**Note de Méthodologie**

Pour ce 7e projet du parcours Data Scientist d’Open Classrooms, l’objectif donné est de mettre en place un modèle de notation de candidats à un crédit à la consommation.

La « matière première » disponible est une base de données de crédits accordés, comprenant les données fournies par le client sur lui-même et les données financières du crédit, dont son issue finale : son remboursement (ou non) par le client.

Si je vais, dans la suite de cette note méthodologique, décrire plus en détail mon travail, commenter son résultat, et évoquer une partie de ses limites, le travail que j’ai effectué pour atteindre cet objectif pourrait être résumé très rapidement en quelques étapes :

- Préparation et mise en forme des données,  
- Choix des critères de sélection du modèle de classifieur,  
- Sélection des paramètres à optimiser pour chacun des modèles de classifieurs à tester,  
- Identification d’une plage de valeurs pertinente pour chacun des paramètres,  
- Premier affinage des paramètres pour chacun des modèles,  
- Sélection du modèle,  
- Affinage poussé des paramètres du modèle sélectionné,  
- Choix du seuil de décision pour accorder (ou non) le crédit,  
- Analyse du modèle obtenu : Feature importance.

**I - Rappels : Objectifs, données**

**II – Preprocessing et échantillonnage**

**III – Préparation et fit du modèle**

**1 – Première série d’itérations : choix des paramètres à optimiser**

**2 – Deuxième série d’itérations : identification des plages de valeur pertinentes**

**3 – Troisième série d’itérations : un premier affinage du modèle**

**4 – Choix du modèle utilisé par la suite**

**5 – Ajustement et fit du modèle**

**IV – Choix du seuil de décision**

**V – Une analyse du modèle obtenu : la Feature Importance**

**Conclusion**

**I - Rappels : Objectifs, données**

Pour comprendre la finalité du projet, il est important de prendre en compte les motivations du client. A défaut d’avoir un vrai client pour répondre à mes questions, je me suis mis à sa place.

Au-delà de la mise en place du modèle de classification, ce que veut le client, c’est de réduire les coûts créés par les défauts de paiement des clients.

Les données mises à notre disposition pour répondre à l’objectif sont les données liées aux crédits accordés à plus de 300 000 clients.

La variable centrale est la variable TARGET, indiquant si le client a fait défaut (1) ou s’il a correctement remboursé son emprunt (0). C’est elle qu’on tente de prédire.

Les 121 autres variables sont liées au crédit (montant, annuité, prix du bien, documents fournis, etc.) ou au client (âge, sexe, montant et nature des revenus, niveau d’éducation, détails sur le logement, le véhicule, le téléphone, etc.). Elles sont de différentes natures (numériques, catégorielles).

**II – Preprocessing et échantillonnage**

Cette étape m’a amené à prendre quelques décisions parfois radicales :

* Le preprocessing auquel j’ai procédé n’a pas été conçu par moi. J’ai réutilisé celui proposé par un candidat à la compétition originale (<https://www.kaggle.com/jsaguiar/lightgbm-with-simple-features> ).  
  Ce processing consiste à “Encoder” les données catégorielles, introduire différents taux (Credit/Annuity, Revenu/Nb Enfants, Annuity/Revenus, Nb jours travaillés/Nbrejours depuis naissance, etc.), rassembler les informations sur les précédentes applications des clients (credits acceptés et refuses), appliquer des min, max, sommes, moyennes et écarts-types.
* J’ai séparé la population totale en une partition train et une partition test.   
  Chacune des partitions comprend 50% des individus… ce qui est, en temps normal, trop peu pour une partition train. Je me suis limité à 50% pour une question de temps de calcul (certains calculs tournant pendant plusieurs jours). Les résultats que j’obtenais étant acceptables, j’en suis resté à 50%.  
  La partition a été faite en conservant la proportion de 0 et de 1 sur la variable TARGET.
* J’ai testé différentes méthodes de rééquilibrage de classes pour la variable TARGET : oversampling, undersampling, SMOTE.  
  J’ai conservé la méthode qui me donnait les résultats les plus satisfaisants : un simple weight=balanced pour les modèles qui le permettaient.

**III – Préparation et fit du modèle**

Cette 3e partie de mon travail s’appuie sur la méthode GridSearchCV, qui consiste à la fois en un GridSearch, i.e le parcours d’une « grille » de paramètres pour mesurer la performance de chaque jeu de valeurs, et une validation croisée (CV), soit le découpage en n plis de la population, puis une série de fits sur chacun des n ensembles possibles de n-1 plis, et de scorings sur chaque pli complémentaire.

J’ai choisi ici de travailler avec 5 plis (5 folds). Donc pour chaque jeu de paramètres, le fit sera fait sur 4 plis, et le modèle fitté sera testé sur le 5e pli.

Le score utilisé pour évaluer la qualité du modèle est le ROC-AUC.

J’ai essayé 5 modèles de classifieurs différents :

* + Random Forest Classifier,
  + Gradient Boosting Classifier,
  + Logistic Regression,
  + MLP Classifier,
  + Light GBM.

Cette partie du travail a été l’occasion de nombreuses itérations et aller et retours sur le choix des paramètres, et m’a fait faire de nombreux essais.

**1 – Première série d’itérations : choix des paramètres à optimiser**

J’ai commencé par sélectionner les paramètres sur lesquels travailler, pour chacun des classifieurs.

Après une consultation de la documentation officielle et de plusieurs exemples trouvés sur internet pour chaque classifieur, j’ai construit une première liste extensive de paramètres à tester.

En lançant des GridSearchCV à un seul paramètre, j’ai pu sélectionner les paramètres qui avaient un impact significatif sur le ROC-AUC. La méthode est très empirique, donc critiquable, mais a permis de commencer à réduire le champ de recherches.

Les paramètres retenus pour chaque modèle ont été :

|  |  |
| --- | --- |
| **Modèle** | **Paramètres** |
| Random Forest Classifier | min\_samples\_split  max\_features  max\_depth |
| Gradient Boosting Classifier | learning\_rate  max\_depth |
| Logistic Regression | C |
| MLP Classifier | solver  alpha  hidden\_layer\_sizes |
| Light GBM | boosting\_type  n\_estimators  learning\_rate  num\_leaves |

**2 – Deuxième série d’itérations : identification des plages de valeur pertinentes**

Dans cette deuxième série d’itérations, j’ai recherché la plage de valeurs pertinentes pour les paramètres (paramètres numériques).

J’ai pu, pour ce faire, m’appuyer sur la série d’essais de la première partie.

Le principe utilisé est simple : si, pour un des paramètres, la valeur qui donne le meilleur score est une valeur extrême, alors il faut translater la plage de valeurs en direction de cet extrême.

La méthode à nouveau très empirique, et si elle ne garantit pas de trouver un maximum global, elle permet au moins de trouver un maximum local.

**3 – Troisième série d’itérations : un premier affinage du modèle**

Sur la base des plages de valeurs trouvées, j’ai commencé à affiner le modèle, pour avoir des résultats sur lesquels baser mon choix.

**4 – Choix du modèle utilisé par la suite**

Les critères de choix du modèle ont été :

* Avant tout, la performance (ROC-AUC) et la stabilité du modèle,
* Dans un second temps, les temps de calcul nécessaires.

Le tableau suivant résume les valeurs trouvées :



J’ai finalement sélectionné le LightGBM, modèle qui me retournait les meilleurs résultats, dans un temps de calcul acceptable.

**5 – Ajustement et fit du modèle**

Je continue d’appliquer la méthode expliquée précédemment sur le modèle choisi, en réduisant progressivement le maillage des paramètres proposé au GridSearchCV.

Je m’arrête quand le gain semble nul. Le ROC AUC obtenu est de 0,781 (moyenne sur les 5 plis).

Je fitte un modèle avec les valeurs optimales identifiées.

Je teste le modèle fitté sur la partition test, pour vérifier qu’il n’y a pas d’overfitting (ROC AUC obtenu : 0.784. C’est donc le cas : le score retourné est même un peu meilleur que la dernière moyenne retournée par GridSearchCV).

J’ai donc construit mon modèle.

**IV – Choix du seuil de décision**

J’ai désormais un modèle qui, grâce à la fonction predict\_proba, m’indique pour un individu donné une probabilité que l’individu fasse défaut, donnée entre 0 et 1.

La fonction predict me permettrait de retourner directement un pronostic de défaut de l’individu, de manière assez sommaire : on prédit un défaut si la probabilité de défaut est > 0,5, on prédit un remboursement si elle est <0,5. 0,5 est ici le seuil de décision.

Si ce projet était réel, on se retrouverait ici à une étape du projet où le besoin d’une expertise métier deviendrait crucial : où doit-on placer le seuil de décision ?

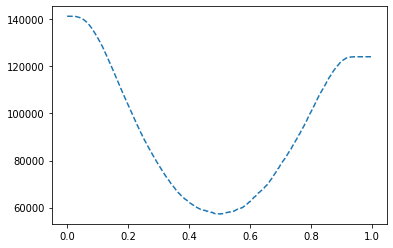
Faut de cette expertise, j’ai construit un modèle de coût hypothétique :

* Je ne prends en compte que les coûts des erreurs de prédiction.
* Les seuls coûts à prendre en compte sont :
  + le coût du défaut du client si le crédit est accordé,
  + le manque de revenus (intérêts) si le crédit est refusé alors que le client allait correctement rembourser.
* Dans le premier cas, on perd tout ou partie du capital + intérêts, dans le 2e cas on perd les intérêts. Je prends donc comme coûts de référence 10 pour un défaut, 1 pour un crédit manqué.

Pour chaque valeur de seuil envisagée, et après avoir appliqué un predict\_proba à ma partition train, je peux dire, pour chaque individu, si le crédit aurait été accordé selon ce seuil.

En comparant la prédiction et l’issue réelle du crédit, je peux construire une matrice de confusion. La matrice nous donne les deux données qui nous importent : combien de crédits auraient été accordés à des individus qui auraient fait défaut, et combien auraient été refusés à des individus qui auraient en fait remboursé.

Je peux donc, pour chaque valeur de seuil, calculer le coût associé, ce qui donne ce résultat (seuil en abscisse, coût en ordonnées):



Le coût minimum est obtenu pour un seuil de décision de 0,505.

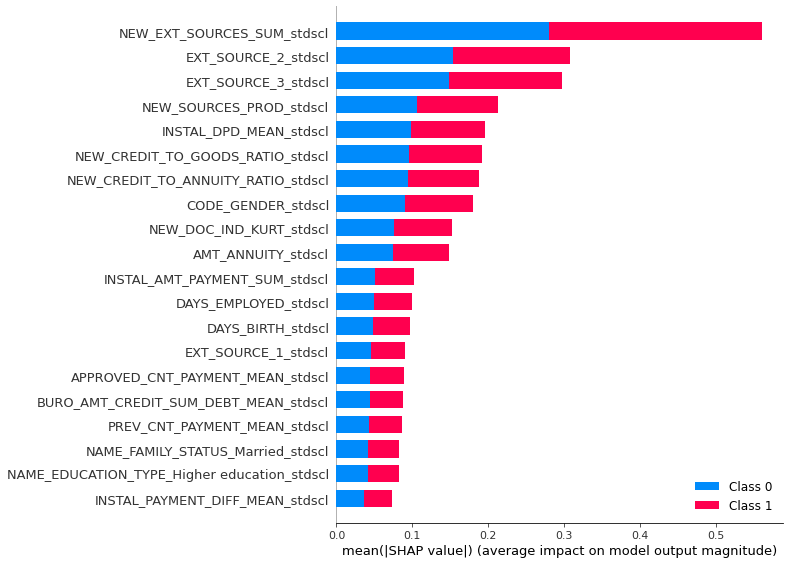
**V – Une analyse du modèle obtenu : la Feature Importance**

Le modèle obtenu est une boîte noire, fonctionnel, mais difficile à expliquer au commanditaire ou à un potentiel utilisateur. Comment fonctionne-t-il ? Pourquoi cette décision pour cet individu ?

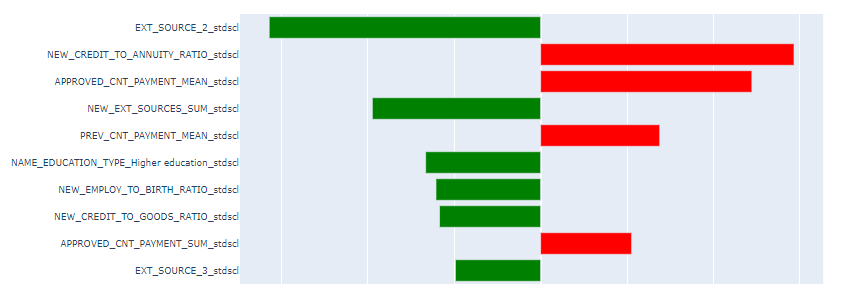
Pour expliquer le modèle, on peut utiliser un outil bien utile, SHAP, qui retourne l’importance relative de chaque variable pour la prédiction finale.

En applicant SHAP à un ensemble d’individu et à un modèle de prédiction, on obtient une matrice de dimension nombre de classes x nombre d’individus x nombre de variables. Chacune des valeurs de la matrice est la contribution de la variable à la différence entre la valeur prédite par le modèle pour l’individu et la moyenne des prédictions de tous les individus.

Pour mon modèle, les variables les plus importantes en moyenne sont les suivantes :



On peut extraire les variables les plus importantes pour un individu donné, qui peuvent être très différentes, comme pour cet individu :



**Conclusion :**

J’ai obtenu déployer un outil fonctionnel, qui semble retourner des résultats acceptables. Mais mon outil a ses limites, liées aussi bien à mon manque d’expérience qu’au manque de support de quelqu’un qui connaitrait bien les enjeux métier.