NOTE METHODOLOGIQUE

Projet 7 : Implémentation d'un modèle de scoring

Formation Diplômant de OpenClassrooms

Parcours : Data Science

Kokou AGBOTO

Mars 2022

Objectifs du projet

- Le Modèle doit permettre de définir la probabilité d'un client de façon automatique sur la base des informations relatives de celui-ci.
- Concernant les données et leurs traitements, le modèle doit offrir un certain niveau de transparence en vue d'implémenter des méthodes d'interprétabilité des variables.

Données (Datasets)

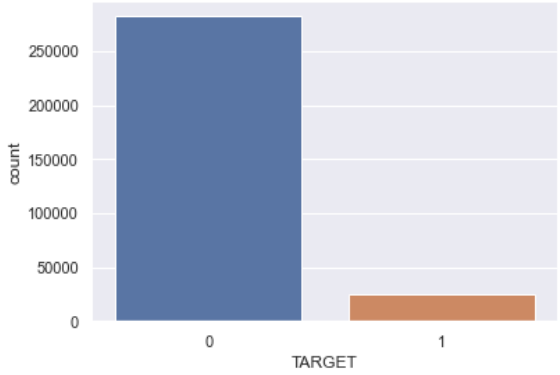
Nous avons un groupe de plusieurs jeux de données à savoir :

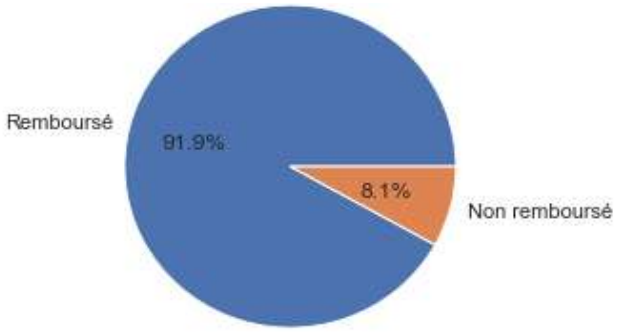
- Un jeu de données principal (**307.511** demandes de prêts en cours et **123 variables ou features**) constitué des informations relatives aux crédits en cours et autres informations externes. Ce jeu de données comprend des étiquettes (***TARGET ou CIBLE***), une valeur binaire indiquant si le crédit a été ***remboursé (0) ou non (1)***, celui-ci sera utilisé pour l'apprentissage automatique.
- Une seconde table qui ne contient pas d'étiquette (**48.744** demandes de prêts en cours et **122 variables ou features**), il s'agit des dossiers en cours en cours pour lesquels il faut réaliser un classement prédictif avec le modèle précédemment entraîné.
- Des tables qui contiennent des informations relatives aux clients, de ceux-ci nous allons extraire des informations pour enrichir les deux précédents en faisant une jointure ; afin une table métadonnées qui donne la signification de chaque variable.

Techniques pour gérer les classes déséquilibrées

Dans notre cas ici on remarque que le nombre d'échantillons dans la **classe 0** (les clients solvables) est **majoritaire** (**environ 92 %**) par rapport au nombre d'échantillons dans la **classe 1** (les clients non solvables) qui est **minoritaire** (**environ 8 %**) ; on a donc à faire à un problème de classes déséquilibrées comme l'indique les graphiques suivants.

```
0    282686
1     24825
Name: TARGET, dtype: int64
```



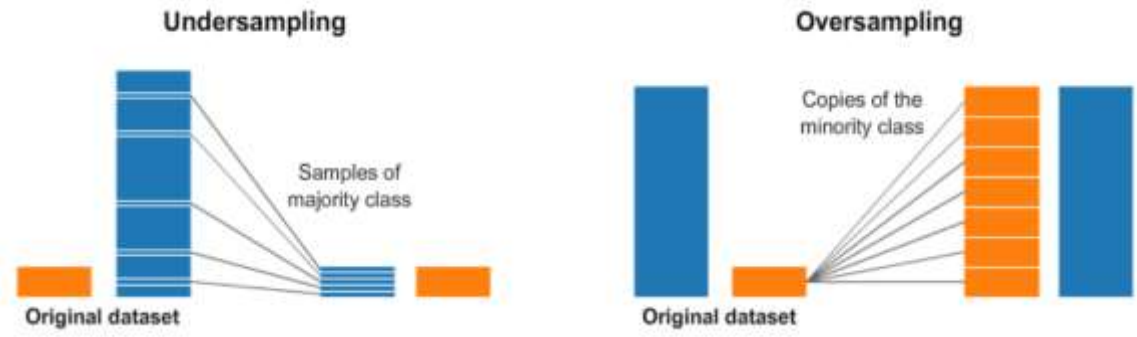


Dans des cas pareils en apprentissage automatique, le classificateur "prédira" toujours la classe la plus courante sans effectuer aucune analyse des caractéristiques et il aura un taux de précision élevé, évidemment pas le bon, pour y remédier il faut faire un rééchantillonnage.

Technique de rééchantillonnage

Il y'a plusieurs techniques pour traiter les problèmes de classes déséquilibrées ; nous allons utiliser entre autres

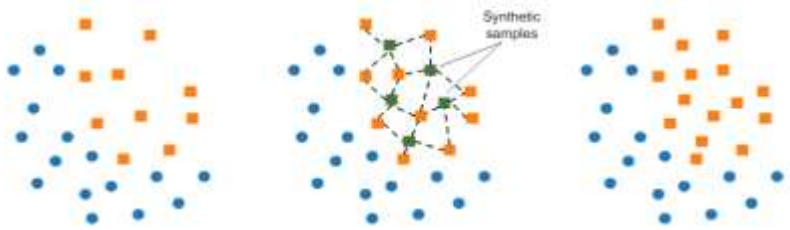
Quelques-unes :



A) SMOTE (Synthetic Minority Oversampling Technique)

Technique de suréchantillonnage synthétique minoritaire, cette technique génère des données synthétiques pour la classe minoritaire, l'algorithme SMOTE fonctionne en 4 étapes simples :

1. Choisissez une classe minoritaire comme vecteur d'entrée
2. Trouver ses k voisins les plus proches (k_neighbors est spécifié comme argument dans la fonction SMOTE ())
3. Choisissez l'un de ces voisins et placez un point synthétique n'importe où sur la ligne joignant le point considéré et son voisin choisi
4. Répétez les étapes jusqu'à ce que les données soient équilibrées



Résultats après smote

```
y_smote.value_counts()
```

```
0.0    1842
1.0    1842
Name: TARGET, dtype: int64
```

B) Pénaliser les algorithmes (formation sensible aux coûts)

La tactique suivante consiste à utiliser des algorithmes d'apprentissage pénalisés qui augmentent le coût des erreurs de **classification sur la classe minoritaire**, nous pouvons utiliser l'argument **class_weight='balanced'** pour pénaliser les erreurs de la classe minoritaire d'un montant proportionnel à sa sous-représentation.

Nous voulons également inclure l'argument **probability=True** si nous voulons activer les estimations de probabilité.

```
imbalanced_test = y_test.value_counts()[0].sum()/y_test.value_counts()[1].sum()
imbalanced_test
```

```
1.058659217877095
```

```
imbalanced_train = y_train.value_counts()[0].sum()/y_train.value_counts()[1].sum()
imbalanced_train
```

```
0.9858490566037735
```

Prétraitement des données :

Nous, nous sommes inspirés de deux kernels Kaggle comme suggéré le projet : LightGBM with Simple Features | Kaggle , Start Here: A Gentle Introduction | Kaggle et HomeCreditRisk: Extensive EDA + Baseline [0.772] | Kaggle , les principaux traitements appliqués sont :

1. Constitution des tables (**train (entraînement) et test (soumission)**) à partir des autres tables.
2. Détection des anomalies (variables : **DAYS_BIRTH, DAYS_EMPLOYED**)
3. Encodage des variables catégorielles (utilisation de **OneHotEncoder**)
4. Imputation des variables manquantes (**suppression de variables en fixant le taux de remplissage de valeurs manquantes à 80%, fillna(0)**), suppression des variables corrélées au seuil de 80 %)
5. Alignement du jeu de données d'entraînement et jeu de données de soumission.

Métriques utilisées

Il est attendu ici par l'institution financière le calcul de la probabilité de défaut de non remboursement du prêt, il faut donc fixer une valeur seuil (threshold) de probabilité. Si la probabilité calculer est inférieure à cette valeur seuil alors elle prendra la valeur 0 (le prêt sera remboursé) sinon la valeur 1 (le prêt ne sera pas remboursé); par défaut c'est à 0,5.

Il s'agit bien ici d'un problème de classification binaire (0 ou 1). Pour évaluer les performances de ces de modèles de classification il faut utiliser la matrice de confusion (En anglais confusion matrix). Développons les terminologies de ce concept.

Classes prédites

	0 (remboursé) N	1 (non remboursé) P	
(remboursé) 0 **Classe réelle N**	**TN (Vrai Négatif)** **Dossiers prédits négatifs et ils le sont réellement**	**FP (Faux Positif)** **Dossiers prédits positifs mais ne le sont pas en réalité**	
(non remboursé) 1 **Classe réelle P**	**FN (Faux Négatif)** **Dossiers prédits négatifs mais ne le sont pas en réalité**	**TP (Vrai Positif)** **Dossiers prédits positifs et il le sont réellement**	$\frac{TP}{TP+FN}$ **Recall**
		$\frac{TP}{TP+FP}$ **Precision**	

Pour l'institution financière, les dossiers prédits négatifs et qui le sont réellement (TN) et les dossiers prédits positifs et qui le sont réellement (TP) sont les clients correctement prédits pas d'incidence sur l'institution financière.

Pour les dossiers prédits positifs mais ne le sont pas en réalité (FP) et les dossiers prédits négatifs mais ne le sont pas en réalité (FN) ont respectivement pour incidence perte de clients (perte des intérêts de prêts) et perte énorme d'argent. Il faut donc se concentrer sur ces types de clients (les clients à qui l'institution a donné le prêt à tort).

Comme ici nous sommes dans une situation de classification de classes déséquilibrées (dans notre cas ici à 2 classes déséquilibrées), nous allons utiliser la métrique $F_{\beta-score}$ (Défini comme étant la moyenne harmonique de la precision et du recall) comme l'indique la formule suivante :

$$F_{\beta-score} = (1+\beta^2)\frac{\text{Precision} \,.\, Recall}{(\beta^2.\ \text{Precision})+Recall} \quad \textbf{ou encore} \quad F_{\beta-score} = \frac{TP}{TP+\frac{1}{(1+\beta^2)}(\beta^2.FN+FP)}$$

Résumons :

- Pour $\beta \geq 1$, on accorde **plus d'importance au recall** (FN).
- Pour $\beta \leq 1$, on accorde **plus d'importance à la precision** (FP).
- Pour $\beta = 1$, on retrouve le F1-score, qui accorde autant d'importance à la precision et au recall.

En pratique on a généralement recours à une **optimisation du recall sous contrainte de précision** ou à une **optimisation de la précision sous contrainte de recall**. Pour évaluer globalement la performance d'un modèle, on calcule l'aire sous la courbe **Precision-Recall**, nommée **AUC Precision-Recall**

Entraînement des différents modèles

Classifieurs

Pour l'évaluation des modèles nous avons utilisé AUC et le temps d'exécution pour évaluer et choisir le bon classifieur. Celui-ci doit intégrer des méthodes de classification binaire vu qu'ici nous avons à faire à la classification à deux classes.
Les classifieurs sont : GaussianNB; LogisticRegressionCV; DecisionTreeClassifier; RandomForestClassifier; LGBMClassifier; GradientBoostingClassifier; MLPClassifier.

Préparations de données

On définit tout d'abord les vecteurs X = {ensemble des features} et Y = {classe cible}. Ces données sont séparées en un jeu d'entraînement (80 % des données) et un jeu d'évaluation ou de test (20 % restants).

La fonction coût métier, l'algorithme d'optimisation

Optimisation technique

Pour augmenter les performances de notre modèle, nous avons réduit le nombre de variables.

- Le modèle retenu est entraîné sur le jeu réduit et affiche des résutats: **AUC = 88.90 %**
- Nous avons ensuite optimisé les paramètres du classifieur avec la méthode **Hyperopt** et obtenu une mesure de **AUC = 99.52 %**, qui passe par quatre étapes à savoir :
  1. Définir une fonction objective **à minimiser**. En général, il s'agit de la perte ou de la validation d'entraînement.
  2. Définir **l'espace de recherche des hyperparamètres**. Hyperopt fournit un espace de recherche conditionnel, qui vous permet de comparer différents algorithmes ML dans la même exécution.
  3. **Spécifier l'algorithme de recherche**. Hyperopt utilise des algorithmes de réglage stochastiques qui effectuent une recherche plus efficace de l'espace hyperparamétrique qu'une recherche de grille déterministe.
  4. Exécuter la fonction **Hyperopt fmin()**. fmin() prend les éléments que vous avez définis dans les étapes précédentes et identifie le jeu d'hyperparamètres qui minimise la fonction objective.

La fonction à minimiser s'appelle hyperparamter_tuning et l'algorithme de classification pour optimiser son hyperparamètre est le **LigthGBM**, nous allons utiliser la validation croisée pour éviter le surajustement, puis la fonction renverra une valeur de perte(loss).

Le principe "bayésien" consiste alors à sélectionner les prochaines valeurs à tester en se basant sur un critère d'optimisation de la fonction de substitution. Cela permet de limiter les évaluations de la fonction objective (gain de temps)

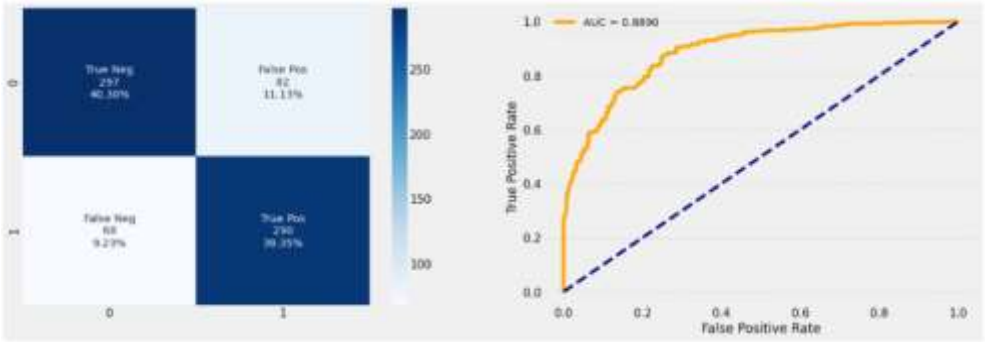
Contrairement aux méthodes aléatoires ou par grille, à chaque nouvelle itération, le raisonnement bayésien prend en compte les résultats de l'itération précédente et la fonction de substitution se rapproche de la fonction objective.

L'avantage de cette approche est qu'elle nous permet de définir la fonction objective selon nos besoins. D'un point de vue purement technique, on va ainsi pouvoir demander à la fonction objective de maximiser la mesure AUC (ou autre mesure comme le Fbeta-Score).

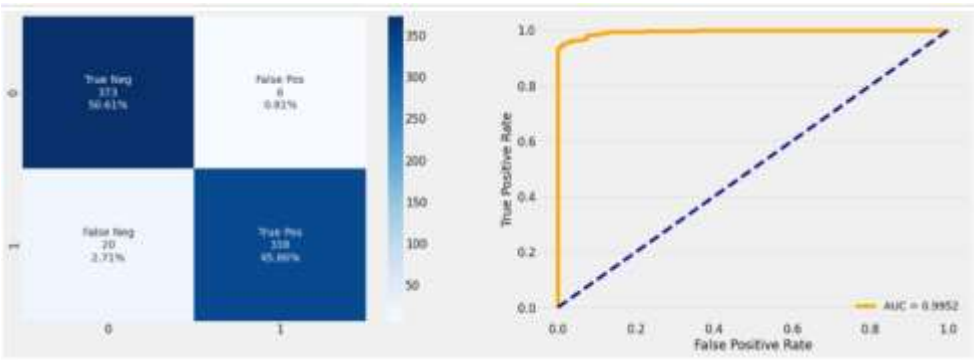
Optimisation métier

Dans la majorité des cas pour mieux mesurer la performance d'un modèle et ne pas se tromper (surtout dans un cas de déséquilibre des classes) alors il faut utiliser la métrique AUC Precision-Recall $F_{\beta-score}$, ainsi comme nous l'avons indiqué plus haut, pour améliorer la performance de notre modèle il faut pénaliser FN (dossier prédit négatif mais ne le sont pas en réalité) au détriment de FP (Dossier prédit positif mais ne le sont pas en réalité); c'est cela qui sera profitable à l'institution financière ; il faut donc choisir une valeur de β Supérieur à 1 que nous allons introduire dans la fonction **custom-score** qui pénalise (FN) et lui à son tour sera introduit dans le modèle en prenant pour paramètre du modèle la fonction d'évaluation (c'est ce qui maximisera le Gain de la société financière ainsi la valeur de AUC passe à ….....

<u>Affichages des matrices de confusion et la courbe de ROC (avec AUC)</u>




```
100%|| 30/30 [01:57<00:00,  3.91s/trial, best loss: 0.5270220248010364]
Wall time: 1min 57s
```




```
100%|| 30/30 [02:18<00:00,  4.61s/trial, best loss: 0.010548111812780592]
Wall time: 2min 18s
```

Commentaire

Le package hyperopt prend 1min 57s et 2min 18s, en termes de temps d'exécution de 30 modèles pour chaque, le premier modèle est moins couteux en temps mais quand on regarde le loss (la meilleure perte est la seconde qui est de 0.011 autrement dit la meilleure précision est 1-0,011 = **0,989** de précision ; on peut aussi voir l'air sous la courbe c'est-à-dire la probabilité d'acceptation de prêt qui est **98,9%.**

L'interprétabilité globale et locale du modèle

Dans la partie interprétabilité, on va tenter d'expliquer les prédictions, c'est-à-dire de comprendre comment se fait l'attribution des classes. Ces informations permettront à la société financière de justifier sa décision d'octroyer ou non de crédit à ses clients, en toute transparence.

Importance des variables

Interprétabilité globale avec la Librairie SHAP (**SH**apley **A**dditive ex**P**lanation)

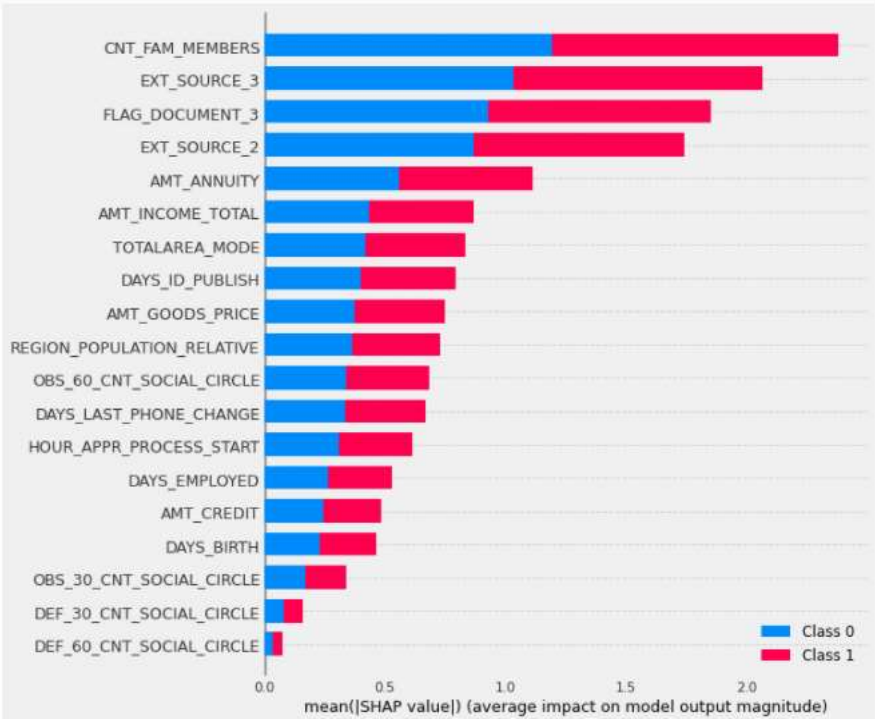
L'idée proposée par ces auteurs est de calculer la valeur de Shapley pour toutes les variables à chaque exemple du dataset. Cette approche explique la sortie d'un modèle par la somme des effets de chaque variable. Ils se basent sur la valeur de Shapley qui provient de la théorie des jeux. L'idée est de moyenner l'impact qu'une variable a pour toutes les combinaisons de variables possibles.

La valeur de Shapley d'une feature est alors la moyenne de la contribution de sa valeur à travers les différentes combinaisons.

Exemple : Supposons que pour une instance donnée du dataset iris, notre modèle prédit la classe « versicolour » avec un score 60% lorsque la largeur des pétales est égale à 1.3cm, leur longueur est égale à 4.5cm, etc. Si en ne changeant que la valeur de la largeur des pétales, en remplaçant 1.3 par 2cm, le score baisse de 10%, alors la contribution de la valeur 1.3cm était de 10%.

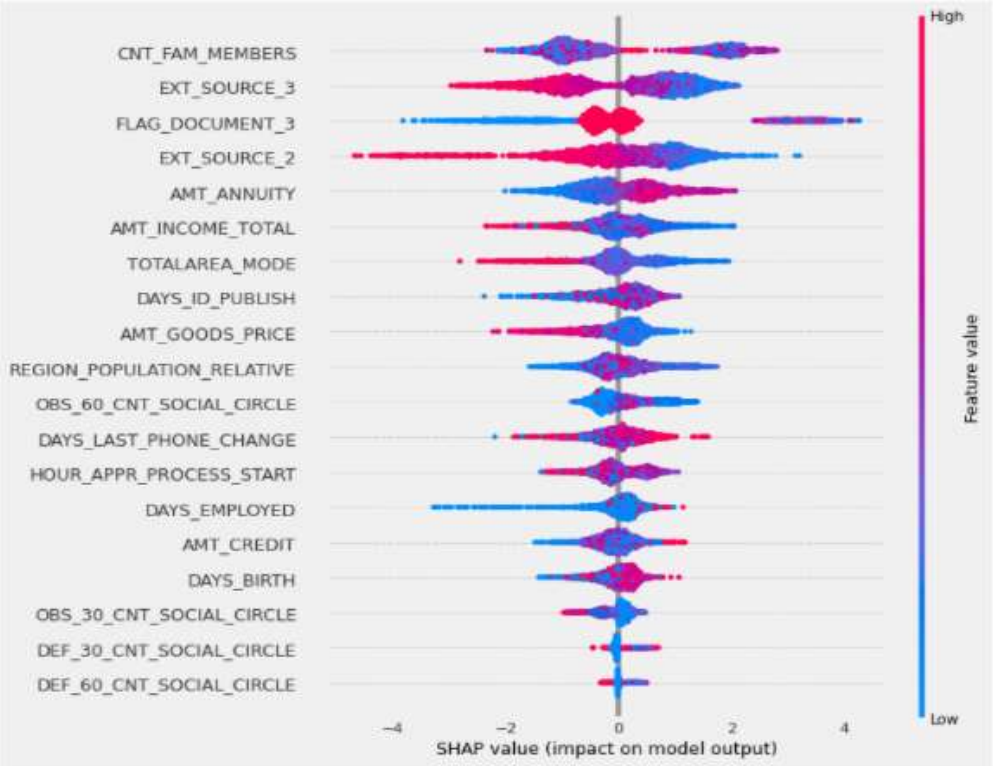
- ❑ Parlant de l'interprétabilité globale (figure ci-dessous), nous pouvons dire que ces variables nous renseignent sur l'influence que chacune d'elle apporte dans la décision de notre classification.

Sur le graphique suivant sont représentées quelques features les plus importantes pour le modèle **LigthGBM** sélectionné.



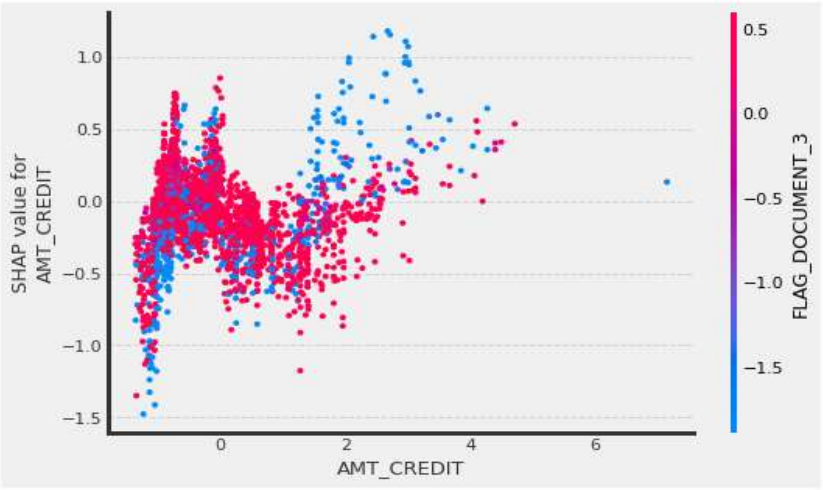
- Par exemple, les valeurs élevées de la variable **EXT_SOURCE_2** ont une contribution négative très élevée(**> - 4**) sur la prédiction, tandis que les valeurs faibles ont une contribution positive élevée (**2**).
- La variable **AMT_ANNUITY** a une contribution positive (**un peu plus de 2**) lorsque ses valeurs sont élevées et une contribution négative pour ses valeurs faibles (**un peu plus de -2**).

Toutes les variables sont affichées dans l'ordre **d'importance globale** des caractéristiques, la première étant la plus importante et la dernière étant la moins importante.



Remarques

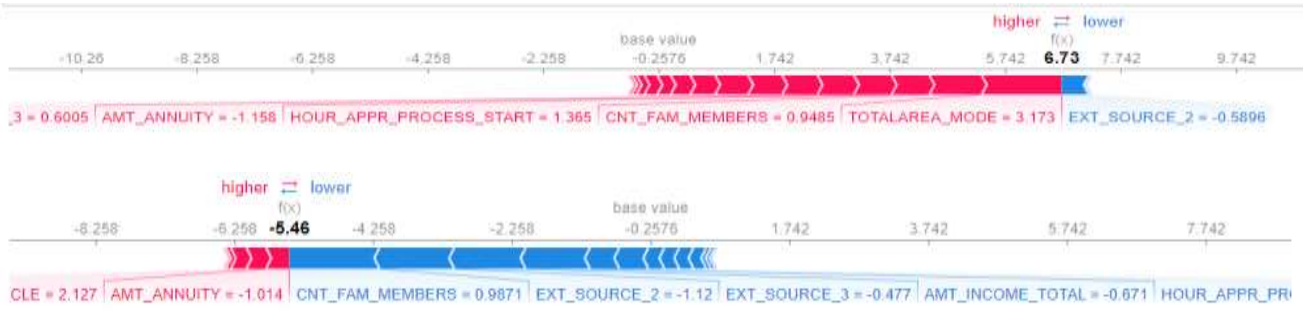
Le **diagramme de dépendance** (dependence_plot) partielle montre l'effet marginal qu'une ou deux variables ont sur le résultat prédit d'un modèle d'apprentissage automatique (JH Friedman 2001). Il indique si la relation entre la cible et une entité est linéaire, monotone ou plus complexe. Pour créer un diagramme de dépendance, vous n'avez besoin que d'une seule ligne de code : shap.dependence_plot("variable", shap_values, data). La fonction inclut automatiquement une autre variable avec laquelle la variable choisie interagit le plus



Chaque point correspond à une donnée ; son abscisse correspond à la valeur de la variable choisie (**Montant du crédit** ici) et son ordonnée à la **valeur SHAP de la variable**. La relation entre Montant du crédit et les données de SOURCE_EXT_2 est un peu complexe à expliquer.

Interprétabilité locale avec la Librairie SHAP (**SH**apley **A**dditive ex**P**lanation)

La valeur de base (**base value**) est la valeur moyenne obtenue comme sortie pour cette classe, alors que la valeur de sortie (**model output value** ici f(x)) est la valeur prédite par le modèle. Les valeurs SHAP de chaque variable, proportionnelles aux tailles des flèches, « poussent » la prédiction depuis la valeur de base jusqu'à la valeur prédite



Limites et améliorations possibles

Dans cette note méthodologique, nous avons présenté les démarches en vue de la modélisation du calcul de défaut de prêt pour une institution financière ; dans notre démarche, il y'a eu beaucoup de transformations des données de base : Constitutions des tables en faisant des jointures, traitement des valeurs manquantes, corrélations entre les variables, seuil de variance, réduction de dimensions ; fonction d'évaluation du gain ; optimisation des hyperparamètres, toutes ces transformations ne sont pas à leurs utilisations optimales et même le choix des modèles testés.

En effet :

- Features engineering qui nécessite de la **connaissance** du domaine des données pour créer des features qui font fonctionner les algorithmes d'apprentissage automatique au mieux, ceci suppose donc qu'il faille discuter avec les personnes du domaine financier pour pouvoir valider cela, ceci aiderait plus à améliorer notre modèle.
- Pour la fonction d'évaluation du gain ; il faut aussi faire valider les choix des coefficients multiplicateurs pour différentes métriques de la matrice de confusion selon les besoins de la société ; en utilisant le fbeta-score en faisant varier beta permet d'optimiser le seuil à partir duquel la probabilité se transforme en 1, mais ne prend pas en compte le contexte « métier »
- Pour l'optimisation des hyperparamètres en utilisant l'algorithme hyperopt peut être limité puisqu'il contient plusieurs paramètres ; et il faut une connaissance approfondie pour en savoir les plus performants à optimiser ; on dispose aussi de certains algorithmes permettant d'optimiser les hyperparamètres d'un modèle comme () qu'il faut aussi explorer et faire un choix.
- Nous avons entraîné et testé sept différents modèles ; nous aurions puis tester plusieurs modèles si on avait du temps et choisir celui le plus adapté (SVM, Réseaux de neurone, k-plus proches voisins, Classification naïve de Bayes, etc.…)
- Nous avons travaillé sur un jeu de données réduit, nous aurions pu travailler sur un jeu de données plus riche et plus grand si les moyens informatiques le permettaient (un serveur performant par exemple).