Note méthodologique : Projet 7

# Introduction



Rappel de la mission :

Je suis Data Scientist au sein de la société financière « Prêt à dépenser ». Cette société est spécialisée dans l’octroi de crédit à la consommation auprès de particuliers n’ayant pas, ou un faible historique de prêt.

« Prêt à dépenser » souhaite mettre en place, afin d’étayer sa décision d’octroi ou non du crédit, un système de scoring client basé sur la probabilité que celui-ci soit en défaut de paiement. Ce modèle reposera sur un ensemble de données : financières, comportementales, autres...

Cette note a pour but de retracer la méthode qui a été mise en place tout au long de ce projet, jusqu’à l’obtention du modèle final.

# 1 prétraitement des données

Les données utilisées pour ce projet proviennent d’un fichier intitulé « application\_train.csv ». Sa taille est d’environ 307 000 lignes pour 122 colonnes.

La variable cible ici s’intitule ‘TARGET’ et représente le fait ou non de présenter des difficultés de paiement.

Il a été important, dans un premier temps, de recatégoriser certaines variables. En effet, nous pouvions retrouver dans le dataset, des colonnes dont le type était ‘int64’, soit, des variables numériques dont les données sont des entiers mais dont les valeurs étaient soit 0 ou 1. Il convient alors de le convertir en variables booléennes : ‘False’, ‘True’.

Dans un premier temps, des fonctions vont identifier les variables étant dans cette configuration et les afficher à l’utilisateur pour validation. Puis si l’utilisateur valide les colonnes identifiées, celles-ci sont alors converties.

(cf image 1 : Conversion des colonnes ‘Booléennes’)

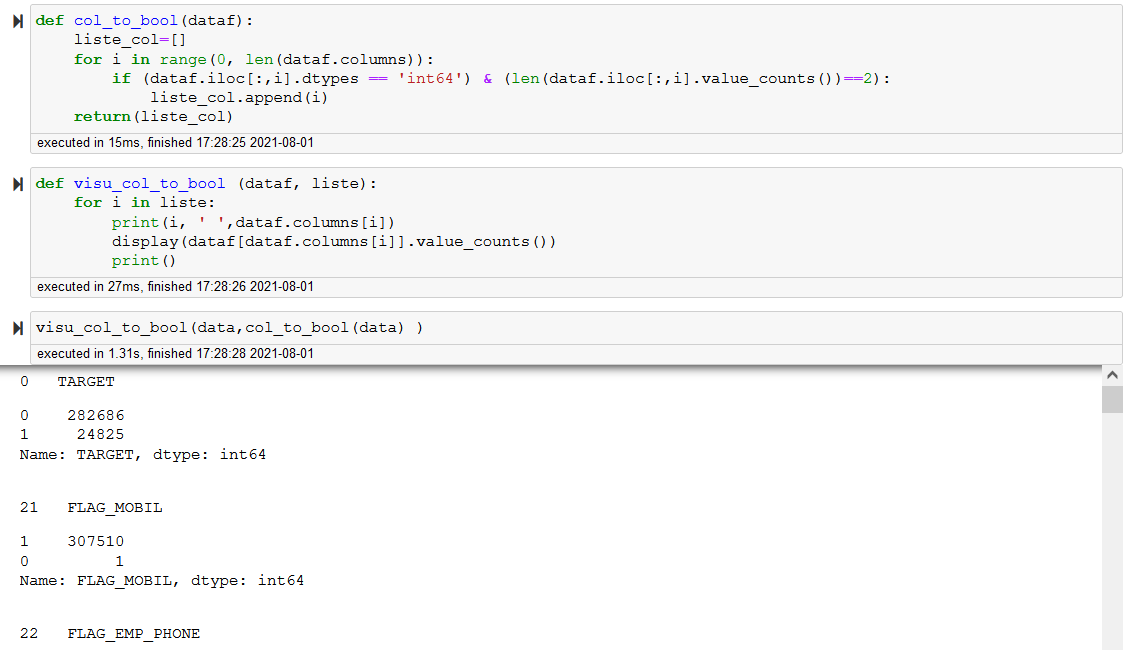


Figure 1 : Conversion en variables booléennes

A été opéré également la création d’une variable effectuant la moyenne mathématique de 3 variables : ‘AVG\_NOTE’



Par la suite, seront intégré dans une pipeline les transformations suivantes :

* Pour les variables numériques :
  + Imputation des valeurs manquantes par la médiane
  + Utilisation de la transformation robust scaler (la médiane est retirée des valeurs et celles-ci sont divisées par l’écart interquartile ). Cette approche est robuste aux valeurs aberrantes (‘outliers’).
* Pour les variables catégorielles :
  + Imputation des valeurs manquantes par la méthode de la valeur la plus fréquente dans la distribution.
  + Transformation des valeurs en valeurs numériques par l’utilisation du module ‘One-hot-Encoder’.

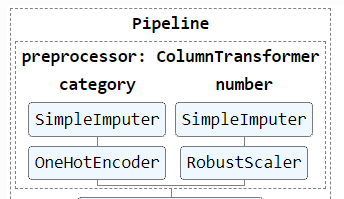


Figure : pipeline de pre processing

# 2 corrections du déséquilibre (cf SMOTE)

Nous sommes ici en présence d’un jeu de données déséquilibré par rapport à la variable cible. En effet, la base contient une très grande majorité de clients qui ne sont pas en difficulté de paiement (92% vs 8%) (CF image 2)

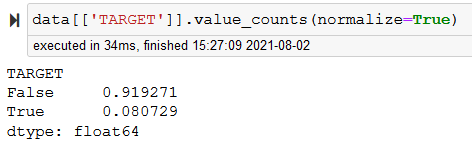


Figure 3 : répartition (%) valeur de la variable cible

En l’état, n’importe quel algorithme de classification aurait naturellement tendance à être fortement influencé par la proportion de la variable majoritaire. En effet, un algorithme qui prédirait uniquement la valeur ‘False’ obtiendrait une métrique d’évaluation basée sur la justesse (‘accuracy\_score’) élevée. Or l’algorithme n’apprendrait rien. Il convient, pour répondre à la problématique de l’entreprise, de veiller à deux points importants :

* Corriger le déséquilibre dans les données
* Se baser sur une métrique d’évaluation adaptée à la problématique.

Pour corriger le déséquilibre, deux opérations ont été menées. Dans un premier temps, de l’« under-sampling » a été effectué : des lignes comportant la variables cible majoritaire (‘False’) ont été écartées.

Les points positifs sont :

* Réduction du nombre de lignes au total, donc gain en rapidité d’exécution, réduction du coût en ressource machine.
* On vient réduire l’écart entre les deux valeurs de la variable cible.

Le point négatif est que nous faisons le choix de perdre de l’information avec ces lignes en moins.

De plus, a été effectué de l’« over-sampling » : à l’inverse des lignes comportant la variable cible minoritaire (‘True’) sont générées. Le module SMOTE\_NC de la librairie imabalanced\_learn a été utilisé. SMOTE fonctionne de la manière suivante : une ligne de la classe minoritaire est aléatoirement sélectionnée, puis, les 5 plus proches voisins sont identifiés. Va être alors sélectionné aléatoirement un des 5 voisins les plus proches, puis, une nouvelle ligne « synthétique » va être composée à partir des informations issues par ces deux points.

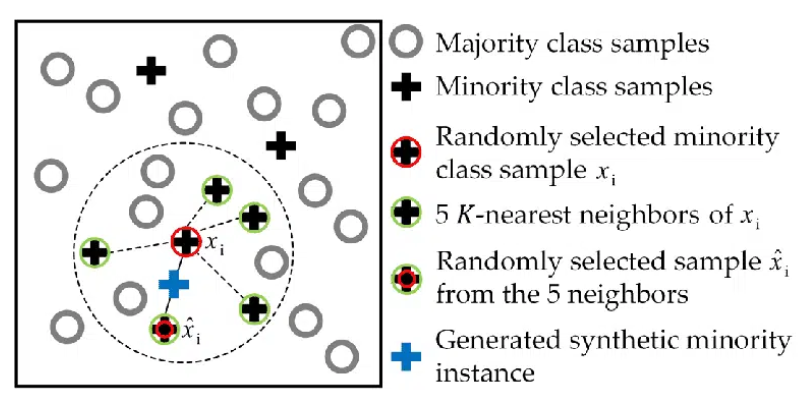


Figure : SMOTE

L’opération est alors répétée jusqu’à ce que les variables cibles soient en proportions égales l’une par rapport à l’autre.

SMOTE\_NC a été utilisé ici, car celui-ci intègre le fait que nous ayons des données catégorielles et numériques dans notre jeu de donnée.

# 3 métrique d’évaluation : f\_beta score

Nous sommes ici dans un problème de classification binaire avec un jeu de données déséquilibré.

Les résultats obtenus à la suite d’une classification peuvent se regrouper dans un tableau de synthèse appelé une matrice de confusion.

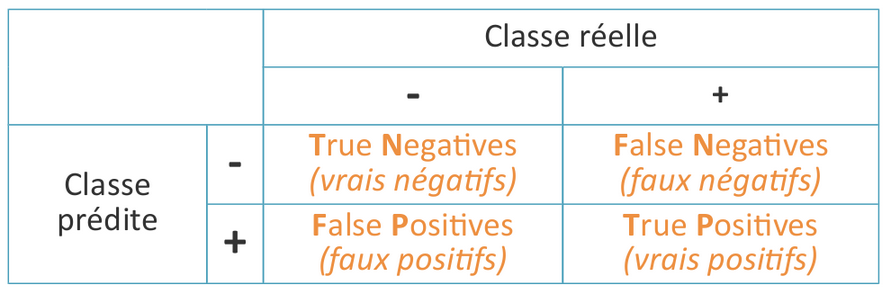
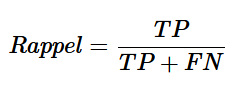


Figure : matrice de confusion

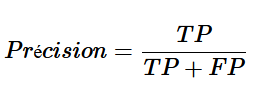
Dans le cas de l’entreprise :

* TN : clients n’étant pas en défaut de paiement
* FN : clients étant en défaut de paiement mais mal catégorisé par l’algorithme
* FP : clients n’étant pas en défaut de paiement mais écarté par l’algorithme
* TP : clients en défaut de paiement.

A partir de ces 4 éléments peuvent être calculées deux mesures : le rappel et la précision.



Le rappel correspond au taux de vrais positifs ; le pourcentage de ceux qui seront correctement identifiés. C’est une manière d’évaluer la capacité de notre modèle à correctement déceler les clients en défaut de paiement.



La précision correspond à la proportion de prédictions correctes parmi les points qui ont été prédits positifs.

Dans le cas de l’entreprise, il sera dommageable pour elle d’octroyer un crédit à un client présentant des difficultés de remboursement ; elle pourrait se retrouver dans la situation où celui-ci ne soit plus en capacité de retourner les sommes dues et l’entreprise perdrait de l’argent dans cette opération.

Il convient donc de bien veiller à identifier les clients en défaut de paiement sans pour autant générer trop de faux négatifs.

La métrique d’évaluation des modèles qui a été retenue ici est le f\_beta score.

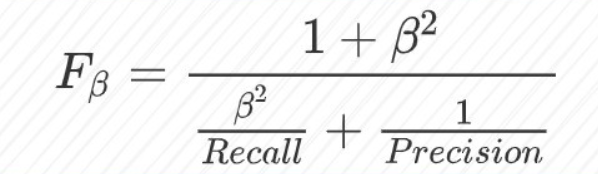
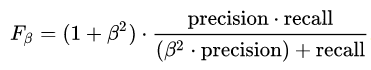


Figure : f\_beta score reformulation

Figure : f\_beta score

Dans cette approche, un coefficient beta vient pondérer les mesures de précision et de rappel. En effet, le rappel est considéré avoir beta-fois plus d’importance que la précision. Dans notre cas, beta a été fixé à 2.



# 4 sélection du modèle de classification & optimisation des hyperparamètres

Liste des modèles testés :

* Régression logistique
* XGB Classifier
* AdaBoost Classifier
* Random Forest Classifier
* ExtraTrees Classifier
* Linear SVC
* Linear Discriminant Analysis
* RidgeClassifier

Le modèle présentant les meilleurs résultats est le suivant :

**LogisticRegression** (max\_iter=350, C=0.6, class\_weight=w, solver = 'sag', random\_state=42)

F\_beta\_score = 0.54

Accuracy\_score = 0.63

Matrice de confusion du modèle :

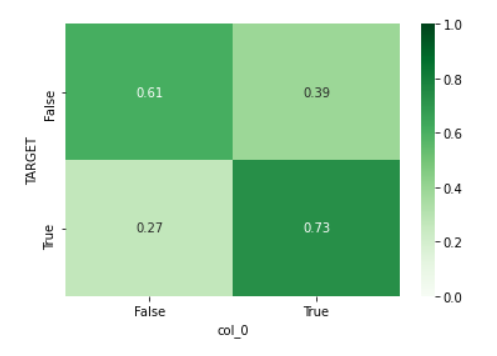


Figure : matrice de confusion modèle retenu

# 5 interprétabilité du modèle

Première approche d’interprétabilité : variables les plus impactantes du modèle.

* Notation du client provenant des organismes extérieurs
* Age du client
* Années d’ancienneté sur son poste
* Montant du prêt
* Prix du bien pour lequel est souscrit le prêt
* Revenu total du client

# 6 limites et piste d’améliorations.

* Il serait pertinent de travailler avec des outils de calcul plus puissant (AWS par exemple) afin de pouvoir travailler sur l’ensemble des clients (pas d’under sampling en amont).

Eventuellement pouvoir avoir une meilleurs accuracy sans dégrader le score f-beta.

* Affiner avec l’entreprise le poids beta de la métrique d’évaluation.
* Peaufiner le travail sur l’interprétabilité des variables du modèle.