**Projet 7 - Implémentez un modèle de scoring**

**Note Méthodologique**

**Thomas MEUNIER**

1. **Méthodologie d’entraînement du modèle**

**La méthodologie d’entraînement de modèle se déroule en 7 étapes :**

**La préparation des données, le choix du modèle, l’entraînement du modèle, l’évaluation du modèle, l’amélioration du modèle, le test final sur l’ensemble de test et la mise en production.**

1. **Préparation des données :**

**Il est important de disposer de données de qualité pour entraîner un modèle efficacement. Cela implique de s'assurer que les données sont propres, complètes et bien étiquetées. Les données doivent également être divisées en ensembles d'entraînement, de validation et de test pour évaluer la performance du modèle.**

**Un kernel Kaggle a été sélectionner pour faciliter la préparation des données.**

1. **Choix du modèle :**

**Il existe de nombreux modèles de traitement disponibles.**

**Il est important de choisir un modèle qui convient à la tâche spécifique et aux données disponibles.**

**Dans mon cas, j’ai choisi 6 modèles (+populaire) à tester :**

**Dummy Classifier, Logistic Regression, Random Forest, Multi-layer Perceptron, XGBoost et LGBM.**

1. **Entraînement du modèle :**

**Il est recommandé de commencer par entraîner le modèle sur un sous-ensemble des données d'entraînement pour s'assurer qu'il converge correctement.**

**On prend donc nos données du « Train » pour faire nos tests**

1. **Évaluation du modèle :**

**Il est important d'évaluer la performance du modèle sur l'ensemble de validation pour s'assurer qu'il généralise bien aux données inconnues.**

**Au début nous n’avons pas des résultats satisfaisants.**

1. **Amélioration du modèle :**

**Si la performance du modèle sur l'ensemble de validation est insatisfaisante, il est possible de l'améliorer en utilisant des techniques telles que la régularisation et la réduction de dimension. Etant donné j’ai préféré utiliser le resampling (plus particulièrement l’oversampling) pour améliorer les performances.**

**On évalue nos modèles avec la courbe ROC, on obtient donc nos meilleurs résultats pour 3 modèles (RandomForest, XGBoost et LBGM) sur les données oversamplé.**

**J’aurai aimé optimiser les hyperparamètres du modèle mais notre base de données étant trop conséquentes je n’ai pu l’optimiser via GridSearchCV ou RandomizedSearchCV …**

**Il est également important de choisir une fonction de coût appropriée (Information en partie II)**

**Pour choisir parmi nos 3 modèles j’ai utilisé fbeta (fonction coût), la précision et le recall.**

**Nous avions tout de même de bon résultat avec les paramètres de base nous décidons donc de conserver le modèle XGBoost pour la suite du projet ayant les meilleurs résultats pour notre fonction coût métier.**

1. **Test final sur l'ensemble de test :**

**Une fois satisfait de la performance du modèle, il est important de le tester sur l'ensemble de test pour obtenir une estimation de sa performance réelle.**

1. **Mise en production :**

**Une fois que le modèle a été entraîné et évalué avec succès, il est prêt à être déployé en production pour être utilisé dans des applications réelles.**

**Dans notre cas, le modèle nous servira pour la conception d’un dashboard interactif.**

1. **La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation**

**On a 4 possibilités dans notre prédiction :**

* **Faux négatif = prêt non remboursé (accordé)**
* **Vrai négatif = prêt remboursé (accordé)**
* **Faux positif = remboursement possible (non accordé)**
* **Vrai positif = remboursement impossible (non accordé)**

**Dans notre cas métier, on accord 10 fois plus d’importance à nos clients où le prêt n’est pas remboursé car c’est ce qui fait perdre le plus d’argent à l’entreprise.**

**On défini donc les poids suivants : (f=false, t=true, n=negatif, p=positif)**

* **fn\_value = -10**
* **tp\_value = 0**
* **tn\_value = 1**
* **fp\_value = 0**

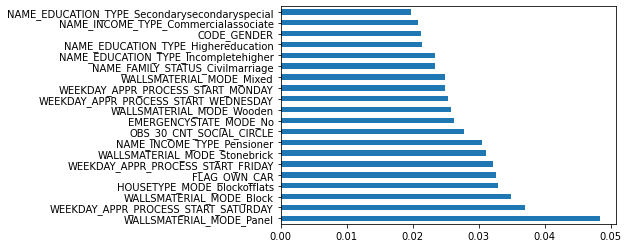
**Les prêts accordés aux individus qui ne sont finalement pas solvables sont dotés d'une pénalisation négative de -10, alors que les prêts accordés aux individus finalement solvables rapportent 1.**

**Les prêts non accordés ne nous importent pas donc ils rapportent 0.**

1. **L’interprétabilité globale et locale du modèle**

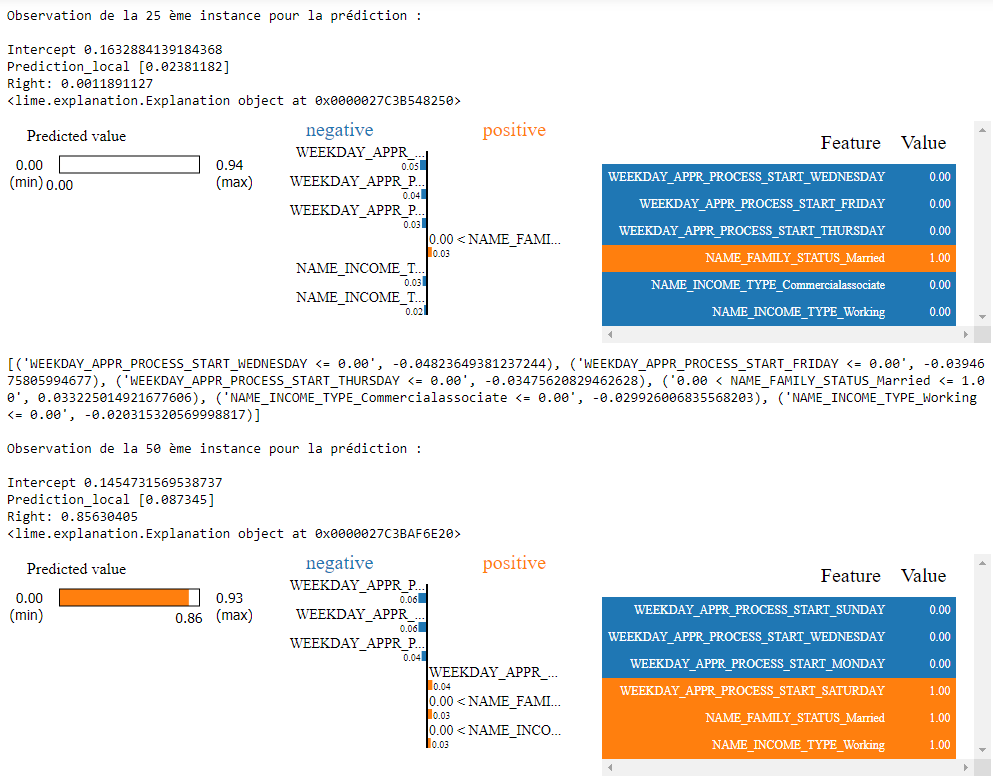
**Pour interpréter globalement et localement on regarde nos features importances.**

**Interprétabilité Global :**



**On voit sur le graphique les 20 caractéristiques ayant le plus d’impact sur la faite qu’un client arrivera ou non à rembourser un prêt.**

**Interprétabilité Local :**



**On voit sur le graphique les 6 caractéristiques ayant le plus d’impact sur la faite qu’un client arrivera ou non à rembourser un prêt.**

**Par exemple ici on peut voir que notre client à une prédiction positive (à 86%) car il travail et il est marié**

1. **Les limites et les améliorations possibles**

**La partie de data engineering est issu d’un kernel Kaggle, il est surement possible d’améliorer la modélisation via d’autres features et d’en crée de nouvelles avec l’aide des équipes métier.**

**La base de données étant trop conséquente, je n’ai pu réaliser l’optimisation des modèles. Il faudrait voir avec les équipes métiers si une sélection de features serai possible afin de la réduire la taille ou bien faire des optimisations sur plusieurs sous échantillons, mais dans ce cas l’optimisation n’est pas globale. De plus nos données ne sont pas équilibrées ce qui rajoute encore plus de données en utilisant un oversampling afin de ne pas avoir à perdre de l’information sur l’une de nos catégories pour améliorer la performance du modèle.**

**Notre modèle à été effectué avec une hypothèse sur la fonction coût métier, on pourrait donc la définir plus précisément avec l’aide des équipes métier.**

**Features importances difficilement interprétable avec des variables issue d’un encodage**

**Dashboard utiliser avec qu’une très petite partie des données dû à la mémoire et au temps de traitement.**