Note méthodologique

Le « Machine Learning » est l’art de construire des systèmes pouvant apprendre à partir de données. Apprendre signifie s’améliorer sur certaines tâches, compte tenu d’une mesure de performance. Cela consiste en pratique à construire une fonction mathématique qui approxime au mieux les données d’entrainement, dans l’objectif d’obtenir les meilleurs résultats possibles sur des données non fournis au modèle. Dans ce cas-ci, les entrées de la fonction sont les informations des clients, tandis que leur solvabilité ou non représente la sortie du modèle.

Un projet de Machine Learning comprend les étapes suivantes :

* Prétraitement des données
* Sélection d’un modèle approprié
* Choix d’une fonction coût et d’une métrique d’évaluation
* Interprétabilité du modèle
* Limites et améliorations du modèle

1. **Pré-traitement des données**

Le prétraitement consiste à nettoyer le jeu de données, en supprimant les valeurs aberrantes et en traitant les valeurs manquantes. Puis il faut ensuite numériser les variables catégorielles de manière à rendre interprétables les variables non numériques. Ce travail s’accompagne d’une étape de « Feature Engineering » qui consiste à partir des variables présentes, de trouver des combinaisons de variables plus pertinentes.

*1.1) Valeurs rares ou aberrantes pour le jeu test et entrainement*

Dans un premier temps, les jeux de données d’entrainement et de test ont été nettoyés.

Pour certaines variables de type catégorielles (non numériques), des valeurs apparaissent uniquement et en faible nombre dans le jeu de données d’entrainement. Les emprunts correspondant peuvent être supprimés.

Pour les variables numériques, les valeurs aberrantes ont été supprimés.

*1.2) Feature Engineering*

Quelques variables additionnelles qui semblent importantes peuvent être construites manuellement comme la proportion du temps travaillé par rapport à l’âge du client, ou le ratio crédit/revenu. A voir avec les équipes métier.

*1.3) Données manquantes*

Certaines variables ne possèdent pas de valeurs pour certains emprunts. Or, tout modèle de ML nécessite d’avoir comme entrée des valeurs numériques pour chaque variable qui lui est fourni. Il existe diverses méthodes qui permettent de remplacer les valeurs manquantes. Pour chaque variable, les valeurs manquantes ont été remplacées par la valeur moyenne de la variable en question.

*1.4) Encodage des variables catégorielles*

Pour les variables catégorielles, deux méthodes peuvent être utilisées :

* Pour les variables dont le nombre de valeurs différentes est égal à deux, elles sont à binariser, i.e. coder par des 0 ou des 1.
* Pour les autres, une nouvelle variable est créée pour chaque valeur différente prise par la variable initiale. Et pour chaque emprunt, la valeur prise est soit 1 ou 0 suivant la valeur de la variable initiale.

1. **Selection d’un modèle ML**

Il existe différents modèles de ML plus ou moins performants et plus ou moins complexes, donc plus ou moins interprétables. Parmi ceux communément utilisés, nous avons sélectionnés les suivants : Régression Logistique, Forêts aléatoires, K Neighbors Classifier, Gradient Boosting Classifier, Linear SVC.

Nous les avons tous testés en cherchant à déterminer la meilleure combinaison d’hyperparamètres.

Démarche de modélisation

La première étape a consisté à gonfler artificiellement le jeu de données pour obtenir une plus grande proportion de personnes non solvables. En effet, environ 90% des prêts existants dans le jeu de données d’entrainement ont été remboursés. Nous avons donc utilisé la librairie SMOTE qui nous a permis de trouver un équilibre à 50/50%.

Comme expliqué précédemment, chaque modèle de Machine Learning présente différents paramètres à fixer. Suivant la combinaison des paramètres choisis pour le modèle, les résultats seront plus ou moins performants.

Pour déterminer quelle combinaison d’hyperparamètres donne les meilleurs résultats sur notre jeu de données, l’algorithme d’optimisation « gridsearch » a été utilisé.

Couplé à l’algorithme d’optimisation « gridsearch », un système de validation croisée a été mis en place. L’échantillon original (données d’entrainement) est divisé en k échantillons, puis on sélectionne un des k échantillons comme ensemble de validation et les k-1 autres échantillons constitueront l’ensemble d’apprentissage. On calcule comme dans la première méthode le score de performance, puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les é{\displaystyle k-1}éééchantillons qui n'ont pas encore été utilisés pour la validation du modèle. L'opération se répète ainsi {\displaystyle k} k fois pour qu'en fin de compte chaque sous-échantillon ait été utilisé exactement une fois comme ensemble de validation. Cette technique permet d’éviter ce qu’on appelle le sur-apprentissage.

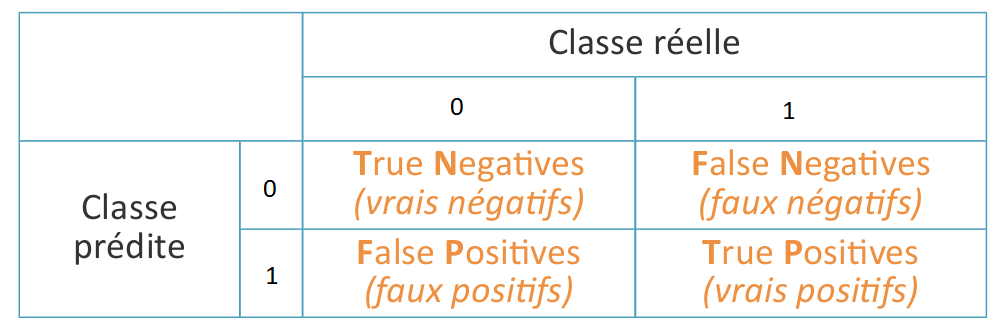
**3) Fonction coût : métrique bancaire**

Un modèle est dit meilleur qu’un autre lorsqu’il donne des meilleurs résultats suivant une métrique de performance bien définie. Pour ce projet, une fonction coût bien précise a été développée dans l’objectif de maximiser les gains obtenus par validation ou non des demandes de prêts bancaires. La fonction coût étant à maximiser pour ce problème.

La procédure est la suivante :

Il s’agit d’un problème de classification binaire, l’étiquette est soit égale à 0 (négatif, solvable) soit égale à 1 (positif, non solvable). Il existe donc 4 combinaisons possibles :

* TN : True Negatif, le modèle prédit 0 et la valeur réelle est 0
* TP : True Positif, le modèle prédit 1 et la valeur réelle est bien de 1
* FN : False Negatif, le modèle prédit 0 alors que la valeur réelle est bien de 0
* FP : False Positif, le modèle prédit 1 alors que la valeur réelle est de 0



Le modèle peut donc se tromper de deux manières différentes, soit en prédisant positif alors que l’individu est négatif (False Positif) soit en prédisant négatif alors que l’individu est positif. En revanche, la perte d’argent est plus conséquente pour un FN (prêt accordé alors que le client n’est pas solvable) qu’un manque à gagner pour un FP (prêt non accordé alors que le client est solvable). Une fonction coût a donc été créée pour tenir compte de l’importance relative de chaque erreur.

J = TP\*TP\_Value + TN\*TN\_value + FP\*FP\_Value + FN\*FN\_Value

TP, TN, FP et FN étant respectivement le nombre de True Postif, le nombre de True Negatif, le nombre de False Positif et le nombre de False Negatif.

Des coefficients ont été posés arbitrairement :

* TP\_value : 0
* FN\_value : -10
* TN\_value : 1
* FP\_value : 0

Ces valeurs de coefficients signifient que les Faux Négatifs engendrent des pertes 10 fois plus importantes que les gains des Vrai Négatifs. Ces poids ont été fixé arbitrairement et il est tout à fait envisageable de les modifier à la convenance de l’optique métier.

1. **Interprétabilité du modèle**

L’interprétabilité d’un modèle est d’autant plus importante que le modèle est complexe. Le modèle sélectionné étant relativement compliqué, la librairie LIME a donc été utilisée.

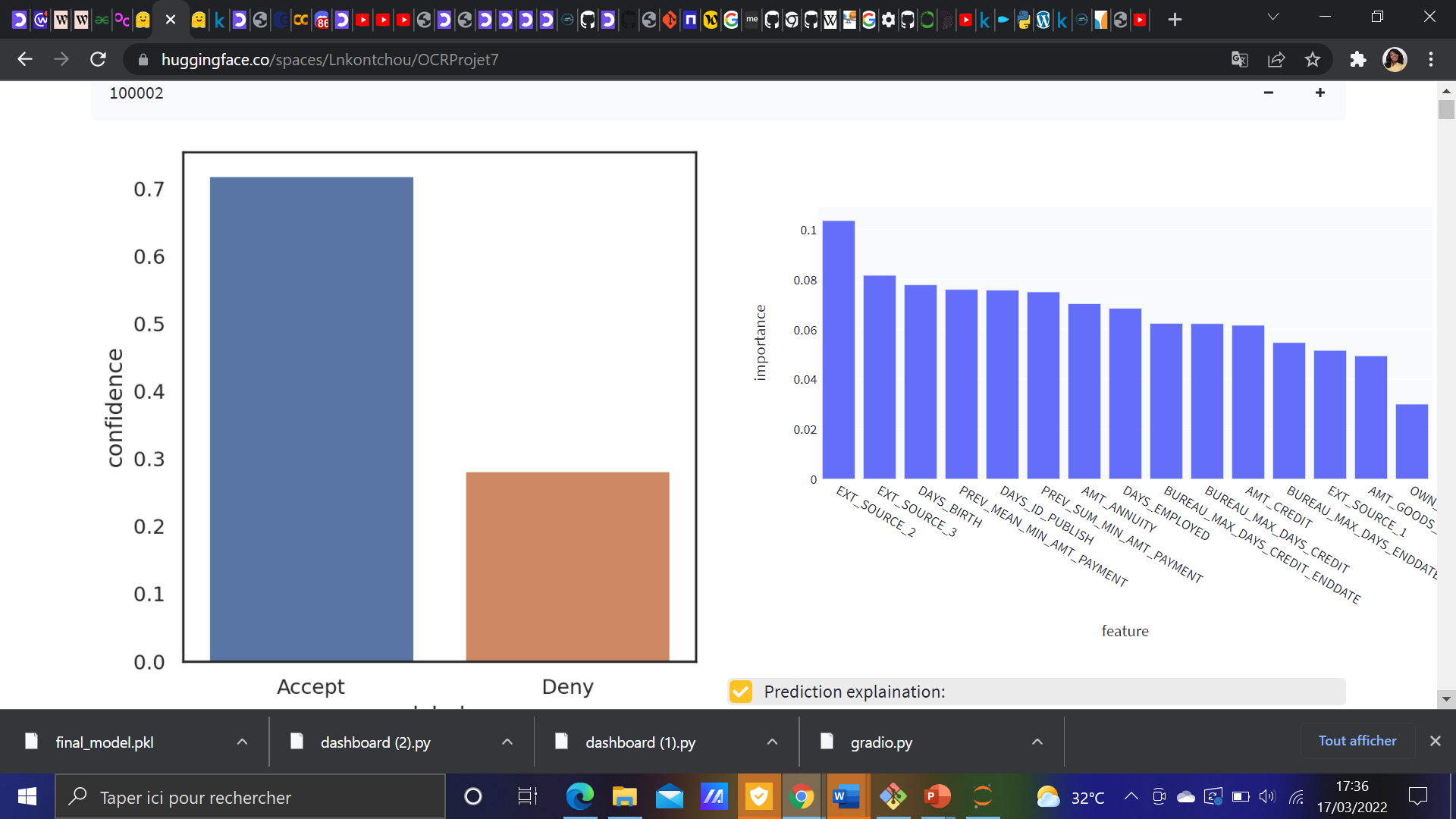
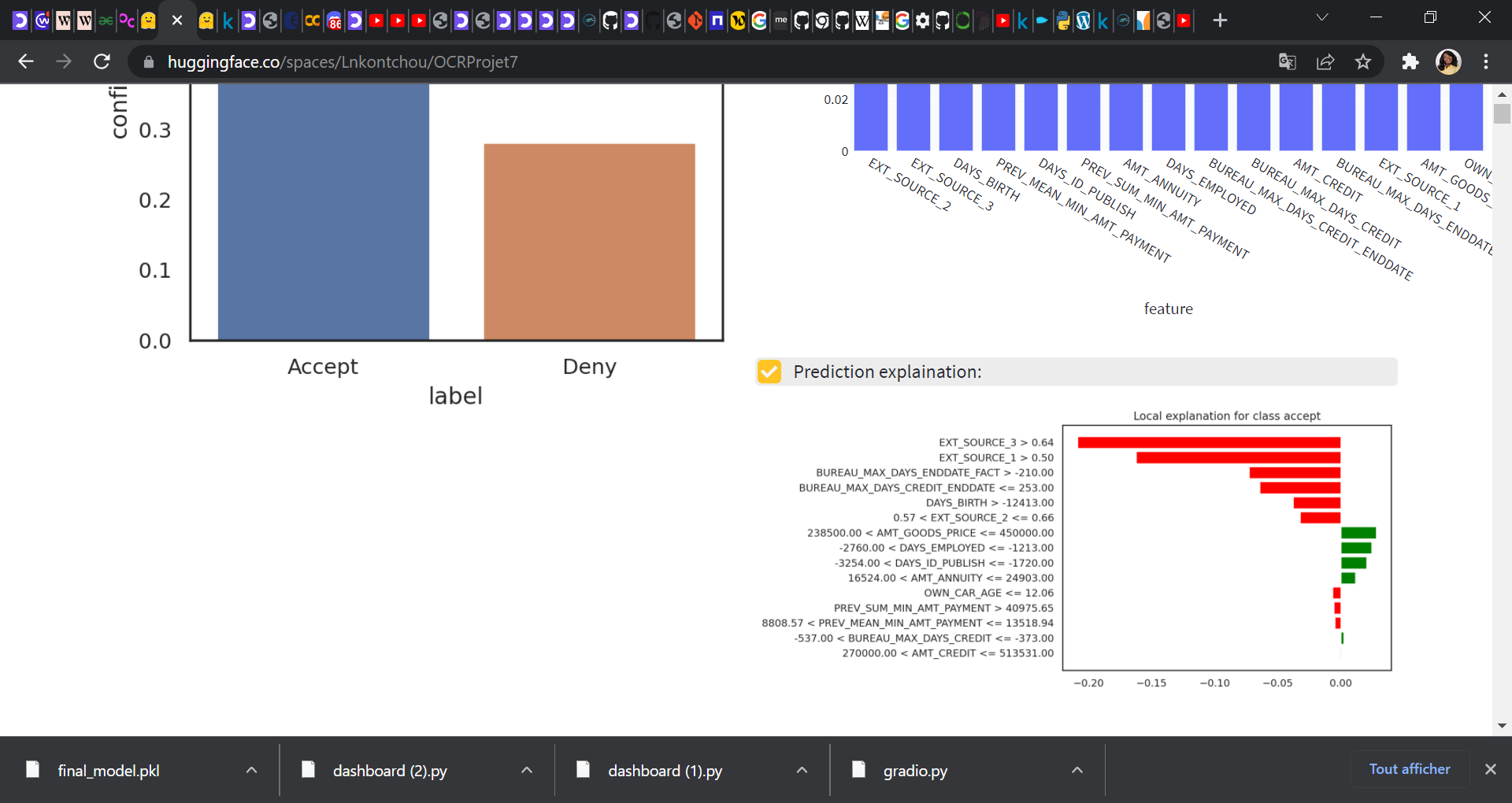
4.1) Principe de fonctionnement

LIME permet de déterminer pour chaque observation les  « features » qui ont le plus d’impact dans le résultat de la prédiction.

Lime fonctionne en 3 étapes :

* Lime génère aléatoirement des nouveaux individus fictifs proches de l’individu sélectionné
* Lime calcule la prédiction en fonction du modèle
* Lime calcule un modèle linéaire (interprétable) sur ces nouvelles données

4.2) Exemple pour un client donné

Prédictions

La solvabilité du client '100002' est supérieure au seuil de solvabilité de 0.5, donc le client est solvable. La variable qui joue le plus en sa défaveur est 'EXT\_SOURCE\_3' qui est un score normalisé provenant d'autres données. A l’inverse la variable qui joue le plus en sa faveur est ‘DAYS\_EMPLOYED'.

1. **Limites et améliorations du modèle**

5.1) Cross validation

Il pourrait être envisageable de diviser le jeu de données en n partitions puis d’entrainer un modèle sur chacune, puis d’utiliser un ‘blender’ sur les n modèles obtenus. Un ‘blender’ consiste à agréger les prédictions de chacun des classificateurs et de prédire la classe qui obtient le plus grand nombre de votes. Ce classificateur par vote majoritaire obtient souvent une exactitude plus élevée que le meilleur classificateur de l’ensemble.

5.2) Sélection des variables et ‘feature engineering’

Il serait judicieux d’effectuer une sélection des variables à partir de « Backward » ou de « Rfecv (Recursive Feature Elimination with Cross Validation) » de « scikit-learn ».

Quant au « Feature Engineering », il serait judicieux de discuter avec les équipes métiers du domaine. Cela permettrait de déterminer quelles compositions de variables sont les plus pertinentes au vu de leur expérience dans leur choix d’accorder un prêt ou non à un client donné.

5. 3) Interprétabilité du modèle

La méthode SHAP qui repose entre autres sur la théorie des jeux compense les défauts de la méthode LIME comme son instabilité et le fait qu’un modèle local simple n’est pas une bonne approximation du modèle global.