5: Mon image

Comme précédemment, la plupart des caractéristiques présentent une corrélation linéaire proche de zéro ou aucune corrélation entre elles. Dans cette heatmap, nous pouvons également observer certaines caractéristiques ayant une corrélation négative :

- *ps_ind_07_bin*
- *ps_ind_06_bin*
- *ps_ind_16_bin*
- *ps_ind_17_bin*
- *ps_ind_18_bin*

La corrélation positive la plus élevée à noter est entre :

- *ps_ind_14* / *ps_ind_12_bin* : z = 0.89
- *ps_ind_14* / *ps_ind_11_bin* : z = 0.56

Une corrélation intéressante à noter est entre *ps_car_03_cat* / *ps_car_05_cat* (z = 0.489). Les deux variables contiennent la plupart de leurs valeurs manquantes, ce qui pourrait expliquer la corrélation raisonnable entre elles.

2 Modèles de Machine learning

Nous allons tester plusieurs modèles, en commençant par le plus simple qui consiste à utiliser une régression logistique classique. Ensuite, nous évaluerons le Random Forest, le Gradient Boosting et le Lightgbm.

Pour chacun de ces modèles, nous mettrons en place un système de validation croisée en divisant notre ensemble d'apprentissage en ensemble d'entraînement et ensemble de validation. Nous entraînerons nos modèles sur l'ensemble d'entraînement et les évaluerons sur l'ensemble de validation. Cette expérience sera répétée plusieurs fois, et nous prendrons la moyenne des résultats sur chaque ensemble de validation.

Nous serons particulièrement attentifs au sur-apprentissage pour chaque algorithme et nous appliquerons une régularisation importante pour l'empêcher.

Nous allons également effectuer une recherche de grille (grid search) pour chaque modèle en utilisant la validation croisée (cross-validation).

2.1 Recherche de grille (Grid Search)

La recherche de grille est une technique utilisée pour sélectionner les meilleurs hyperparamètres d'un algorithme d'apprentissage automatique. Les hyperparamètres sont les paramètres que nous devons définir avant de lancer l'apprentissage du modèle, et ils influencent la performance et la complexité du modèle. Par exemple, dans un modèle de régression logistique, l'hyperparamètre peut être le taux d'apprentissage ou la régularisation. La recherche de grille consiste à définir une "grille" de valeurs possibles pour chaque hyperparamètre. Ensuite, nous testons toutes les combinaisons possibles de ces valeurs pour trouver la combinaison qui donne la meilleure performance du modèle. Pour chaque combinaison d'hyperparamètres, nous entraînons le modèle et l'évaluons sur un ensemble de validation.

2.2 Validation croisée (Cross-Validation)

La validation croisée est une méthode d'évaluation des performances d'un modèle qui consiste à diviser l'ensemble de données en plusieurs parties, appelées "plis" (folds). L'idée est de s'assurer que le modèle est bien généralisable et qu'il ne surapprend pas aux données d'apprentissage. Le processus de validation croisée se déroule comme suit :

1. Divisez l'ensemble de données en k plis. 2. Entraînez le modèle sur k-1 plis et évaluez-le sur le pli restant. 3. Répétez cette étape k fois, de manière à ce que chaque pli soit utilisé comme ensemble de validation une fois. 4. Calculez la moyenne des performances du modèle sur les k évaluations pour obtenir une estimation plus robuste de ses performances.

La validation croisée est particulièrement utile lorsque l'ensemble de données est limité, car elle permet d'obtenir une estimation plus fiable de la performance du modèle en utilisant toutes les données disponibles. En utilisant la validation croisée avec la recherche de grille, nous pouvons sélectionner les meilleurs hyperparamètres de manière plus robuste et éviter de biaiser notre choix en fonction d'une seule division des données.

2.3 Selection de modèles

Pour classer nos modeles, nous allons utiliser l'AUC et la matrice e confusion.

2.3.1 AUC (Area Under the Curve)

L'AUC (Area Under the Curve) est une mesure d'évaluation couramment utilisée en apprentissage automatique pour évaluer la performance d'un modèle de classification binaire. Cette mesure quantifie la capacité d'un modèle à distinguer entre les classes positives et négatives.

Dans le contexte de la courbe ROC (Receiver Operating Characteristic), l'AUC représente la surface sous la courbe ROC. La courbe ROC est un graphique qui représente le taux de vrais positifs (True Positive Rate - TPR) en fonction du taux de faux positifs (False Positive Rate - FPR) pour différents seuils de classification. Un modèle idéal aurait un AUC de 1, indiquant qu'il a une capacité parfaite à distinguer les classes positives et négatives. Un modèle qui prédit aléatoirement aurait un AUC de 0.5, car sa courbe ROC serait une ligne droite diagonale.

2.3.2 Matrice de confusion

La matrice de confusion est une table qui permet d'évaluer les performances d'un modèle de classification en comparant les prédictions du modèle aux valeurs réelles de la classe cible.

Elle est généralement de forme 2x2 pour un problème de classification binaire et ressemble à ceci :

* Vrai positif (True Positive - TP) : le modèle prédit correctement une instance de la classe positive (la classe que l'on souhaite prédire). * Faux positif (False Positive - FP) : le modèle prédit à tort une instance de la classe positive. * Faux négatif (False Negative - FN) : le modèle prédit à tort une instance de la classe négative (la classe que l'on ne souhaite pas prédire). * Vrai négatif (True Negative - TN) : le modèle prédit correctement une instance de la classe négative.

La matrice de confusion permet de calculer différentes métriques d'évaluation du modèle, telles que la précision, le rappel, le F-score, etc. Elle offre une vue globale des performances du modèle en mettant en évidence les erreurs de classification effectuées par celui-ci.

2.4 La régression logistique

Le premier modèle est un modèle paramétrique utilisé pour la classification binaire.

La régression logistique présente plusieurs avantages dans le contexte de notre étude. Tout d'abord, elle est facile à mettre en œuvre et à interpréter, ce qui facilite sa mise en pratique. De plus, les paramètres prédits par la régression logistique fournissent des informations sur l'importance de chaque caractéristique, ce qui peut être très utile pour comprendre les facteurs qui influencent les prédictions du modèle. De plus, ce modèle a montré de bonnes performances sur des données à faible dimensionnalité, ce qui est important pour notre jeu de données actuel. En outre, la régression logistique est très efficace lorsque l'ensemble de données a des caractéristiques qui sont linéairement séparables, ce qui est un avantage supplémentaire dans notre contexte.

Malgré ses avantages, la régression logistique présente également certaines limites dans notre cas. Elle est sujette au surapprentissage sur des données à haute dimensionnalité, ce qui peut réduire ses performances prédictives. De plus, ce modèle a une surface de décision linéaire, ce qui signifie qu'il ne peut pas résoudre des problèmes non linéaires. Une autre hypothèse de la régression logistique est la linéarité entre les variables dépendantes

et indépendantes, ce qui peut être une limitation dans certains cas où les relations entre les variables sont complexes. Enfin, il est essentiel de sélectionner avec soin les caractéristiques importantes et pertinentes pour éviter que la valeur prédictive du modèle se dégrade.

Lorsqu'on applique ce modèle sur nos données, on obtient un AUC score de 0.6283755131740453. Les variables les plus importantes obtenues à partir du modèle de régression logistique sont :

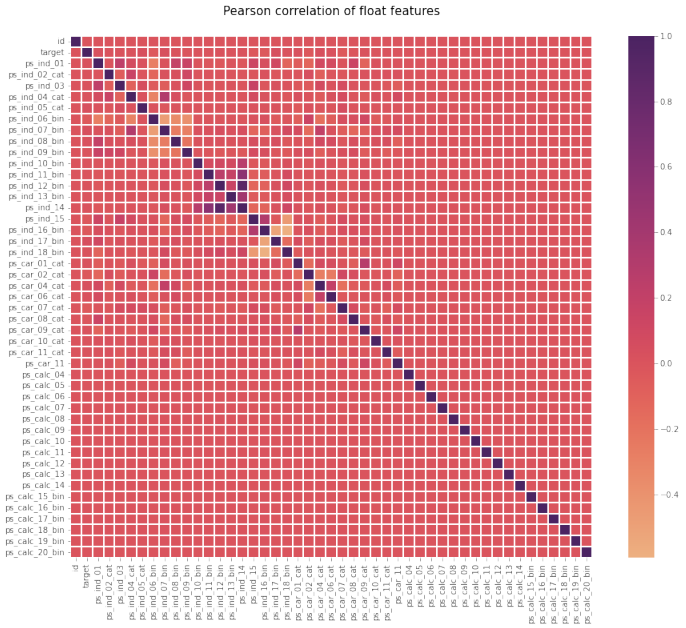

Figure 6: Mon image

2.5 Random Forest

Avantages et inconvénients de l'algorithme Random Forest :

Les avantages de l'algorithme Random Forest sont les suivants :

* L'algorithme Random Forest peut être utilisé pour résoudre à la fois des problèmes de classification et de régression. * Il est considéré comme un modèle très précis et robuste car il utilise un grand nombre d'arbres de décision pour effectuer des prédictions. * Les forêts aléatoires prennent la moyenne de toutes les prédictions faites par les arbres de décision, ce qui annule les biais. Ainsi, il ne souffre pas du problème de surajustement (overfitting). * Le classificateur Random Forest peut gérer les valeurs manquantes. Il existe deux façons de gérer les valeurs manquantes : utiliser les valeurs médianes pour remplacer les variables continues ou calculer la moyenne pondérée des proximités des valeurs manquantes. * Le classificateur Random Forest peut être utilisé pour la sélection des caractéristiques. Cela signifie sélectionner les caractéristiques les plus importantes parmi celles disponibles dans l'ensemble de données d'entraînement.

Les inconvénients de l'algorithme Random Forest sont les suivants :

* Le plus grand inconvénient des forêts aléatoires est leur complexité computationnelle. Les forêts aléatoires sont très lentes pour effectuer des prédictions car un grand nombre d'arbres de décision est utilisé pour les prédictions. Tous les arbres de la forêt doivent faire une prédiction pour la même entrée, puis effectuer un vote. C'est donc un processus long.

* Le modèle est difficile à interpréter par rapport à un arbre de décision, où nous pouvons facilement faire une prédiction.

Lorsqu'on applique ce modele sur nos données on obtient un AUC score: 0.6290513140332482. Voici les variables les plus importante:

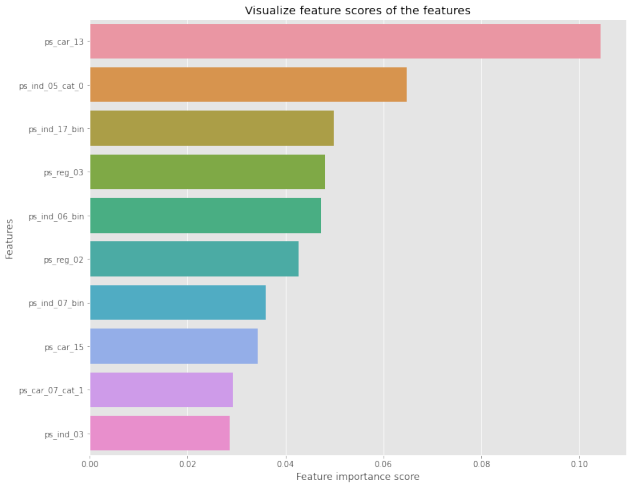

Figure 7: Mon image

2.6 LightGBM

Les avantages de l'algorithme LightGBM sont les suivants :

* L'algorithme LightGBM est rapide et efficace en termes de vitesse d'entraînement et de prédiction. Il utilise des techniques de gradient boosting pour améliorer la vitesse de formation des arbres. * Il peut gérer de grands ensembles de données et présente une grande évolutivité. * LightGBM utilise des techniques de division basées sur l'histogramme, ce qui le rend plus adapté aux ensembles de données avec un grand nombre de fonctionnalités ou avec des valeurs manquantes. * Il offre des performances de pointe et est généralement très précis dans ses prédictions. * LightGBM permet de contrôler la complexité du modèle en ajustant des paramètres tels que la profondeur des arbres et le nombre d'itérations.

Les inconvénients de l'algorithme LightGBM sont les suivants :

* Comme les autres modèles basés sur le gradient boosting, LightGBM est sensible aux données bruitées et peut surajuster (overfit) sur des ensembles de données trop petits. * Bien que LightGBM soit efficace en termes de mémoire, il peut quand même consommer plus de mémoire que des modèles linéaires plus simples. * L'interprétation du modèle peut être plus complexe par rapport à des modèles linéaires ou des arbres de décision, car il s'agit d'un modèle basé sur des ensembles d'arbres de décision. * En résumé, l'algorithme LightGBM est un excellent choix pour les ensembles de données volumineux, avec un grand nombre de fonctionnalités et nécessitant des prédictions rapides et précises. Cependant, il est important de prendre en compte le potentiel de surajustement et de sélectionner judicieusement les paramètres pour obtenir les meilleurs résultats.

Lorsqu'on applique ce modele sur nos données on obtient un AUC score de 0.6412083584132896. Voici les variables les plus importante:

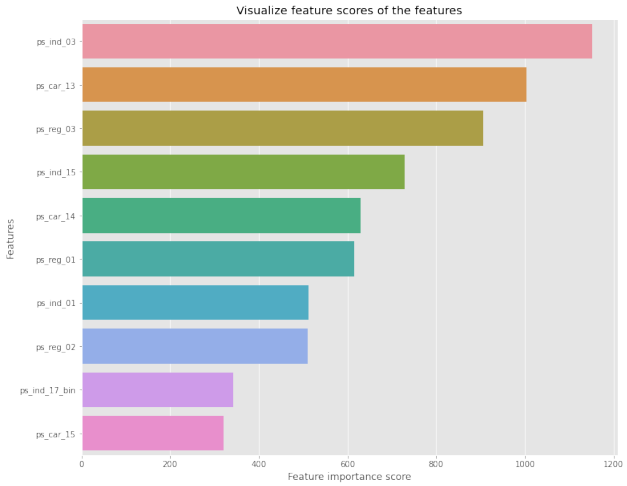

Figure 8: Mon image

3 Conclusion

D'après les résultats obtenus, nous pouvons tirer la conclusion suivante :

Parmi les trois modèles évalués, LightGBM a obtenu le meilleur résultat avec une Aire sous la courbe ROC (AUC) de 0.6412. Cela signifie que le modèle LightGBM a une meilleure capacité à distinguer entre les classes positives et négatives que les autres modèles.

La régression logistique a obtenu un AUC de 0.6283, ce qui est légèrement inférieur à celui du Random Forest qui est de 0.6290. Bien que la différence entre les performances de ces deux modèles soit faible, LightGBM a réussi à surpasser les deux autres en termes de performances prédictives.

En conclusion, pour cette tâche de classification, LightGBM se révèle être le modèle le plus performant parmi ceux testés, offrant une meilleure discrimination entre les classes cibles. Il serait donc préférable de choisir LightGBM pour de futurs projets nécessitant une prédiction précise dans des ensembles de données similaires. Cependant, il est toujours important de prendre en compte d'autres facteurs tels que la complexité du modèle, les ressources informatiques disponibles et la taille de l'ensemble de données avant de prendre une décision finale.