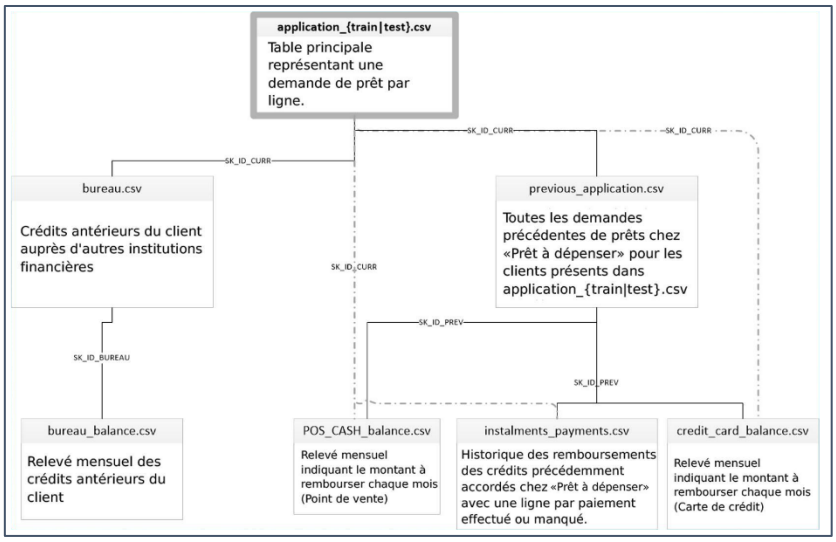
La méthodologie d'entraînement du modèle

Jeu de données

Huit fichiers au format csv sont fournis et composent notre jeu de données. Ils contiennent 218 informations bancaires et personnelles anonymisées pour 307511 clients recueillies auprès de Home Crédit Group et auprès d'autres institutions financières.



La variable cible à prédire prend 2 valeurs et est fortement déséquilibrée. Modélisation en classification binaire, il faudra faire attention car certains algorithmes sont sensibles aux données déséquilibrées.

Pré-traitement des données

- Nettoyage : transformation des types, correction des valeurs aberrantes, harmoniser les valeurs uniques, suppression de valeurs manquantes (supérieures à un taux), imputation par médiane/mode
- Feature Engineering : création de nouvelles variables :
    - Automatique : à partir de variables numériques (moyenne, minimum, maximum, count, écart-type
    - Manuelle : compréhension métier (différence, ratio…)
- Assemblage : assembler tous les fichiers en utilisant les différents identifiants, chaque client représente une ligne unique
- Feature Selection : sélection de variables, trouver les variables pertinentes et minimiser la perte d'information
    - Filtrage : suppression de variables colinéaires (supérieure à un coefficient de 0.8)
    - Automatique : Boruta, BorutaShap, LightGBM

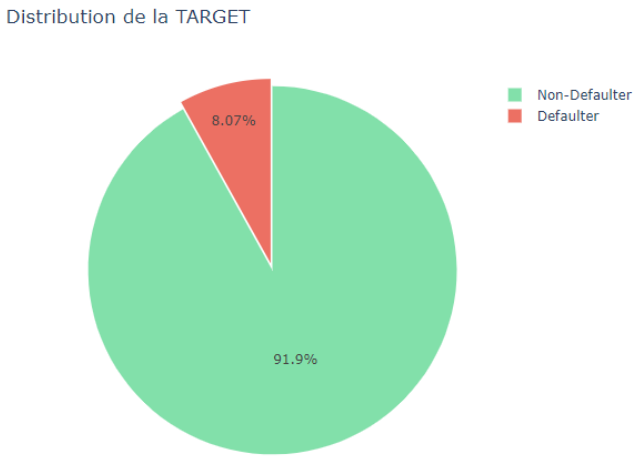
Plusieurs modèles pour une première idée :

	Model	Accuracy	AUC	Precision	Recall	F1 Score	Duration (s)
0	Dummy Classifier	0.916300	0.500000	0.000000	0.000000	0.000000	14.437106
1	Logistic Regression	0.915700	0.524378	0.033333	0.000291	0.000577	102.792960
2	LGBM	0.916050	0.755509	0.475732	0.037225	0.068964	25.767222
3	CatBoost	0.916400	0.762968	0.506176	0.033698	0.063120	232.737668
4	XGBoost	0.913750	0.729554	0.402847	0.061451	0.106526	28.925791
5	Random Forest	0.916300	0.718234	0.483333	0.002084	0.004149	363.728751
6	Gradient Boosting	0.916350	0.759331	0.507741	0.029826	0.056250	809.791891
7	AdaBoost	0.915425	0.744897	0.446886	0.040229	0.073561	167.412745
8	Extra Trees	0.916300	0.720354	0.000000	0.000000	0.000000	109.194871
9	Bagging Classifier	0.912525	0.655885	0.303106	0.034990	0.062704	809.409461

LGBM modèle retenu pour sa rapidité, composante essentielle pour un dashboard interactif.

Le traitement du déséquilibre des classes

La target (0 = Aucun problème de paiement | 1 = Soucis de paiement) est très déséquilibrée.



Un tel déséquilibre soulève de nouvelles problématiques :

1) Performance trompeuse : Un modèle qui prédit toujours la classe majoritaire obtiendra une précision (accuracy) élevée malgré le fait qu'il ne soit pas vraiment efficace.

2) Biais du modèle : Car il est davantage "récompensé" pour avoir correctement prédit cette classe étant donnée sa prédominance.

Plusieurs méthodes de rééquilibrage sont à notre disposition :

- undersampling : consiste en un sous-échantillonnage de la classe majoritaire.
- oversampling : sur-échantillonnage de la classe minoritaire (ex : SMOTE, ADASYN).
- une combinaison d'oversampling suivi d'undersampling (possible avec SMOTE).
- hyperparamètre du modèle Ligthgbm : class_weight='balanced'.

La fonction coût métier et la métrique d'évaluation

Matrice de confusion : elle permet de comptabiliser les cas suivants :

TN (vrais négatifs) : la banque refuse un prêt qui aurait été remboursé.

FP (faux positifs) : la banque refuse un prêt qui aurait pu être remboursé.

FN (faux négatifs) : la banque accorde un prêt, mais le client fait défaut.

TP (vrais positifs) : la banque refuse à juste titre un prêt risqué.

Gain total (gain_tot) : calculé en utilisant les taux associés à chaque type de prédiction et en multipliant ces taux par le nombre de cas respectifs dans la matrice de confusion :

En pénalisant 10x plus les FN que les FP.

gain_tot=(tn×taux_tn) + (fp×taux_fp) + (fn×taux_fn) + (tp×taux_tp)

Gain maximum (gain_max) : hypothèse où toutes les prédictions sont correctes (aucune erreur de classification) :

gain_max=(fp+tn) × taux_tn + (fn+tp) × taux_tp

Gain minimum (gain_min) : hypothèse où la banque fait uniquement des erreurs (aucune prédiction correcte) :

gain_min=(fp+tn) × taux_fp + (fn+tp) × taux_fn

Score final (custom_score) : il s'agit d'une normalisation du gain total entre le gain minimum et le gain maximum, de manière à obtenir un score compris entre 0 et 1 :

$$CustomScore = \frac{Gain_{tot} - Gain_{min}}{Gain_{max} - Gain_{tot}}$$

Un score élevé (proche de 1) montre une bonne performance du modèle, avec un équilibre entre les prédictions correctes et les pertes potentielles.

En résumé :

Cette fonction calcule un score basé sur les gains ou pertes économiques associés aux décisions de la banque. Le but est de maximiser les vrais négatifs et minimiser les faux négatifs, car accorder un prêt à une personne qui ne rembourse pas est la situation la plus coûteuse.

AUC (Area Under Curve)

Elle mesure l'aire sous la courbe ROC (Receiver Operating Characteristic). La courbe ROC trace le taux de vrais positifs par rapport au taux de faux positifs à différents seuils de classification. Une AUC de 1 indique une classification parfaite, tandis qu'une AUC de 0,5 indique une performance équivalente à une prédiction aléatoire.

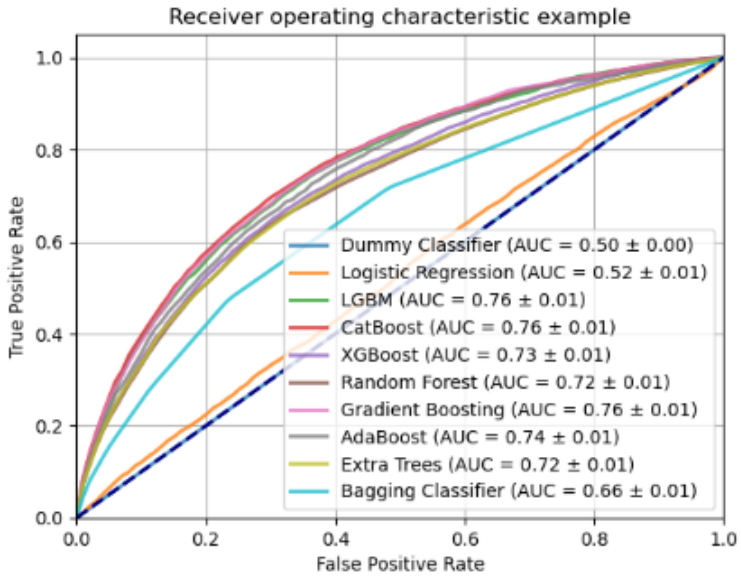

Receiver operating characteristic example
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Dummy Classifier (AUC = 0.50 ± 0.00)
Logistic Regression (AUC = 0.52 ± 0.01)
LGBM (AUC = 0.76 ± 0.01)
CatBoost (AUC = 0.76 ± 0.01)
XGBoost (AUC = 0.73 ± 0.01)
Random Forest (AUC = 0.72 ± 0.01)
Gradient Boosting (AUC = 0.76 ± 0.01)
AdaBoost (AUC = 0.74 ± 0.01)
Extra Trees (AUC = 0.72 ± 0.01)
Bagging Classifier (AUC = 0.66 ± 0.01)

Tableau de synthèse des résultats

	Modèle	Jeu_donnees	FN	FP	TP	TN	Metrique	Optimisation	Class_weight	Recall	Precision	F1 Score	F5 Score	F10 Score
10	lgbm_optuna_opt_5	train	1471	15908	3494	40629	F10	optuna	oui	0.703726	0.180085	0.286781	0.632940	0.684033
12	lgbm_optuna_opt_F5	train	1483	16198	3482	40339	F5	optuna	non	0.701309	0.176931	0.282573	0.629547	0.681317
5	lgbm_hyperparam_base_std_bal	train	1516	15054	3449	41483	logloss	std	oui	0.694663	0.186402	0.293932	0.628726	0.676402
9	lgbm_optuna_opt_4	train	1562	14871	3403	41666	pr_auc	optuna	oui	0.685398	0.186221	0.292870	0.621339	0.667677

```
LGBMClassifier
LGBMClassifier(class_weight='balanced', colsample_bytree=0.942643005209126,
               force_col_wise=True, max_depth=8, min_child_samples=47,
               min_child_weight=0.5225481447282847, n_jobs=-1, num_leaves=14,
               objective='binary', reg_alpha=1.3788599856023566e-05,
               reg_lambda=1.4867533700934656e-07, subsample=0.5572465639400604,
               subsample_freq=2, verbosity=-1)
```

Le modèle **lgbm_optuna_opt_5** (LightGBM class_weight='balanced' avec une optimisation bayésienne) détecte :

- le moins de faux négatifs (des clients détectés non défaillants mais qui ne remboursent pas le prêt)
- et le plus de vrais positifs (des clients détectés défaillants qui sont défaillants),

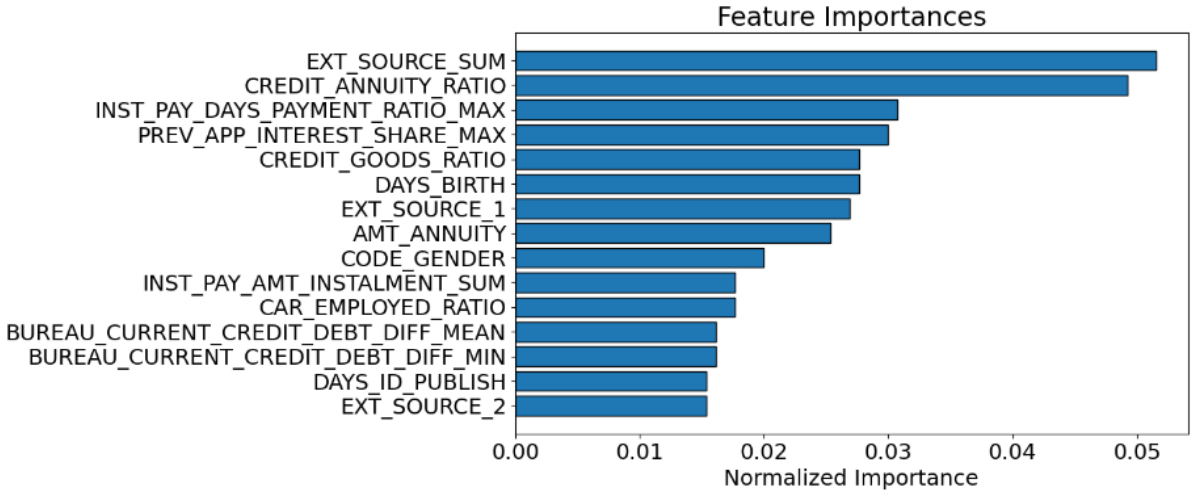
On a tenté d'optimiser ce modèle **lgbm_otuna_opt_5** en recherchant le seuil de probabilité optimal ayant un impact sur la décision de classer en classe 0 (non-défaillant) ou 1 (défaillant) : pas concluant car détection de plus de TN et le compromis faux négatifs et faux positifs devra être discuté avec nos clients pour régler au mieux le seuil de probabilité avec le nombre de faux négatifs/positifs à atteindre.

Sans client disponible et sans consigne dans le projet, le seuil par défaut à 0.5 sera conservé.

L’interprétabilité globale et locale du modèle

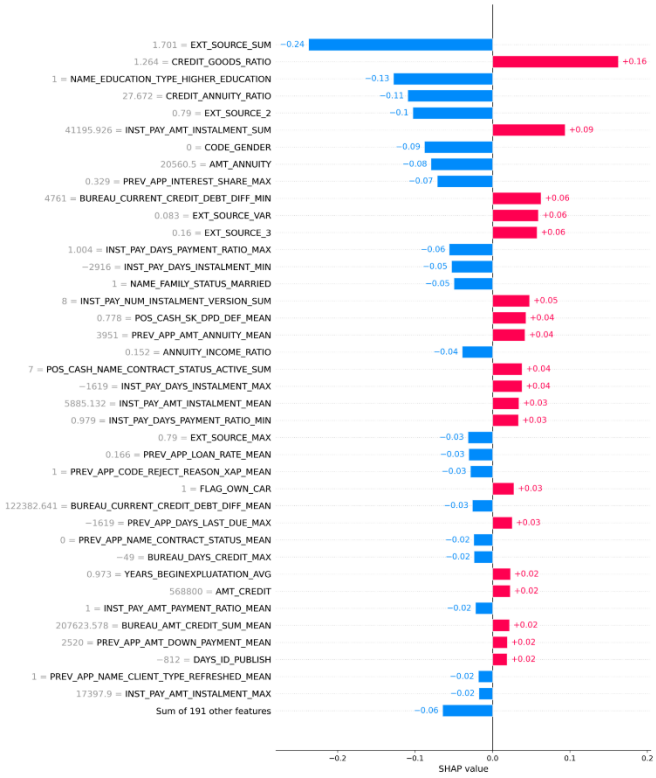
Interprétabilité globale

L'interprétabilité globale d'un modèle se réfère à notre capacité à comprendre comment le modèle prend ses décisions pour l'ensemble d’un jeu de données. Pour répondre à ces questions, l’importance des features a été utilisée. Cette métrique est intéressante pour détecter du data leakage (fuite de données). Le data leakage se produit lorsque des informations provenant de la variable cible s'infiltrent dans les caractéristiques utilisées pour former le modèle, ce dernier pourrait par conséquent afficher une performance artificiellement élevée.



Interprétabilité locale

Contrairement à l'interprétabilité globale, qui cherche à comprendre comment un modèle fonctionne sur l'ensemble de ses données, l'interprétabilité locale cherche à comprendre pourquoi un modèle a fait une prédiction particulière pour un seul échantillon. Par exemple, la méthode SHAP (SHapley Additive exPlanations) sera utilisée pour expliquer la probabilité donnée par le modèle. Cette méthode attribue une valeur à chaque fonctionnalité. Si cette valeur est positive, alors elle a tendance à augmenter la probabilité finale, et inversement.



Les limites et les améliorations possibles

Utilisation d'un kernel Kaggle pour la création du dataset final

Pour accélérer ce processus, l'utilisation d'un kernel Kaggle fut choisie, disponible à l'adresse suivante : (https://www.kaggle.com/code/jsaguiar/lightgbm-with-simple-features/script)

Ce choix est intéressant car ce kernel en particulier donne des scores tout à fait satisfaisants. Cependant, il serait toujours judicieux de vouloir améliorer ce kernel en ajoutant nos propres approches concernant l'analyse exploratoire, la préparation des données et le Feature engineering.

Le manque d'expertise

Bien que l'expertise en science des données soit axée sur la manipulation, l'analyse et la modélisation des données, les critères d'octroi de crédit présentaient des spécificités complexes liées au domaine de la banque.

La compréhension de chaque Feature était un défi de taille, par conséquent, la collaboration avec un expert du domaine bancaire aurait apporté une valeur ajoutée significative. Cet expert aurait pu aider à clarifier la signification et la pertinence de chaque variable et à s'assurer de l'adéquation des méthodes de traitement des données.

Enfin, une évaluation du Dashboard par le client permettrait d'apporter des améliorations techniques et esthétiques en fonction de ses attentes.

L’analyse du Data Drift

Le Data Drift se réfère à une modification des distributions de données au fil du temps par rapport aux données initiales sur lesquelles le modèle a été formé. Cela peut avoir un impact sur les performances du modèle, car si les données d'entrée évoluent de manière significative par rapport aux données d'entraînement, les prédictions du modèle peuvent ne plus être précises ou pertinentes.

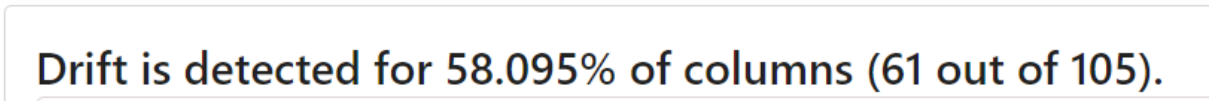
L’hypothèse qui fut choisie est que le dataset “application_train” représente les données pour la modélisation et le dataset “application_test” représente les nouveaux clients une fois le modèle en production.

Pour les colonnes numériques, le test statistique de Kolmogorov-Smirnov sera utilisé, et pour les colonnes catégorielles, l’indicateur PSI (Population Stability Index) sera choisi. Un seuil de 0,2 fut défini pour chaque test statistique :

- Si la p-value du test K-S est inférieure à 0,2, cela suggère que les deux échantillons ne proviennent pas de la même distribution à un niveau de confiance de 80%.

- Si le PSI est supérieur à 0,2, cela suggère aussi une indication de Data Drift.

Le package evidently permet d’effectuer cette analyse de Data Drift et de créer une page html permettant l’analyse de ces résultats. Il est considéré que si plus de la moitié des fonctionnalités possèdent du Data Drift, alors le Data Drift sur l’ensemble des données est détecté.



Il est noté que 58% des fonctionnalités possèdent du Data Drift. En conclusion, le Data Drift a eu lieu entre application_train et application_test.