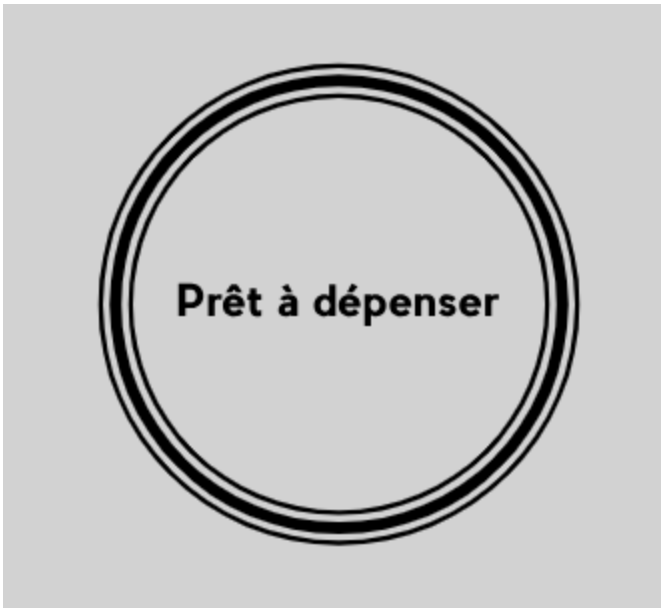
Note Méthodologique

Implémentez un modèle de scoring

Openclassroom - parcours Data Scientist - projet 7



CONTEXTE

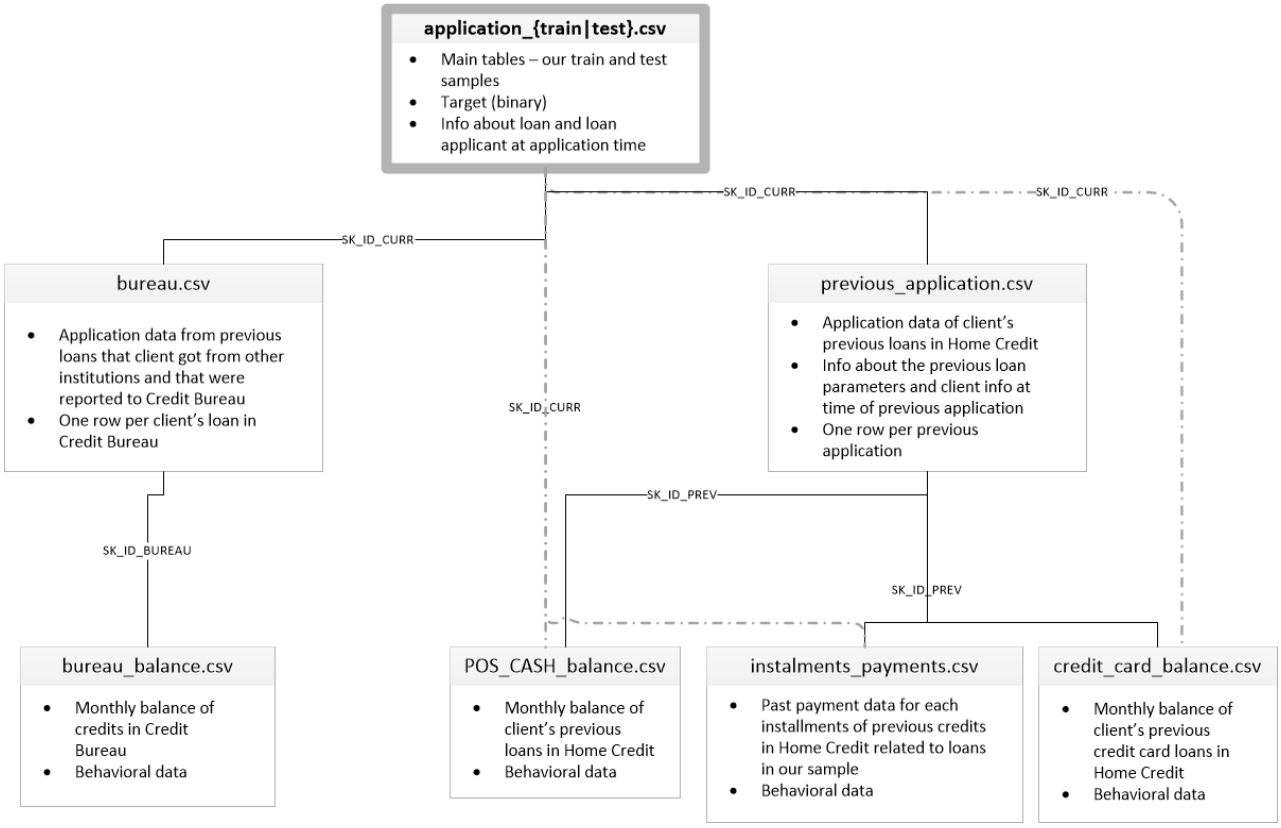
L'entreprise "Prêt à dépenser" souhaite développer un modèle de scoring crédit pour calculer la probabilité qu'un client rembourse son crédit afin de pouvoir classer les nouvelles demandes en crédit accordé ou refusé.

Ce mémoire représente la note méthodologique du projet ayant permis la mise en place d'une première version de ce modèle de scoring. Les données utilisées sont une base de données de 307 000 clients issues d'une compétition proposée par kaggle en 2018. Les fichiers de données sont accessibles au lien suivant :

- https://www.kaggle.com/c/home-credit-default-risk/data

DONNÉES DISPONIBLES

Le kernel kaggle met à disposition 7 tables de données réparties selon le schéma de données suivant:



La société Prêt à dépenser met en avant la volonté de proposer un service fonctionnel pour des clients n'ayant pas ou peu d'historique relatif aux crédits à la consommation. C'est pourquoi cette première version du modèle de scoring fera abstraction de 6 tables contenant les informations relatives aux historiques des clients pour ne retenir que les fichiers "application_train.csv" et "application_test.csv".

Les observations disponibles dans les tables "application_{train | test}.csv" correspondent aux données relatives à une demande de crédit, rassemblant les informations personnelles d'un client au moment de sa demande. Près de 121 variables sont décrites et seront les données d'entrées du modèle. La variable "Target" représente le statut de "défaut" ou de "non-défaut" associé à un crédit, cela représente le critère de classification cible de notre modèle.

La variable "target" étant absente du jeu de données test et ces valeurs n'étant pas accessibles publiquement en raison des conditions de compétition kaggle, le jeu de données "application_test.csv" ne sera pas exploitable directement pour l'entraînement et l'évaluation d'un modèle. Ce jeu de données pourra constituer une base de "nouveaux clients" pour alimenter le prototype d'une interface de visualisation de dossier client.

MÉTHODOLOGIE D'ENTRAÎNEMENT DU MODÈLE

	application_train.csv	
split layer	train split (75%)	test split (25%)
balance layer	balanced	not balaced
preprocessing layer	fit transform	transform
model layer	fit (GridSearchCV)	predict score

Tableau récapitulatif des traitements effectués sur le jeu de données 'home credit default risk'.

Séparation du jeu d'entraînement

La première étape de notre méthode consiste à générer 2 sous-ensembles de notre jeu de données. Le sous-ensemble principal contenant 75% du jeu de données initial, identifié comme "train split", servira à l'entraînement des pipelines de transformation lors du preprocessing ainsi que l'entraînement et l'optimisation paramétriques des modèles envisagés. Le sous-ensemble restant, identifié en tant que "test split", sera mis à l'écart des phases d'entraînement afin de n'intervenir qu'en phase finale d'évaluation des hypothèses de modélisation. Ce procédé permet de garantir la mise en place d'un jeu de données de référence pour l'évaluation et la comparaison des performances de différents modèles en écartant les risques d'introduction de biais lors de leurs entraînements.

Rééchantillonnage et équilibrage du jeu de données

Une des caractéristiques principales du jeu de données est le déséquilibre de représentativité entre les différentes classes cibles. En effet, 91 % des observations sont associées à un statut sans-défauts tandis que 9 % sont associées à un statut en défaut. Afin d'éviter de construire un modèle ne prédisant que la classe majoritaire et éviter notamment le risque de générer un modèle naïf, il convient de réaliser les phases d'entraînements sur un jeu de données équilibré.

Toutefois, le sous-ensemble de données test ne sera pas altéré et conservera une distribution de classes initiales afin de permettre l'évaluation des modèles sur un jeu de données représentatifs d'une situation réelle.

Pour atteindre l'équilibrage du jeu de données plusieurs approches sont possible, nous avons abordés 3 solutions différentes présentées ci-après :

- **Under Sampling**

  Cette méthode consiste à réduire le nombre d'observations associées à la classe majoritaire afin d'obtenir un nombre d'individu équivalent entre chaque classe. Cela a pour effet de réduire la population totale du jeu d'entraînement. Pour des soucis de performance, la méthode de sélection implémentée est celle d'un RandomUnderSampler proposé par la librairie *imblearn.*

- **Synthetic Minority Oversampling Technique**

  Cette méthode à pour effet de générer des observations synthétiques associées à la classe minoritaire en compilant des informations par la méthode des plus proches voisins. Cela à pour effet d'augmenter significativement le nombre total d'individus au sein du jeu d'entraînement. La méthode implémentée est celle du SMOTE proposé par la librairie *imblearn*.

- **Class Weights**

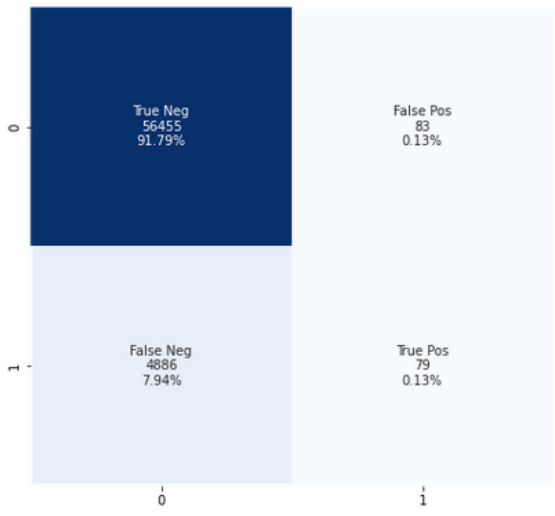
  Cette méthode permet de modifier le poids associé a chaque observation lors des phases d'entraînement de sorte que le coefficient associé aux membres de la classe minoritaire pénalise davantage la fonction de perte que la classe majoritaire. Plusieurs méthodes de pondération pourraient être appliquées, mais pour des soucis de simplicité la méthode implémentée est celle d'associer à la classe minoritaire un coefficient fixe correspondant au ratio entre le nombre d'individus de la classe majoritaire et celui de la classe minoritaire.

	Unbalanced	Under Sample	SMOTE	Class Weights
n_ class 0	226148	19860	226148	226148
n_ class 1	19860	19860	226148	19860

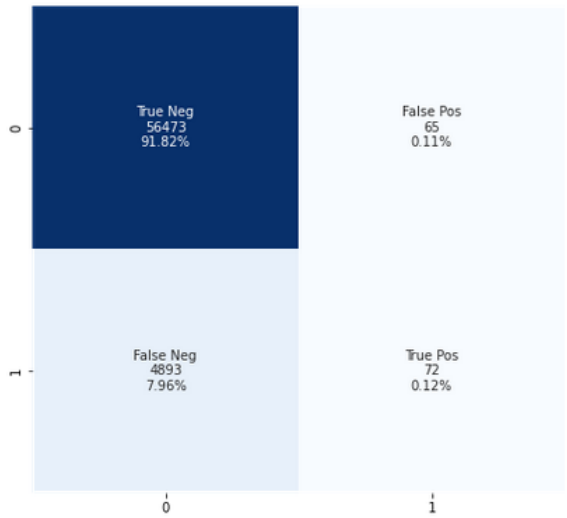
Tableau récapitulatif du nombre d'individus représentant chaque classe après équilibrage du train split

Chaque méthode d'équilibrage induit des performances de modélisation différentes. Une comparaison des performances à été opérée sur un modèle témoin de type LGBM. Il apparaît qu'un non-équilibrage ou un équilibrage par méthode SMOTE conduit à des performances de modélisation médiocre ou seul la classe initialement majoritaire peut être prédite. Les méthodes under sample et class weights permettent d'obtenir des performances similaires et raisonnables

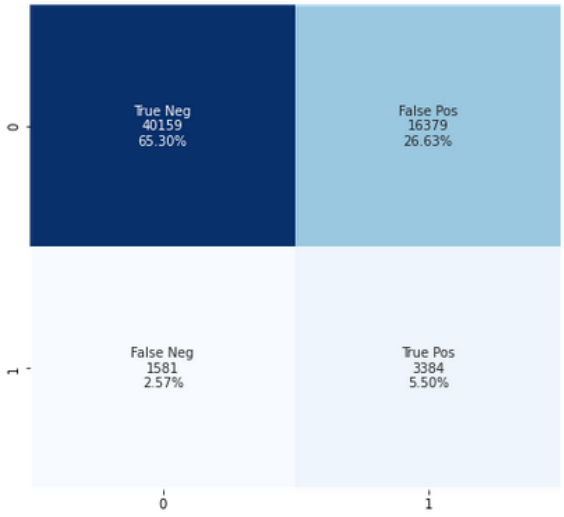
au regard des objectifs souhaités. La méthode d'équilibrage SMOTE sera donc abandonnée pour la suite.



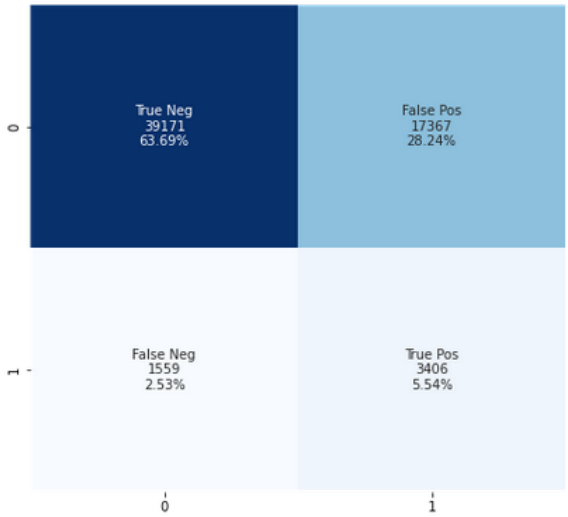
Pas d'équilibrage



SMOTE



Under Sample



Class Weights

Matrices de confusions d'un modèle LGBM entraîné avec un jeu de données équilibré selon différentes méthodes, a. pas d'équilibrage, b. SMOTE over-sampling, c. random under sampling, d. class weights.

Preprocessing & Feature Engineering

L'étape de preprocessing et de feature engineering est une approche minimale reprise du kernel kaggle source, sans altération majeure. Le code du kernel à été repris et adapté afin d'être exécuté au travers de pipeline respectant l'interface des Transformers proposé par le framework scikit-learn.

Une première phase de formatage consiste à encoder les variables qualitatives au travers d'un label encoder, et à corriger certaines valeurs manifestement aberrantes. Une seconde étape consiste à imputer les valeurs manquantes par la médiane de la variable en cours afin de lisser l'impact des outliers. Une étape de feature engineering est réalisé afin de calculer 4 nouvelles variables :

- CREDIT_INCOME_PERCENT
- ANNUITY_INCOME_PERCENT
- CREDIT_TERM
- DAYS_EMPLOYED_PERCENT

L'étape de polynomial feature engineering proposé dans le kernel source n'a pas été repris, sur la base des résultats non concluants que ceux-ci démontrent (cf notebooks du kernel source pour plus d'information).

Une méthode de feature engineering à été reprise d'un autre kernel ayant eu de bons résultats lors de la compétition kaggle sur ce jeu de données. Une variable KNN_300, correspondant à la moyenne des valeurs "TARGET" des 300 plus proches voisins de l'individu en cours, calculé sur les variables EXT_SOURCE (1, 2 et 3), AMT_ANNUITY et AMT_CREDIT.

Un MinMaxScaler est appliqué à l'ensemble du jeu de données afin de produire le format de données fournis en entrés pour l'étape d'entraînement et d'optimisation paramétrique du modèle.

Sélection du modèle

Différentes approches de modélisations ont été abordés, chacune mettant en oeuvre différentes type de modèles cités ci-aprés, chacun associés à une recherche d'hyper-paramètres par cross valiadation à 5 folds selon la méthode implémenté par la classe GridSearchCV de la librairie *scikit-learn*.

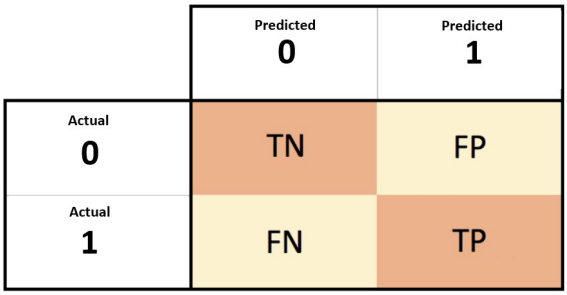
- LogisticRegression
- Random Forest Classifier
- XGBoost / LGBM

- Model Stacking (combinaison de 3 modèles cités ci dessus)

Le choix du meilleur modèle est effectué en retenant le modèle avec le meilleur score sur le sous-ensemble de test du jeu de données. Bien que les différentes approches présentent des résultats similaires, le modèle LGBM représente le meilleur compromis.

MÉTRIQUE D'ÉVALUATION

Le modèle de classification mis en place ne peut être parfait, de fait des erreurs sont possibles. Des clients identifiés sans risques peuvent à terme aboutir à un défaut de paiement, nous avons ici affaire à des faux négatifs. A l'inverse, des clients peuvent être identifiés à tort comme étant des profils à risque sans qu'aucun défaut réel ne soit constaté, il s'agit ici de faux positifs.



Matrice de confusion: TN True Negative, FN False Negative, FP False Positive, TP True Positive

Ces deux types d'erreurs ont un impact différent sur la vision métier de la société prêt à dépenser. Les Faux Négatifs représentent des pertes financières à la hauteur des montants de crédits non remboursés. On cherchera donc à minimiser le taux de faux négatifs, tandis que l'on cherchera à maximiser le taux de vrais positifs.

Autrement dit, nous cherchons a maximiser le recall :

$$Recall = \frac{TP}{TP+FN}$$

De la même manière, les Faux Positifs représentent un manque à gagner et un coût d'opportunités. On cherchera donc à limiter le nombre de faux positifs, pour cela nous pouvons donc chercher à maximiser la précision :

$$Précision = \frac{TP}{TP+FP}$$

La fonction $F\beta$ score représente la moyenne harmonique entre le recall et la précision,

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

cette fonction représente donc une métrique à maximiser adaptée à notre problématique.

Le paramètre β représentent un coefficient permettant de pondérer l'importance relative du recall face à la précision.

- $\beta < 1$ privilégie la précision
- $\beta > 1$ privilégie le recall

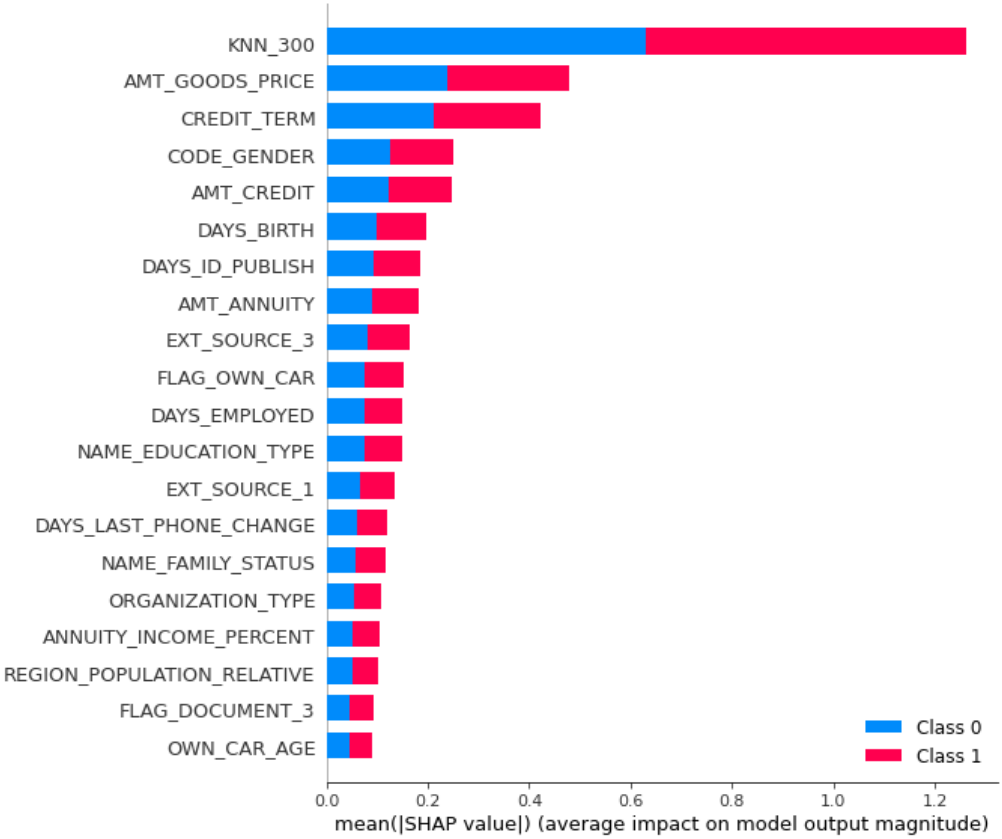
L'importance relative du risque de pertes financières par rapport au risque que représente le manque à gagner est un arbitrage qui revient aux responsables stratégiques de la société. Mais le risque de pertes financières reste vraisemblablement le plus important. Nous chercherons donc à associer un poids plus fort au recall plutôt qu'à la précision, c'est-à-dire un coefficient β supérieur à 1.

Une valeur précise de ce coefficient pourrait être mesuré en tenant compte du montant moyen des dépenses qu'entraîne un défaut de paiement ainsi que le coût d'opportunité moyen d'un client potentiel refusé. Les données à disposition ne permettent que de répondre partiellement à ce problème, l'analyse de ce point n'a pas été poussée plus loin. De fait, nous assumons une valeur arbitraire pour le coefficient $\beta = 2$.

INTERPRÉTABILITÉ DU MODÈLE

L'interprétabilité globale et locale est permise grâce à la librairie SHAP (SHapley Additive exPlanations) qui propose une méthode d'expliquabilité générique adaptée à tout type de modèle de machine learning.

L'expliquabilité globale peut être observée par l'analyse de feature importance par permutation afin d'identifier sur quelles variables s'appuie le modèle pour effectuer ses prédictions. La feature importance globale représente l'impact moyen de chaque variable sur l'ensemble du jeu de données.



Ici, le statut de défaut des 300 plus proches voisins, représenté par la variable KNN_300 est la variables ayant l'importance globale la plus forte, suivis par les informations sur le montant du bien (AMT_GOODS_PRICE), la durée du crédit (CREDIT_TERM) ainsi que le montant du crédit (AMT CREDIT).

La libraire SHAP permet également de visualiser l'importance locale des variables. C'est-à-dire à l'échelle d'un client, quels sont les critères ayant le plus d'importance dans l'estimation faite par le modèle.

Ici, l'âge et la possession d'un véhicule personnel sont des critères favorables à l'obtention du prêt, tandis que la situation précaire du client ainsi que le montant d'apport externe et le montant du crédit total ont un impact négatif sur le calcul du score client.

LIMITES ET AMÉLIORATION

Nous avons observé tout au long du processus de modélisation des performances relativement pauvre. La première piste d'amélioration peut être orientée sur des méthodes de pré-processing plus avancées. En effet, le kernel source pris en exemple se limite à des actions très basique, une étape de feature engineering en phase avec le domaine métier seraient d'une grande utilité pour identifier des variables utiles et synthétiques.

La définition d'une fonction score plus précise pourraient également être une autre piste d'amélioration. En reprenant la fonction Fβ score, le coefficient β pourrait être mesuré avec plus d'assurance. Mais également, d'autres métriques pourraient faire sens après discussion avec les équipes métiers.