Benjamin SOULEAU - 2202532
Maxime ATZA

Projet de Statistiques Descriptives

2021-2022

Sommaire

Code

Le code de ce projet se sépare en plusieurs parties, tout en suivant l'arborescence demandée dans le sujet.

Tout d'abord, nous avons voulu mettre en place nos dataframe sous le pourcentage suivant :

- 60% de données d'entraînement
- 20% de données de test
- 20% de données de validation

Cependant, dû à une quantité trop grande d'observations, tous les tests de modèles ont été réalisés sous une répartition bien plus faible.
En effet, malgré la certaine puissance que nos ordinateurs pouvaient posséder, le nombre d'informations à stocker restait beaucoup trop grand pour traiter 60% des données fournies.

Dans ce projet, nous avons essayé différents modèles, plus ou moins fonctionnels. Certains ne fonctionnaient pas du tout mais nous avons tenu à laisser le code pour garder une trace de chacune de nos recherches.

Rôle des fichiers R

- Main.r - Il permet de sélectionner le DF et de le split afin d'obtenir un échantillon correspondant à 10% du DF afin de faciliter le training en le rendant plus rapide et en détériorant au minimum les portions de données.
- Visualisation.r - Donne les visuels correspondant aux corrélations entre les différentes variables, aux analyses des données et aux intervalles.
- Neuralnet.r - Fichier créant un réseau de neurones en vérifiant ses performances, le contenu est fonctionnel mais montre un taux d'erreur beaucoup trop élevé (vérification de la performance du réseau malheureusement non fonctionnel).
- Arbre.r - Donne le modèle final avec les meilleures performances possibles
- Logistic.r - Donne le modèle avec valeurs nulles (cf. Modèle)
- Classifier.r - Donne le modèle avec des valeurs beaucoup trop basses (cf. Modèles)

Les lignes de définition des données d'entraînement et de test sont dans le fichier Main.R entre les lignes 19 à 22.

Explication du code

Pour la réalisation des échantillons nous avons utilisé la fonction *setDiff* qui compare deux dataframes pour récupérer les observations non utilisées dans le dataframe d'entraînement.
Pour ce qui est du choix des variables explicatives, nous avons décidé d'utiliser les variables *type, amount, nameOrig, nameDest, isFraud, isFlaggedFraud* car nous en avons déduit qu'elles devaient être les plus importantes.
Pour la visualisation des données, les fonctions utilisées sont celles vues en TD avec les librairies GGally et ggplot2.

Réseau neuronal
Le réseau de neurones quant à lui utilise la librairie neuralnet qui forme un réseau pour expliquer la variable *isFraud* selon les variables *amount, nameOrig, nameDest*.
Pour l'optimisation du réseau de neurones, nous avons travaillé en faisant différentes couches neuronales.
Nous avons aussi utilisé une fonction afin de calculer l'EQM pour évaluer la folrce de ce modèle.

Régression linéaire multiple
Recherche du meilleur modèle grâce à la méthode stepwise qui permet de choisir le meilleur modèle en testant toutes le possibilités en utilisant la variante:

- forward : on commence par un modèle vide, que l'on remplit jusqu'au modèle complet pour trouver le plus adapté.

Nous avons basé l'étude des performances grâce au critère AIC qui est régulièrement utilisé pour ce type de modèle. Nous avons ensuite calculé l'EQM pour valider notre modèle.

Régression logistique
Nous avons utilisé ce type de modèle qui est adapté à la variable *isFraud*. Nous avons aussi fait appel à cette méthode mais cette fois-ci avec la variante

- backward : on part du modèle le plus complet et on le réduit jusqu'à trouver le plus adapté.

Là aussi l'AIC a été utilisé pour choisir le modèle le plus adapté.

Régression Lasso
Nous avons commencé par transformer les données d'apprentissage et de test en centrant et réduisant.
Nous avons testé divers lambdas pour voir le plus adapté, càd celui minimisant l'erreur. Ici encore, nous avons calculé l'EQM pour évaluer les compétences du modèle.

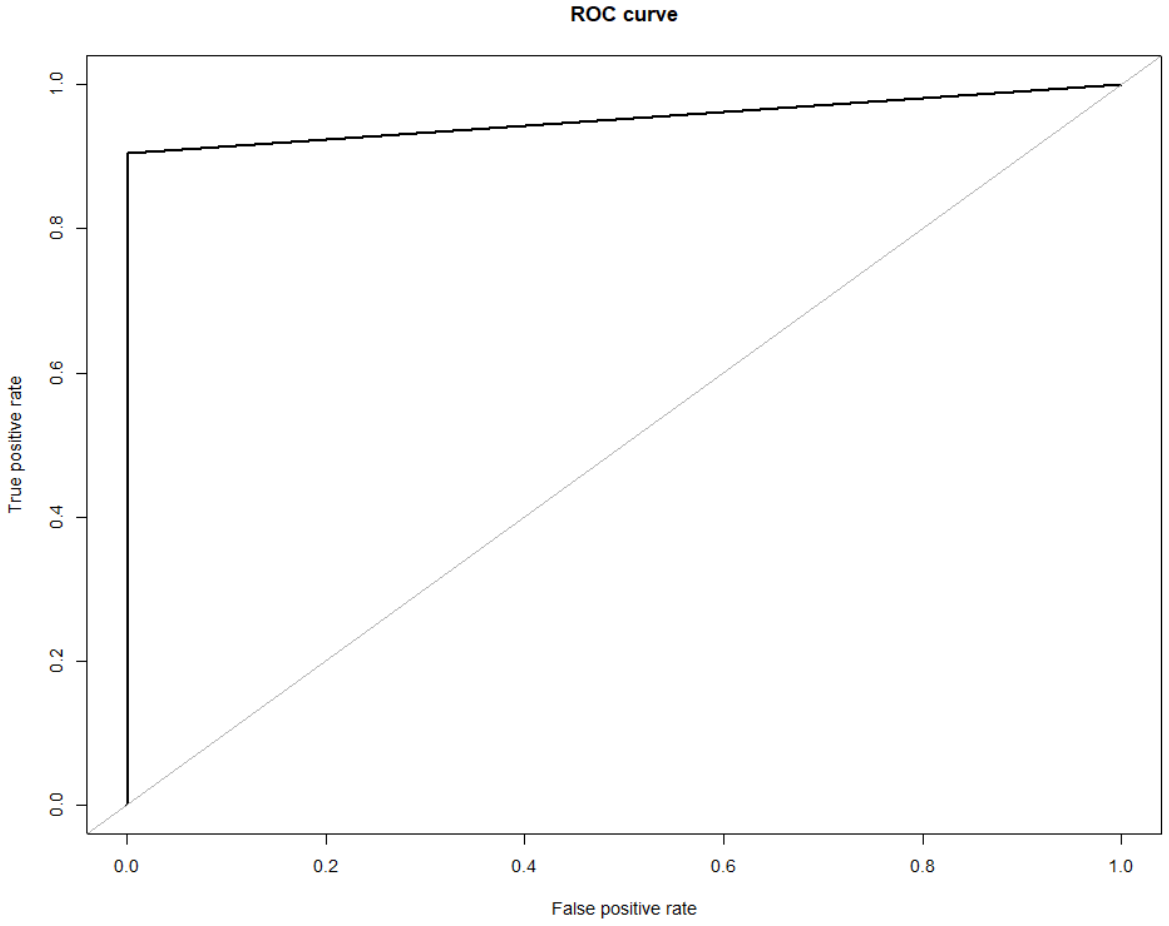
Modèles sélectionnés

5 modèles ont donc été testés pour analyser ce DF de la façon la plus performante et la plus précise possible.

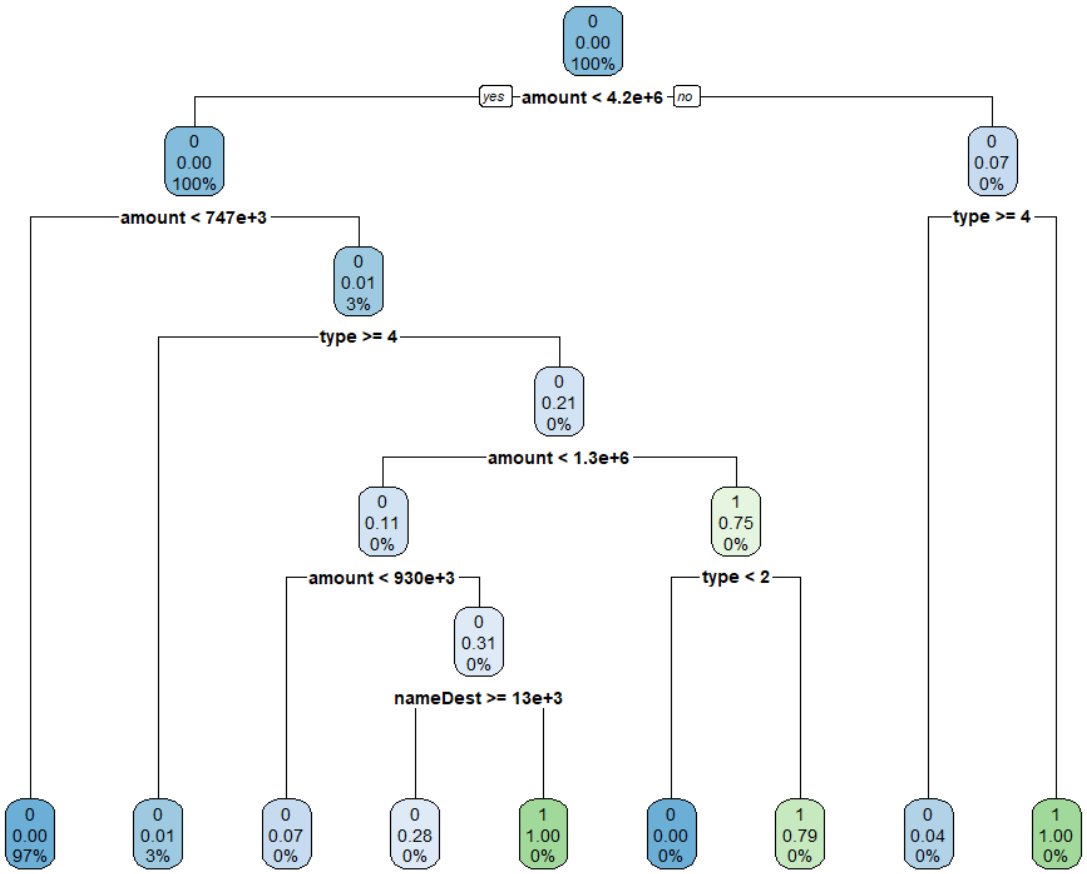
- Arbre de classification
- Régression Logistique
- Réseau de neurones
- Classifieur bayésien naïf
- Régression linéaire multiple

Suite à une erreur de notre part, les données qualitatives ont été factorisées puis mises sous forme d'entiers, nos tests et modèles sont alors biaisés et donc ils ne sont pas significatifs. Voici quand même les sorties de l'arbre de classification nous ayant donné les meilleures performances.

Précision du modèle : 0.904761904761905
F1 Score : 0.238993710691824
AUC : 0.951760485310026



Arbre de classification (Biaisé) :

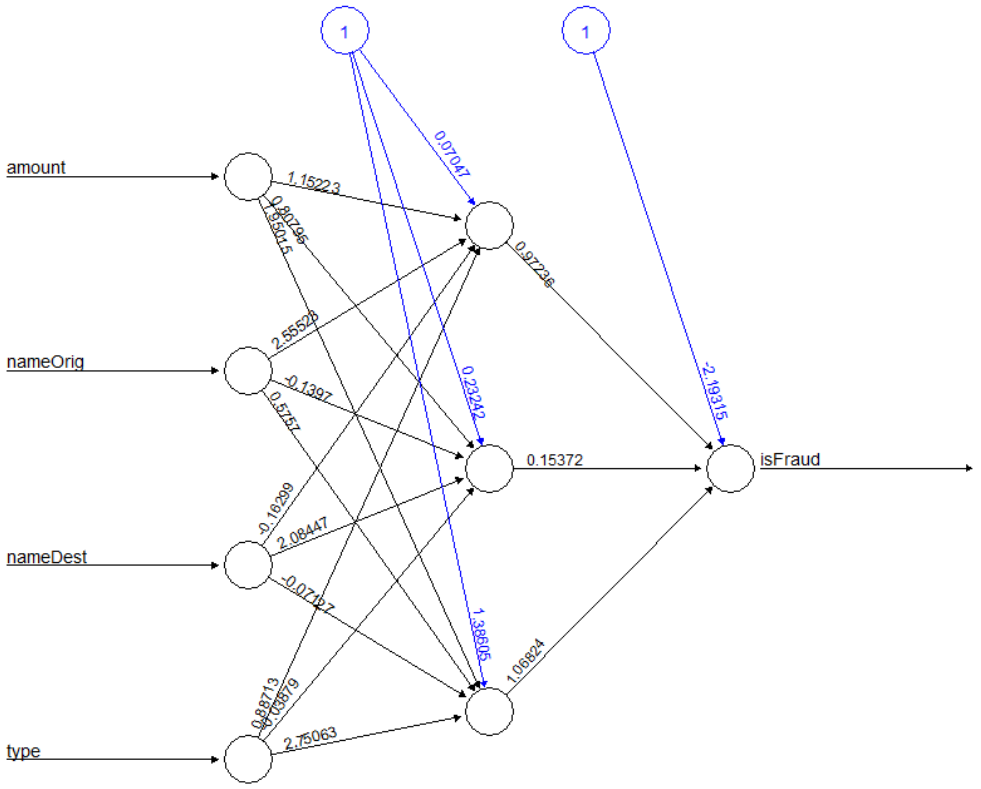


Ces performances peuvent paraître hautes (hormis le F1 Score), mais comme énoncé plus haut, notre modèle n'a aucun intérêt. En effet, les variables *nameDest* *nameOrig* ont été factorisées puis mises sous forme d'entier, ce qui fait que leur intérêt à été réduit à l'état de nombre qui n'a au final plus aucun sens.

Nous tenons cependant à garder l'intégralité du code suite à nos recherches pour ce qui est des réseaux de neurones et du modèle de régression linéaire avec la librarie lasso. Nous avons essayé avec des jeux de données plus petits et cela fonctionnait.

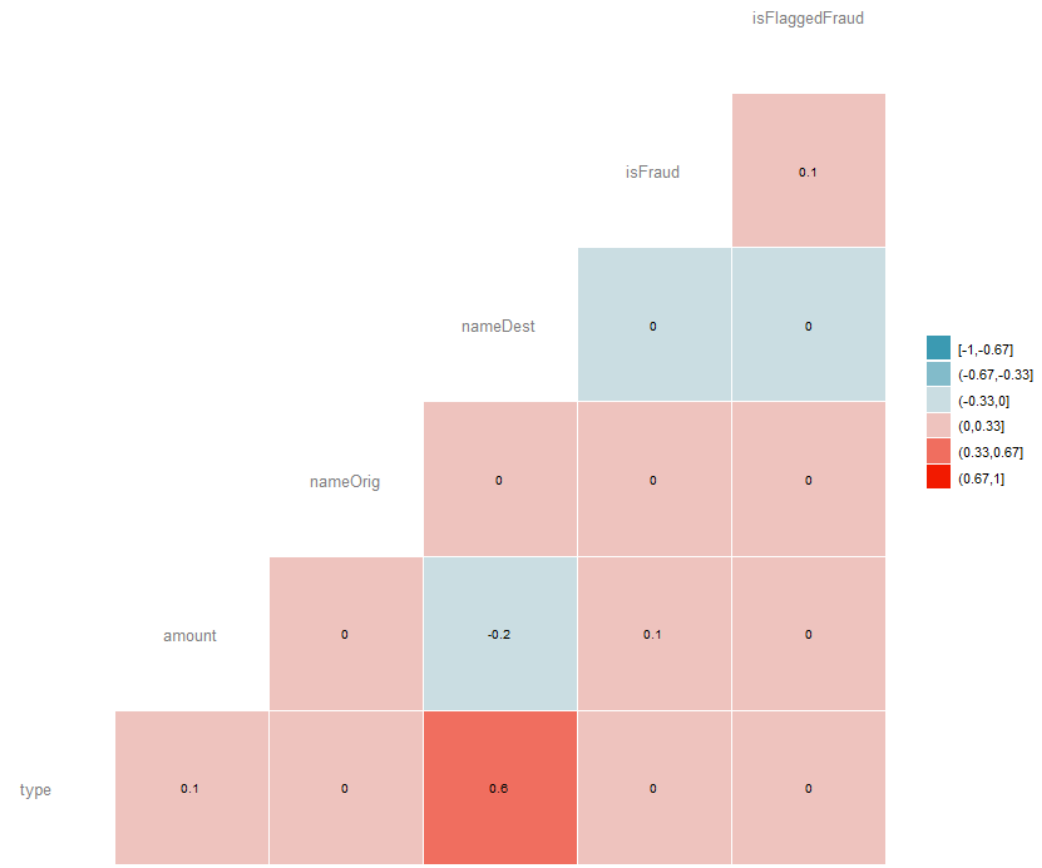
Analyses

Réseau de neurone permettant d'expliquer les implications de toutes les variable dans la variable isFraud (taux d'erreur trop hauts et valeurs non significatives) :



Error: 2.497067 Steps: 52

Matrice de corrélation entre chaque variable



Nous observons une corrélation positive significative entre le type de transaction et le nom du destinataire. Le type et le destinataire semblent donc évoluer dans le même sens, mais pas à la même vitesse.