A dark blue vertical bar on the left side of the page. A blue arrow points to the right from the bar, containing the date.

26/08/2023

Note méthodologique

Projet 7 OpenClassrooms :
Implémentez un modèle de scoring

Several thin, curved lines in dark blue and light grey that originate from the bottom left and sweep upwards and to the right.

Keshika Dabidin Audam

LIEN GITHUB : [HTTPS://GITHUB.COM/KESHIKA-DABIDIN-AUDAM/PROJET-7/TREE/MASTER](https://github.com/Keshika-Dabidin-Audam/projet-7/tree/master)

Table des matières

1. Introduction	2
1.1. Récupération des données.....	2
1.2. Analyse Exploratoire des données	2
1.2.1. La variable TARGET	2
1.2.2. Résumé des données sur les clients.....	2
2. Prétraitement des données et le traitement des déséquilibre des classes	5
2.1. Prétraitement et feature engineering	5
2.2. Traitement du déséquilibre des classes	5
3. La méthodologie d'entraînement du modèle (2 pages maximum)	6
3.1. Classification binaire	6
3.2. Algorithmes et suivi pour construire le modèle de scoring	6
3.3. Sélection des variables pertinentes	9
4. La fonction coût métier, l'algorithme d'optimisation et la métrique d'évaluation	10
4.1. La Fonction coût métier	10
4.2. L'algorithme d'optimisation.....	11
5. Tableau de synthèse des résultats	12
6. L'interprétabilité globale et locale du modèle	12
7. Pipeline de déploiement	13
8. Les limites et les améliorations possibles	16
9. L'analyse du Data Drift.....	16

1. Introduction

L'entreprise 'Prêt à Dépenser' souhaite développer un modèle de Scoring de la probabilité de défaut de paiement du client afin d'assister à la décision d'accorder ou non un prêt à un client potentiel en s'appuyant sur des sources de données variées (données personnelles, données comportementales, données provenant d'autres institutions financières, etc.).

Le modèle de Scoring s'établit à partir des algorithmes de machine learning. Ces algorithmes divergent dans leurs méthodes de modélisation mais se basent sur le même principe. Le projet a été traité à l'aide de 3 algorithmes de gradient boosting (LightGBM, XGBoost et CatBoost) utilisés en raison de leur simplicité d'interprétation et aussi en raison de leur efficacité.

La démarche suivante est appliquée pour mettre en place le modèle de machine learning :

1.1. Récupération des données

La première étape consiste à récupérer des données cohérentes afin de former la base sur laquelle l'algorithme de Machine Learning va apprendre. Des données relatives à différents clients sont récupérées sur [Kaggle](#).

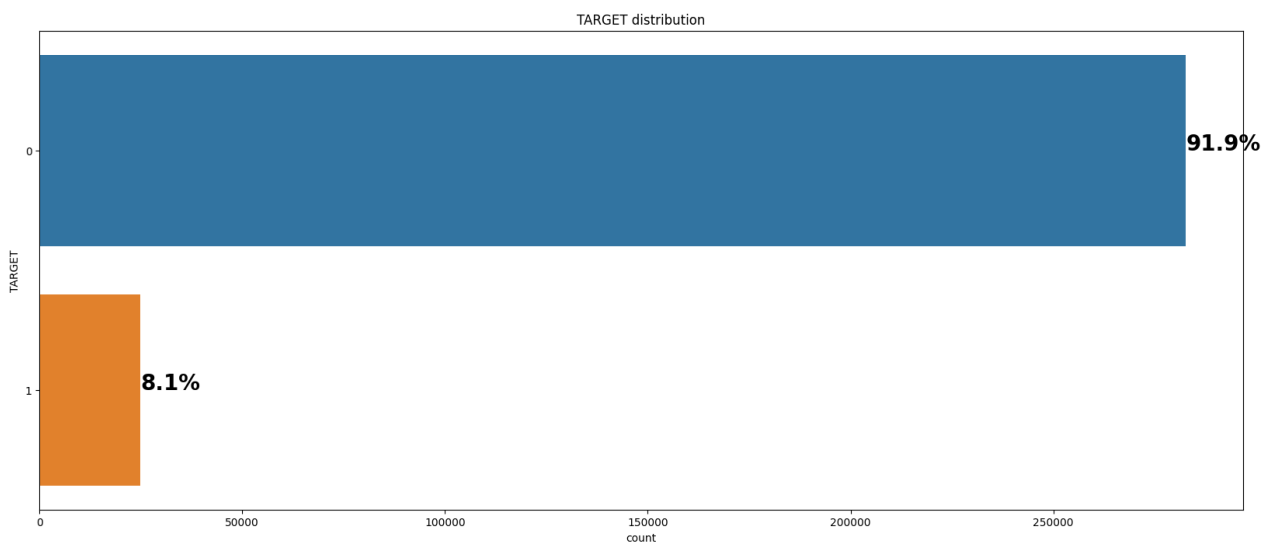
Un [kernel Kaggle](#) est utilisé pour faciliter l'exploration des données nécessaires à l'élaboration du modèle de scoring. La mise en œuvre de statistiques descriptives découle ensuite et permet de comprendre et d'avoir une vue d'ensemble des données grâce à de la visualisation graphique.

1.2. Analyse Exploratoire des données

L'entreprise met à disposition 7 fichiers CSV contenant des données spécifiques à certains paramètres.

Nous disposons d'une base de données nommée « train », qui nous a servi à entraîner notre modèle. Cette base contient une variable « cible ». Il y a 307511 observations pour chaque prêt séparé et 122 features incluant le TARGET (label que l'on souhaite prédire) Le target est 1 si l'individu est éligible pour le prêt, 0 sinon.

1.2.1. La variable TARGET



Le jeu de données est non-équilibré. Il y a beaucoup plus de prêts remboursés à temps que de prêts non remboursés. Une fois que nous entrons dans des modèles d'apprentissage automatique plus sophistiqués, nous pouvons pondérer les classes par leur représentation dans les données pour refléter ce déséquilibre.

1.2.2. Résumé des données sur les clients

1. Les valeurs manquantes sont plus présentes dans les caractéristiques des habitats.

2. Le nombre de jours de travail possèdent des anomalies (travail plus de 1000 ans pour certains clients) et a été modifié avec des 'Nans' puis rempli avec l'algorithme de Simple Imputer.

3. Les prêts renouvelables ne représentent qu'une petite fraction (10%) du nombre total de prêts. Cependant, un plus grand nombre de crédits renouvelables, par rapport à leur fréquence, ne sont pas remboursés.

4. Le nombre de clients féminins est presque le double du nombre de clients masculins. En ce qui concerne le pourcentage de crédits en souffrance, les hommes ont plus de chances de ne pas rembourser leurs prêts (10%), comparativement aux femmes (7%).

5. En termes de pourcentage de non-remboursement du prêt, le mariage civil a le pourcentage le plus élevé de non-remboursement (10%), la veuve étant le plus bas.

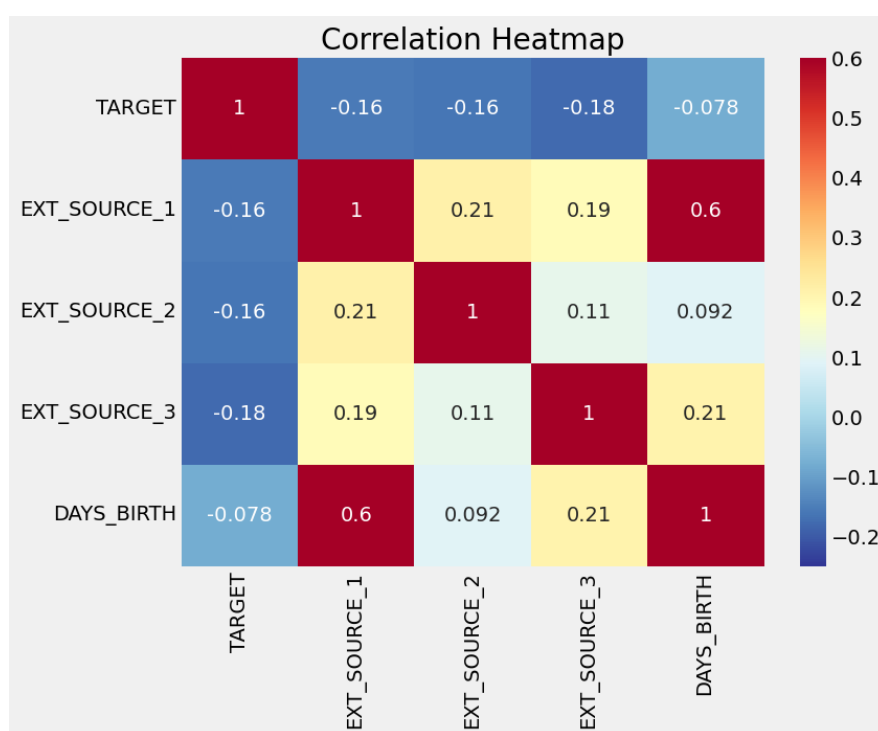
6. Les demandeurs avec le type de revenu Congé de maternité ont un ratio de près de 40% de prêts non remboursés, suivis des chômeurs (37%). Les autres types de revenus sont inférieurs à la moyenne de 10% pour ne pas rembourser les prêts.

7. La plupart des prêts sont contractés par des ouvriers, suivis par les vendeurs/commerciaux. Le personnel informatique prend le montant de prêts le plus bas. La catégorie avec le pourcentage le plus élevé de prêts non remboursés est celle des ouvriers peu qualifiés (plus de 17%), suivis des chauffeurs et des serveurs / barmen, du personnel de sécurité, des ouvriers et du personnel de cuisine.

8. La catégorie du premier cycle du secondaire, bien que rare, a le taux le plus élevé de non-remboursement du prêt (11%). Les personnes ayant un diplôme universitaire ont un taux de non-remboursement inférieur à 2%.

9. Les loueurs d'appartements (non propriétaires de leur résidence principale), ainsi que ceux qui vivent chez leurs parents, ont un taux de non-remboursement supérieur à 10%.

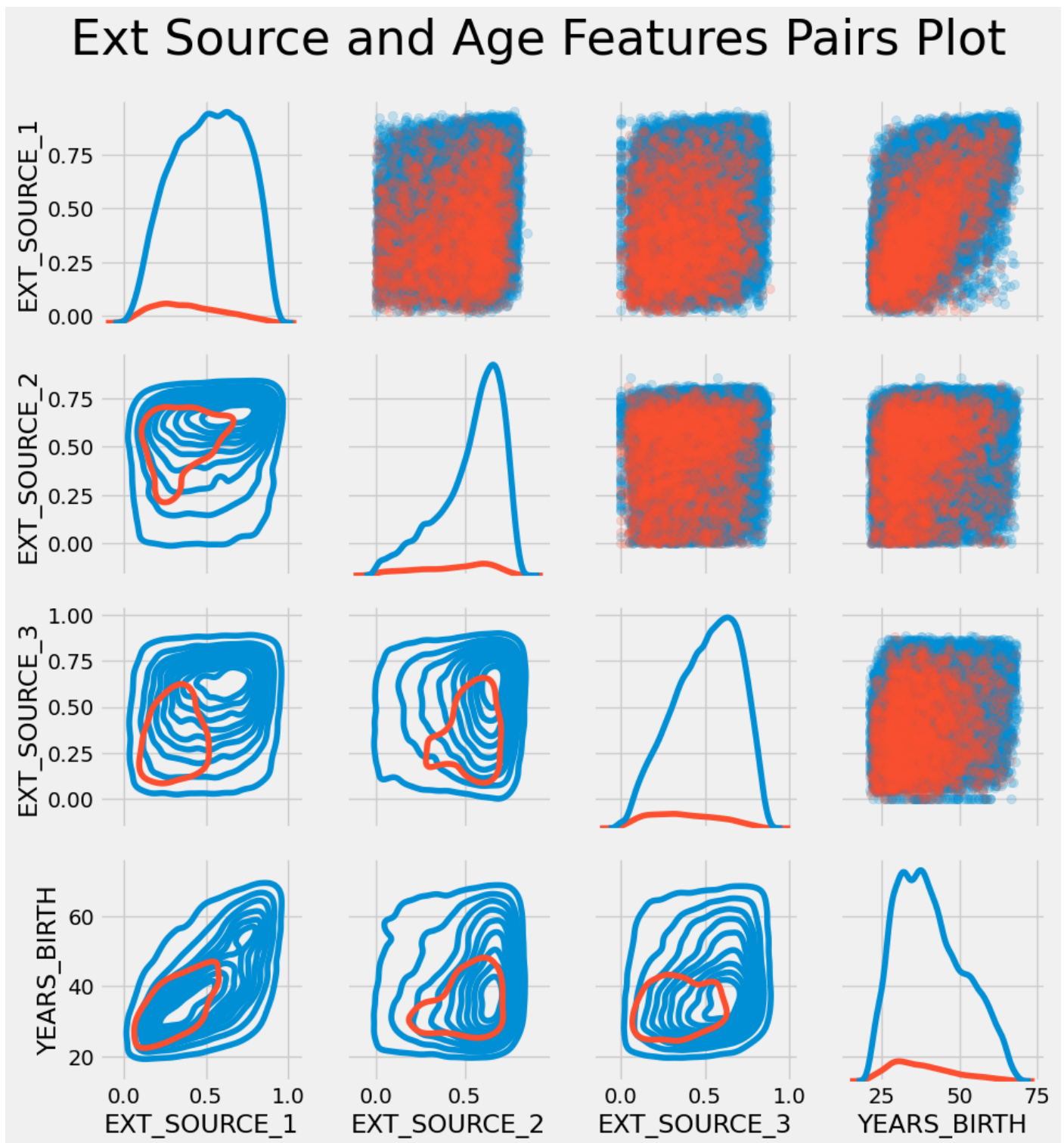
10. Diagramme de corrélations :



Les trois caractéristiques EXT_SOURCE ont des corrélations négatives avec la cible, ce qui indique qu'à mesure que la valeur de EXT_SOURCE augmente, le client est plus susceptible de rembourser le prêt. Nous pouvons également voir que DAYS_BIRTH est positivement corrélé avec EXT_SOURCE_1 indiquant que l'un des facteurs de ce score est peut-être l'âge du client.

Ensuite, nous pouvons examiner la distribution de chacune de ces caractéristiques colorées par la valeur de la cible. Cela nous permettra de visualiser l'effet de cette variable sur la cible.

11. Pairs plot :



Dans ce graphique, le rouge indique les prêts qui n'ont pas été remboursés et le bleu les prêts qui sont payés. Nous pouvons voir les différentes relations au sein des données. Il semble y avoir une relation linéaire positive modérée entre EXT_SOURCE_1 et DAYS_BIRTH (ou de manière équivalente YEARS_BIRTH), indiquant que cette caractéristique peut prendre en compte l'âge du client.

2. Prétraitement des données et le traitement des déséquilibre des classes

2.1. Prétraitement et feature engineering

Les données ont été fusionnées et retravaillées à partir des fichiers train, test, fichiers bureaux, fichiers liés au cash balance ainsi que les fichiers liés aux applications précédentes. Les étapes précédentes consistaient uniquement à établir des liens entre nos fichiers, des fusions de table dans le but d'enrichir la base de données.

On a pu facilement extraire 3 variables de moyenne et de comptage :

1. PREVIOUS_LOANS_COUNT de bureau.csv : Nombre total des précédents crédits pris par chaque client.
2. MONTHS_BALANCE_MEAN de bureau_balance.csv : Solde moyen mensuel des précédents crédits.
3. PREVIOUS_APPLICATION_COUNT de previous_application.csv : Nombre de demandes antérieures des clients au crédit immobilier.

On a également créé des nouvelles variables pertinentes pour la suite de la modélisation qui sont les suivantes :

1. CREDIT_INCOME_PERCENT : Pourcentage du montant du crédit par rapport au revenu d'un client.
2. ANNUITY_INCOME_PERCENT : Pourcentage de la rente de prêt par rapport au revenu d'un client.
3. CREDIT_TERM : Durée du paiement en mois.
4. DAYS_EMPLOYED_PERCENT : Pourcentage des jours employés par rapport à l'âge du client.

En ce qui concerne le feature engineering, voici les différentes étapes qui ont été effectuées :

1. Splitting en train/test (80%/20%)

Training Features shape with categorical columns: (307511, 192)

Testing Features shape with categorical columns: (48744, 191)

2. Encodage des features catégorielles avec le label encoder et get_dummies.
3. Imputation des valeurs manquantes par la moyenne ou la médiane et traitement des valeurs aberrantes.
4. Standardisation des données avec le MinMaxScaler.
5. Une version de données avec les valeurs manquantes a également été conservée car certains algorithmes tels que le light gradient boost peuvent effectuer des prédictions même avec des valeurs manquantes.

2.2. Traitement du déséquilibre des classes

Le problème est un problème de classification binaire avec une classe sous représentée (9 % de clients en défaut contre 91 % de clients sans défaut). Ce déséquilibre des classes doit être pris en compte dans l'entraînement des modèles puisqu'un modèle de base effectuera des prédictions majoritairement naïves, il prédira que les clients sont sans défaut aurait une justesse (accuracy) de 0.92 et pourrait être considéré à tort comme un modèle performant alors qu'il ne permettrait pas de détecter les clients à risque.

Deux approches pour rééquilibrer les deux classes ont été testées, Sample Weights et Smote :

1. [SMOTE](#) permet de créer des données synthétiques à partir des données existantes. Nous avons utilisé une librairie Python nommée Imbalance learn. Dans notre cas, dans un souci d'optimisation de temps de calculs, nous avons utilisé le Random Under Sampler qui s'avère le plus rapide. Le tableau suivant résume la répartition des données avant et après l'application de SMOTE :

	Classe 0	Classe 1
Avant Rééquilibrage	197845	17412
Après Rééquilibrage	197845	197845

2. [Sample Weights \(scale_pos_weight X\)](#) permet de modifier les poids associés aux observations de sorte qu'une observation mal classée dans la classe minoritaire pénalise davantage la fonction de perte qu'une observation mal classée dans la classe majoritaire. X est la valeur du coefficient qu'on souhaite attribuer pour la pénalité.

3. La méthodologie d'entraînement du modèle (2 pages maximum)

3.1. Classification binaire

La classification consiste à identifier les classes d'appartenance de nouveaux objets à partir d'exemples antérieurs connus. Dans le contexte métier du projet, la classification est binaire représentée par une variable de sortie à deux classes, à savoir acceptation du crédit ou refus du crédit. Les décisions d'octroi de crédit sont prédéterminées, d'où le cadre de l'apprentissage supervisé.

3.2. Algorithmes et suivi pour construire le modèle de scoring

Dans le cadre de ce projet, le modèle de baseline (Régression Logistique) ainsi que trois algorithmes de gradient boosting (CatBoost, XGBoost et LightBoost) ont été testés sous différentes conditions : sans équilibrage de données, avec équilibrage en utilisant SMOTE et avec équilibrage en testant plusieurs sample weights.

Le suivi des tests a été effectué avec MLFlow, un outil faisant partie des processus de base de MLOps. Voici un exemple d'algorithme intégrant MLFlow :

```
with mlflow.start_run():

    trained_model=model.fit(X_train, y_train)
    predicted_qualities=trained_model.predict(X_test)

    #Evaluation metrics
    (rmse,mae,r2)=eval_metrics(y_test,predicted_qualities)
    accuracy=accuracy_score(y_test, predicted_qualities)
    precision=precision_score(y_test,predicted_qualities)
    recall=recall_score(y_test,predicted_qualities)
    f_one=metrics.f1_score(y_test,predicted_qualities)

    #print("Model used:" % str(model))
    print(" RMSE: %s" % rmse)
    print(" MAE:%s" % mae)
    print(" R2:%s" % r2)
    print(" Accuracy: %s" % accuracy)
    print(" Precision:%s" % precision)
    print(" Recall:%s" % recall)
    print(" F-1 Score: %s" % f_one)

    roc_auc = roc_auc_score(y_test, trained_model.predict_proba(X_test)[:,:1])
    print('AUC : %0.4f' %roc_auc)
    print(classification_report(y_test, trained_model.predict(X_test)))
    cf_matrix_roc_auc(model, y_test, trained_model.predict(X_test), trained_model)

    mlflow.log_metric("rmse",rmse)
    mlflow.log_metric("r2",r2)
    mlflow.log_metric("mae",mae)
    mlflow.log_metric("accuracy",accuracy)
    mlflow.log_metric("precision",precision)
    mlflow.log_metric("recall",recall)
    mlflow.log_metric("F-1 score",f_one)
    mlflow.log_metric("AUC",roc_auc)

    mlflow.sklearn.log_model(model,"model")
```

Voici à quoi peut ressembler le tableau de suivi des différents tests effectués sur MLFlow :

mlflow 2.4.1 Experiments Models

Experiments

Search Experiments

- Default
- Optimized_lightboost_with_rfe_and_custom...
- Recursive_feature_selection_experiments
- Gradient-Boosting-models
- Linear-Regression-SMOTE
- Linear-Regression

Displaying Runs from 5 Experiments

Table view Chart view Artifact view

metrics.rmse < 1 and params.model = "tree"

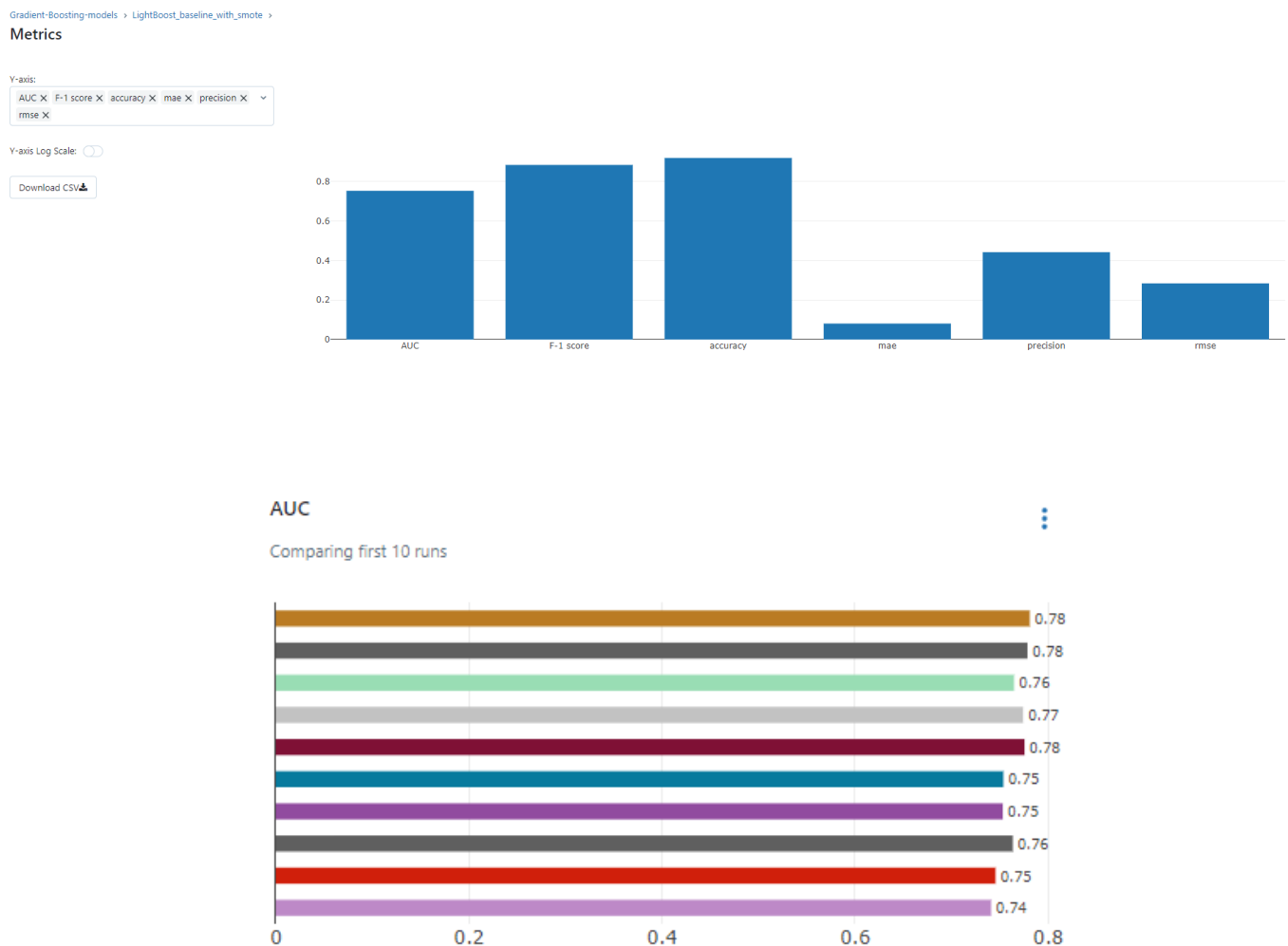
Time created State Active

Sort: Created Columns

	Run Name	Created	Duration	Source	Models	Metrics					
						AUC	F-1 score	accuracy	mae	precision	r2
	LightBoost_optimized_2	19 days ago	28.7s	C:\Users\...	sklearn	0.781	0.887	0.92	0.08	0.567	
	Lightboost_optimized_1	19 days ago	25.5s	C:\Users\...	sklearn	0.779	0.886	0.92	0.08	0.565	
	XGBoost_baseline_with_rfe	20 days ago	9.3s	C:\Users\...	sklearn	0.765	0.888	0.919	0.081	0.456	
	Lightboost_baseline_with_rfe	20 days ago	10.7s	C:\Users\...	sklearn	0.774	0.886	0.92	0.08	0.542	
	Catboost_baseline_with_rfe	20 days ago	20.1s	C:\Users\...	sklearn	0.775	0.886	0.92	0.08	0.562	
	XGBoost_baseline_with_smote	20 days ago	16.2s	C:\Users\...	sklearn	0.754	0.888	0.918	0.082	0.433	
	LightBoost_baseline_with_smote	20 days ago	17.3s	C:\Users\...	sklearn	0.753	0.884	0.919	0.081	0.442	
	Catboost_baseline_with_smote	20 days ago	1.1min	C:\Users\...	sklearn	0.763	0.885	0.92	0.08	0.522	
	lr_c_1_0_max_iter_100_smote	20 days ago	26.4s	C:\Users\...	sklearn	0.745	-	-	0.3	-	
	lr_c_0_001_max_iter_100_smote	20 days ago	28.8s	C:\Users\...	sklearn	0.741	-	-	0.306	-	
	XGBoost_scale_pos_weight_5	20 days ago	15.9s	C:\Users\...	sklearn	0.762	0.305	0.869	0.131	0.266	
	Lightboost_scale_pos_weight_5	20 days ago	11.6s	C:\Users\...	sklearn	0.776	0.32	0.865	0.135	0.269	
	Catboost_scale_pos_weight_5	20 days ago	20.4s	C:\Users\...	sklearn	0.778	0.324	0.872	0.128	0.282	
	XGBoost_scale_pos_weight_3	20 days ago	10.9s	C:\Users\...	sklearn	0.764	0.265	0.901	0.099	0.328	
	LBGBMBoost_scale_pos_weight_3	20 days ago	11.8s	C:\Users\...	sklearn	0.776	0.272	0.905	0.095	0.354	
	Catboost_scale_pos_weight_3	20 days ago	18.4s	C:\Users\...	sklearn	0.777	0.269	0.906	0.094	0.361	
	Lightboost_baseline	20 days ago	12.2s	C:\Users\...	sklearn	0.767	0.096	0.919	0.081	0.45	
	XGBoost_baseline	20 days ago	17.5s	C:\Users\...	sklearn	0.774	0.059	0.92	0.08	0.531	
	Catboost_baseline	20 days ago	23.0s	C:\Users\...	sklearn	0.776	0.057	0.92	0.08	0.56	

24 matching runs

On peut également afficher des graphiques avec des paramètres d'évaluation du modèle pertinentes :



Les paramètres d'évaluation des différents modèles sont principalement :

1. Precision : La précision détermine que, quand le classifieur déclare que la prédiction est un 1, il a raison à X%.

$$Precision = \frac{vrais\ positifs}{frais\ positifs + faux\ positifs}$$

2. Recall : Ce coefficient détermine le pourcentage de détection des 1 du classifieur.

$$Recall = \frac{vrais\ positifs}{frais\ positifs + faux\ négatifs}$$

3. F1 Score : Ce paramètre prend en compte et la précision et le recall. Plus le score est élevé, plus le modèle est performant.

Le F1 Score se calcule à partir de la formule suivante :

$$F_1 = \frac{2}{recall^{-1} + precision^{-1}} = 2 \frac{precision \cdot recall}{precision + recall} = \frac{2tp}{2tp + fp + fn}$$

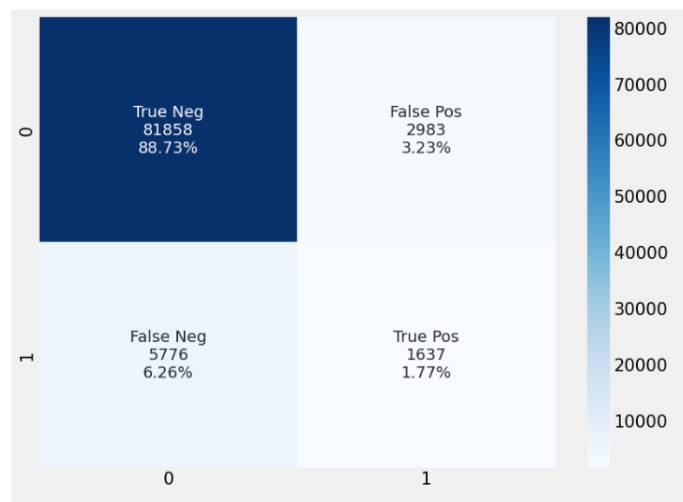
Voici un exemple de CatBoost de scale_pos_weight_3 avec les 3 paramètres d'évaluation :

	precision	recall	f1-score	support
0.0	0.93	0.97	0.95	84841
1.0	0.36	0.21	0.27	7413
accuracy			0.91	92254
macro avg	0.65	0.59	0.61	92254
weighted avg	0.89	0.91	0.90	92254

Ce modèle a une précision moyenne de 65%, un recall de 59% et un F1 Score de 61%. On pourrait traduire ces résultats en disant que notre modèle arrive à trouver 59% des classes 1, mais que, quand il prédit une classe 1, il a raison dans 61% des cas.

4. Analyse de la matrice de confusion :

La matrice de confusion consiste à compter le nombre de fois où des observations de la classe 0 ont été rangées dans la classe 1. Par exemple, si nous voulons connaître le nombre de fois où le classifieur a bien réussi à classer une classe 1, on examinera la cellule à l'intersection de la ligne 1 et de la colonne 0 et 1.



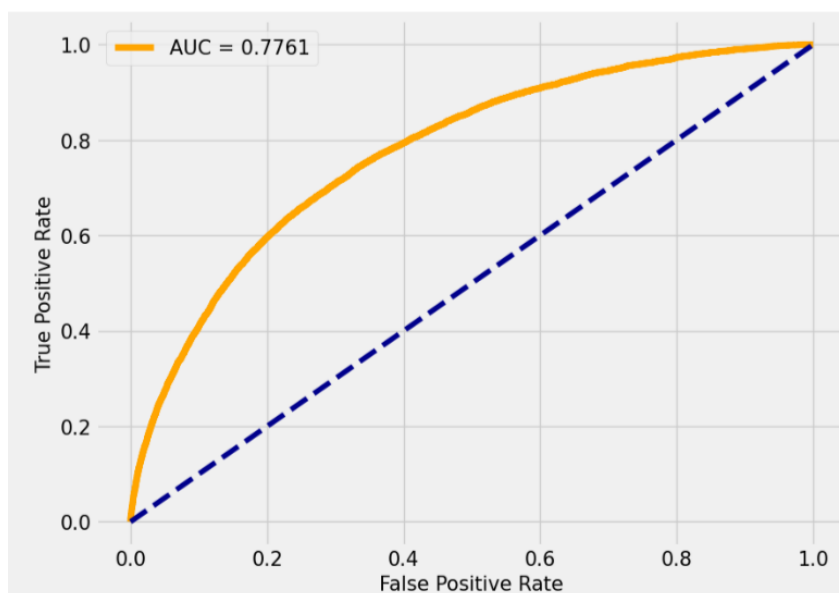
Nous distinguons quatre catégories dans la matrice de confusion :

1. Les True Negatives (TN) : L'intersection de la ligne 0 avec la colonne 0. Ce sont des individus représentant un gain pour l'entreprise.
2. Les False Negatives (FN) : L'intersection de la ligne 1 avec la colonne 0. Ce sont des individus ayant comme valeur réelle 1 mais que le modèle a prédit en 0. Ces individus sont potentiellement un risque pour l'entreprise.
3. Les False Positives (FP) : L'intersection de la ligne 0 avec la colonne 1. Ce sont des individus ayant comme valeur réelle 0 mais que le modèle a prédit en 1. Ces individus pourraient être un risque pour l'entreprise.

4. Les True Positives (TP) : L'intersection de la ligne 1 avec la colonne 1. Ce sont des individus étant à risque pour l'entreprise.

Notre modèle idéal serait de retrouver 100% de TP, car ce sont les individus qui ne remboursent pas leur prêt et donc, d'éventuellement détecter les FN aussi.

5. Score ROC et Score AUC :



La courbe ROC (Receiver Operating Characteristic) est un outil utilisé avec les classifieurs binaires. Elle croise le taux de TP avec le taux de FP.

Sur la figure, la ligne en pointillée représente la courbe ROC d'un classifieur purement aléatoire. Un bon classifieur s'en écarte autant que possible (vers le coin supérieur gauche).

Une autre façon de comparer des classifieurs consiste à mesurer l'aire sous la courbe (Area Under the Curve ou AUC). Un classifieur parfait aurait un score AUC égal à 1, tandis qu'un classifieur purement aléatoire aurait un score AUC de 0.5.

Voici un tableau comparant les AUC Scores des trois modèles évalués lors de ce projet :

	Model	AUC	Accuracy	Precision	Recall	F1	Time
0	CatBoostClassifier	0.775818	0.920166	0.56	0.030217	0.05734	16.268135
1	LGBMClassifier	0.773816	0.919938	0.530752	0.031431	0.059348	11.755588
2	XGBClassifier	0.76713	0.918681	0.449944	0.053959	0.096362	9.122981

Le LightGBM et Le Catboost ont un AUC plus élevé de 0.77. Cependant le Lightboost est beaucoup plus rapide.

3.3. Sélection des variables pertinentes

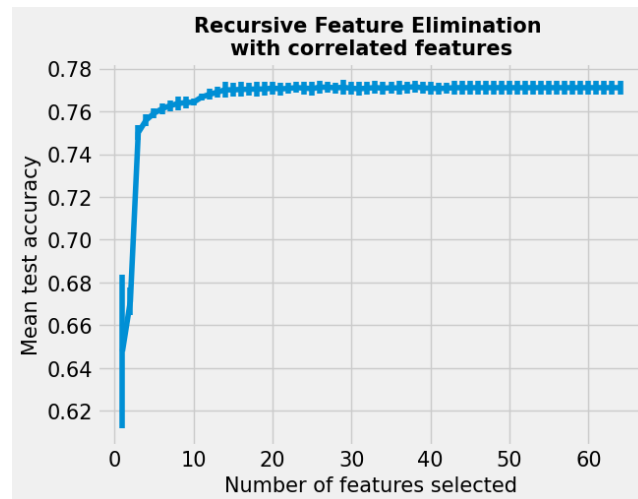
Nous pouvons sélectionner les variables pertinentes avec l'utilisation de la technique d'élimination des caractéristiques récursives avec validation croisée (RFECV). C'est essentiellement une sélection rétrograde des prédicteurs. Elle commence par construire un modèle sur l'ensemble des prédicteurs et calcule un score d'importance pour chaque prédicteur. Le ou les prédicteurs les moins importants sont ensuite supprimés, le modèle est reconstruit et les scores d'importance sont à nouveau calculés. En pratique, l'analyste spécifie le nombre de sous-ensembles de prédicteurs à évaluer ainsi que la taille de chaque sous-ensemble. La taille du sous-ensemble qui optimise les critères de performance est utilisée pour sélectionner les prédicteurs en fonction des classements d'importance. Le sous-

ensemble optimal est ensuite utilisé pour entraîner le modèle final.

À ce stade, nos ensembles de données contiennent 309 features, dont beaucoup peuvent ne pas contenir d'informations utiles. RFECV avec Scikit-learn appliquera une validation croisée pour trouver l'ensemble des features optimal qui maximisera nos performances. Le but est donc d'optimiser la métrique AUC tout en éliminant les features les moins importantes.

```
Optimal number of features : 182
Selected Features: [ True  True  True  True  True  True  True  True  True  True  True  Fal
se
False  True False  True False  True  True  True]
Feature Ranking : [1 1 1 1 1 1 1 1 1 1 7 7 1 7 1 3 1 1 1]
```

RFECV conserve les features avec un Rank 1 > True.



Le seuil optimal de features est après 20 dans le diagramme ci-dessus.

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

4.1. La Fonction coût métier

Le modèle ne permettra pas d'éviter totalement ce risque, à titre d'exemple une erreur de prédiction aura pour conséquence soit un défaut de paiement du client, soit un refus de crédit à un client qui pourrait rembourser sa dette sans aucune défaillance. Les erreurs de prédiction doivent être minimisées, dans cette logique une fonction coût ayant pour objectif de pénaliser les Faux Positifs et les Faux Négatifs a été implémentée.

Ainsi, les pertes d'un crédit en raison d'une mauvaise classification dépendent des probabilités Faux Positifs et Faux Négatifs. L'idée est d'éviter les clients avec un fort risque de défaut. Il est donc nécessaire de pénaliser les FP et FN cités précédemment. Pour réduire ce risque de perte financière, il faut maximiser deux critères Recall et Precision. La Fonction d'optimisation Recall et Precision a une importance plus forte pour le critère Precision. On l'appelle FScore et sa formule est la suivante :

$$Fscore = \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

Avec Beta (β) :

$$Beta = \frac{coef Recall}{coef Precision}$$

L'application de cette métrique métier passe par la quantification de l'importance relative entre Recall et Precision, à savoir Beta (β). Cela revient à estimer le coût moyen d'un défaut, et le coût d'opportunité d'un client refusé par erreur. Cette connaissance métier n'est pas évoquée à ce stade du projet, nous allons donc l'estimer. Cette hypothèse pourra bien entendu être modifiée avec un interlocuteur métier.

- Défaut de paiement 100% du montant du crédit en pertes et autres recouvrements.
- 10% de chance d'obtenir un crédit pour un client lambda qui souhaite emprunter.

L'hypothèse fixée dans le projet est Beta = 10.

4.2. L'algorithme d'optimisation

Choisir les hyperparamètres appropriés est nécessaire pour affiner et booster les performances d'un algorithme d'apprentissage automatique. La métrique utilisée dans le contexte de notre classification binaire sera l'AUC score et le custom score. Grid Search ou Random Search étaient également une alternative possible, à la différence d'Hyperopt une "méthode basique" ne permet pas de traiter un large espace de paramètres, ici très largement privilégié. Avec Hyperopt, on peut facilement analyser notre modèle de Boosting tout en variant les hyperparamètres dans l'espace que nous allons définir ci-dessous.

```
#Parameter space
space = {
    'n_estimators': hp.uniform('n_estimators', 100, 600, 100),
    'learning_rate': hp.uniform('learning_rate', 0.001, 0.03),
    'max_depth': hp.quniform('max_depth', 3, 7, 1),
    'subsample': hp.uniform('subsample', 0.60, 0.95),
    'colsample_bytree': hp.uniform('colsample_bytree', 0.60, 0.95),
    'reg_lambda': hp.uniform('reg_lambda', 1, 20)
}
```

La mise en œuvre de LightGBM est facile, la seule chose compliquée est le réglage des hyperparamètres. LightGBM couvre plus de 100 hyperparamètres. Dans le contexte du projet l'idée est de pouvoir optimiser quelques hyperparamètres via HyperOpt.

Il est nécessaire d'identifier des hyperparamètres pouvant avoir un impact dans l'amélioration de la métrique d'évaluation, on a donc choisi les hyperparamètres suivants :

- `n_estimators` : nombre d'arbres séquentiels.
- `learning_rate` : détermine l'impact de chaque arbre sur le résultat final.
- `max_depth` : profondeur maximale d'un arbre.
- `subsample` : fraction de samples des données train à sélectionner pour chaque arbre.
- `colsample_bytree` : fraction de features à sélectionner pour chaque arbre.

5. Tableau de synthèse des résultats

	Model	AUC	Accuracy	Precision	Recall	F1	Time	
3	CatBoostClassifier	0.777736	0.872103	0.281574	0.381357	0.323956	16.992299	<div style="border: 1px solid black; padding: 5px; display: inline-block;">Avec SMOTE</div>
0	CatBoostClassifier	0.77731	0.906508	0.360626	0.21152	0.266644	16.448041	
4	LGBMClassifier	0.77625	0.865112	0.26886	0.394712	0.319851	10.059992	
1	LGBMClassifier	0.776072	0.905045	0.354252	0.220828	0.272062	9.614727	
2	XGBClassifier	0.763585	0.900872	0.327901	0.222582	0.265167	8.230987	
5	XGBClassifier	0.762459	0.869491	0.266478	0.356131	0.30485	9.089757	

Parmi les trois modèles de gradient boosting le LightBoost est celui qui est le plus performant en termes de temps et de AUC score suivi du XGBoost. Les résultats avec équilibrage des données sont plus satisfaisants.

Après le features selection et l'optimisation de l'algorithme de LightBoost, ainsi qu'en prenant en considération la fonction coût, on choisira l'algorithme optimisé pour l'intégrer à la prédiction.

6. L'interprétabilité globale et locale du modèle

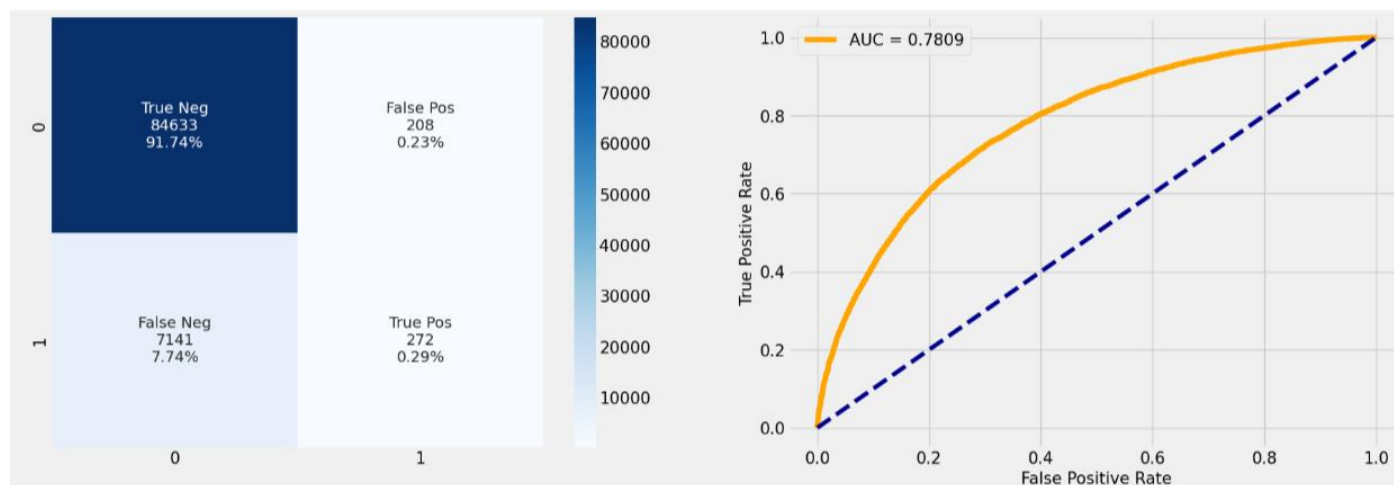
Voici les résultats obtenus avec le LightBoost optimisé :

Accuracy: 0.9203394974743643

Precision:0.5666666666666667

Recall:0.03669229731552678

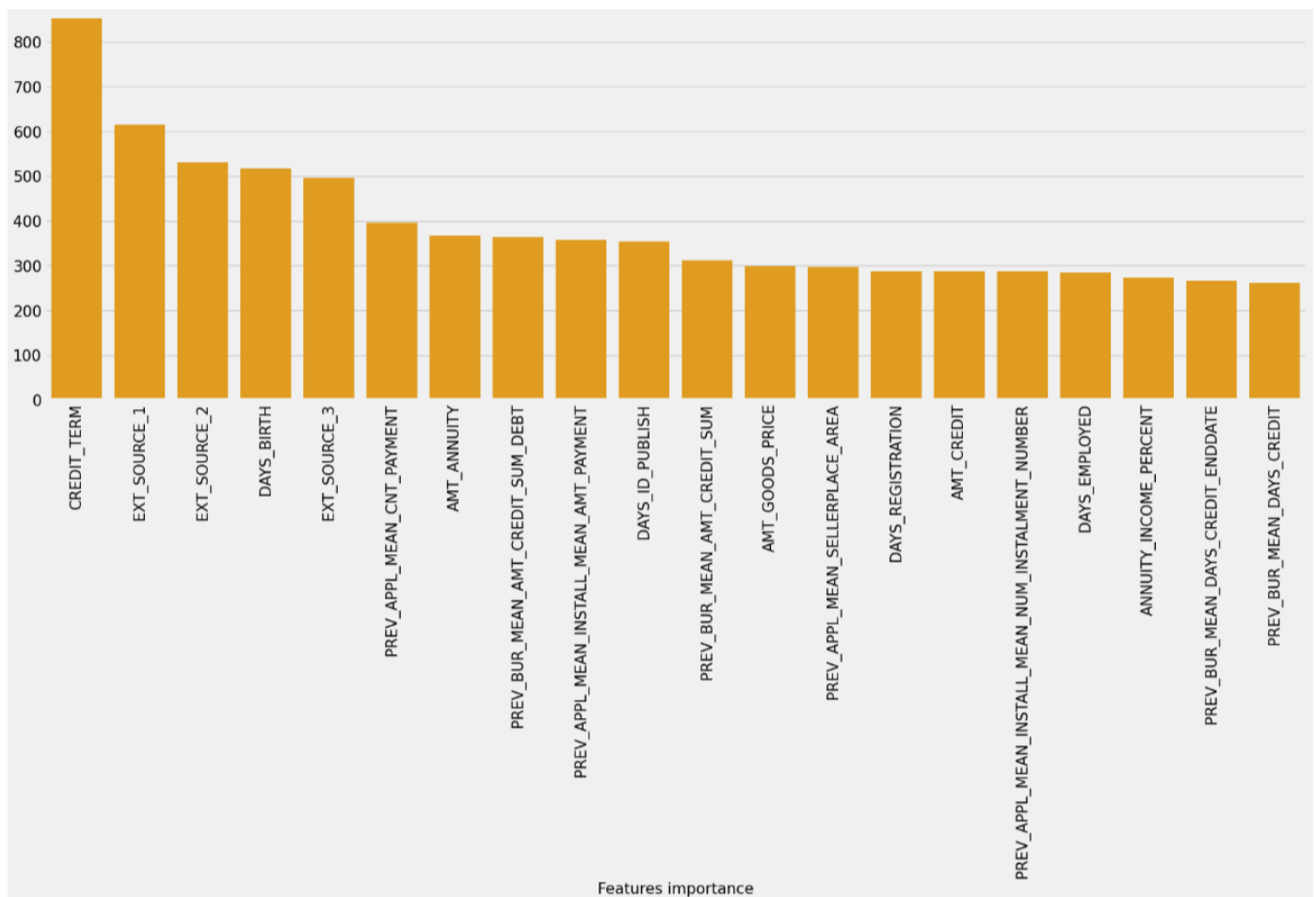
F-1 Score: 0.8869172021298004



Après une baseline faite avec un algorithme simple de régression logistique, l'AUC score avait été estimé ≈ 0.72 avec rééquilibrage (SMOTE) des données. La suite de l'étude a été déroulée vers 3 algorithmes plus complexes de gradient boosting implémentés par LightGbm vs CatBoost vs XGBoost. Nous avons pu démontrer les performances de ces algorithmes par une sélection de features, à l'origine > 300 , après RFECV 182. LightGbm ressort comme étant le plus rapide, le plus performant sur la métrique classique de l'AUC, il a donc été choisi pour l'optimisation des Hyperparamètres (Hyperopt).

La fonction coût permet de pénaliser les erreurs de prédiction qui peuvent coûter cher à l'entreprise. Au final la métrique métier permet de pénaliser légèrement mieux les erreurs du modèle.

L'interprétation désigne l'évaluation globale du processus de prise de décision. Elle vise à représenter l'importance relative de chaque variable. L'idée est donc d'explicitier au mieux le score renvoyé par le modèle. La classe `lightgbm.LGBMClassifier` permet de mieux comprendre le choix et l'importance de des caractéristiques via un attribut `feature_importances_` :



La figure ci-dessous désigne les top 20 features qui influencerait la prédiction.

7. Pipeline de déploiement

Il est nécessaire tout au long de des tests de différents modèles de pouvoir avoir accès à des versions précédentes de notebooks ou de code. [GitHub et Git commits](#) sont des outils qui nous permettent de sauvegarder en ligne des versions précédentes de changements du projet et de mettre à disposition en ligne tous ces changements en cas de collaboration avec des collègues. Le suivi s'effectue via des commits Git de push and pull sur l'espace GitHub. Voici un exemple de l'espace GitHub ainsi qu'un extrait de l'historique des commits effectués :

github.com/keshika-dabidin-audam/projet-7/tree/master

Product Solutions Open Source Pricing

Search or jump to... Sign in Sign up

keshika-dabidin-audam / projet-7 Public

Notifications Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Security Insights

master 3 branches 0 tags Go to file Code

This branch is 39 commits ahead, 1 commit behind main.

File	Commit Message	Commit Hash	Time
Notebook Files	api dashboard and notebooks	e4e4c4ff	last week
app_deploy	Merge branch 'master' of https://github.com/keshika-dabidin-audam/pr...		yesterday
app_local	with sample app_train and app_test		2 days ago
tests flask	api dashboard and notebooks		last week
tests_streamlit	api dashboard and notebooks		last week
Note méthodologique.docx	modification intro doc		yesterday
README.md	Create README.md deployment requirements		last week
~\$te méthodologique.docx	modification intro doc		yesterday
~WRL2889.tmp	modification intro doc		yesterday

README.md

About

Openclassrooms projet 7 2023

Readme Activity 0 stars 1 watching 0 forks Report repository

Releases

No releases published

Packages

No packages published

Commits

master

Commits on Aug 23, 2023

- modification intro doc keshika-dabidin-audam committed yesterday e4e4c4ff <>
- Merge branch 'master' of https://github.com/keshika-dabidin-audam/pro... keshika-dabidin-audam committed yesterday fc22296 <>
- modification fichiers api keshika-dabidin-audam committed yesterday 4ad0caf <>
- Update Profile keshika-dabidin-audam committed yesterday Verified c160455 <>

















Commits on Aug 22, 2023

- with sample app_train and app_test keshika-dabidin-audam committed 2 days ago 35b182a <>

Commits on Aug 18, 2023

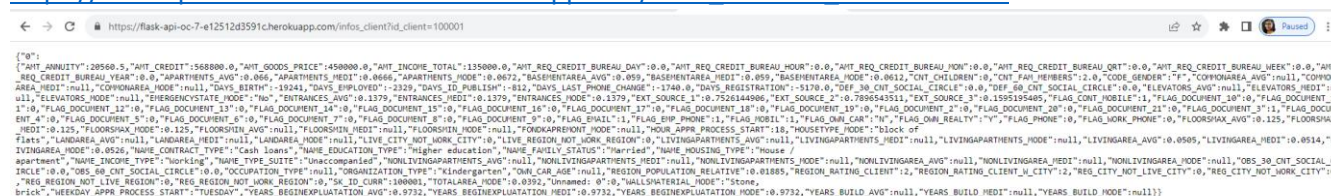
- dashboard added keshika-dabidin-audam committed last week 6b054ee <>
- deplacement dans dossier app deploy et api keshika-dabidin-audam committed last week c1f6c56 <>
- edit in app flask keshika-dabidin-audam committed last week 7c1c4f1 <>
- changes for heroku keshika-dabidin-audam committed last week 1c6172e <>
- only deploy api keshika-dabidin-audam committed last week c2f1428 <>
- Create runtime.txt keshika-dabidin-audam committed last week Verified 414ee1e <>
- Update Profile keshika-dabidin-audam committed last week Verified 414ee1e <>

La plateforme GitHub aide également à déployer le modèle. Dans le cadre de ce projet, Heroku a été utilisé comme principale plateforme de déploiement à partir de Git. Le dossier app-deploy contient les fichiers de l'API de données (créé à partir de Flask) et le Dashboard interactif (créé à partir de Streamlit). On peut également retrouver les versions antérieures des applications sur Heroku :

-   **keshika.dabidin@gmail.com:** Deployed `d379fc83`
Yesterday at 4:58 PM · v34
 -   **keshika.dabidin@gmail.com:** Build succeeded
Yesterday at 4:58 PM · [View build log](#)
 -   **keshika.dabidin@gmail.com:** Deployed `d1c76737`
Yesterday at 4:56 PM · v33 · [Roll back to here](#)
 -   **keshika.dabidin@gmail.com:** Build succeeded
Yesterday at 4:55 PM · [View build log](#)
 -   **keshika.dabidin@gmail.com:** Deployed `bf86cac1`
Yesterday at 4:52 PM · v32 · [Roll back to here](#)
 -   **keshika.dabidin@gmail.com:** Build succeeded
Yesterday at 4:51 PM · [View build log](#)
 -   **keshika.dabidin@gmail.com:** Build failed
Yesterday at 4:48 PM · [View build log](#)
 -   **keshika.dabidin@gmail.com:** Deployed `46e13ee6`
Yesterday at 4:44 PM · v31 · [Roll back to here](#)

1. API : <https://flask-api-oc-7-e12512d3591c.herokuapp.com/>

<https://flask-api-oc-7-e12512d3591c.herokuapp.com/infos> client?id client=100001



...

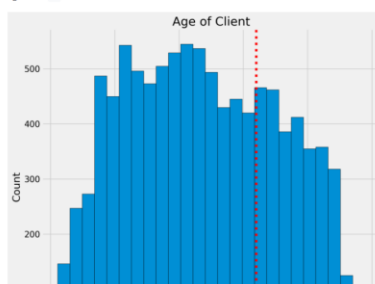
API répondant aux besoins du projet 7 pour le parcours Data Scientist OpenClassRoom

Vous avez sélectionné le client : 100001

☒ Afficher les informations du client?

Statut famille : ** Married **

Age client : 52 ans.



Probabilité de risque de faillite du client :

Request prediction done

94.84 %

Données client :

	AMT_ANNUITY	AMT_CREDIT	AMT_GOODS_PRICE	AMT_INCOME_TOTAL	AMT_REQ_CREDIT_BUREAU
0	20,560.5	568,800	450,000	135,000	

8. Les limites et les améliorations possibles

La modélisation effectuée dans le cadre du projet a été effectuée sur la base d'une hypothèse forte : la définition d'une métrique d'évaluation : le F Score avec Beta fixé suivant certaines hypothèses non confirmées par le métier. L'axe principal d'amélioration serait de définir plus finement la métrique d'évaluation en collaboration avec les équipes métier. Le coefficient Beta pourrait être affiné dans ce sens pour éventuellement améliorer la métrique d'évaluation.

L'espace de recherche HyperOpt permet de larges possibilités, le choix des hyperparamètres est évolutif, la question d'élargissement vers d'autres hyperparamètres peut également permettre d'augmenter les performances actuelles.

Enfin, la partie de traitement préalable du jeu de données a été abordée de façon superficielle en réutilisant un notebook issu de Kaggle qui se base uniquement sur une table du jeu de données. Il y a très probablement l'opportunité d'améliorer la modélisation en utilisant d'autres features issues de données complémentaires fournies, ainsi qu'en créant de nouvelles features en collaboration avec les équipes métier.

9. L'analyse du Data Drift

La dérive des données, également appelée décalage de variable, se produit lorsque la distribution des données d'entrée change au fil du temps. Il s'avère nécessaire donc de suivre cette dérive. Le rapport Data Drift permet de détecter et d'explorer les changements dans les données d'entrée. Nous utiliserons la librairie Evidently pour obtenir des rapports de Data Drift. Voici les différents rapports obtenus à partir de l'analyse du Data Drift :

1. Data Drift Summary

Dataset Drift		
Dataset Drift is NOT detected. Dataset drift detection threshold is 0.5		
312 Columns	3 Drifted Columns	0.00962 Share of Drifted Columns

Dataset Drift définit une règle en plus des résultats des tests statistiques pour les fonctionnalités individuelles. Par défaut, la dérive de l'ensemble de données est détectée si au moins 50 % des entités dérivent. Il y a ici 3 colonnes qui dérivent mais le data drift n'est pas détecté.

2. Data Drift Table

Drift is detected for 0.962% of columns (3 out of 312).

Column	Type	Reference Distribution	Current Distribution	Data Drift	Stat Test	Drift Score ↑
> PREV_APPL_MEAN_DAYS_DECISION	num			Detected	K-S p_value	0.044248
> PREV_APPL_MEAN_DAYS_LAST_DUE_1ST_VERSION	num			Detected	K-S p_value	0.079211
> PREV_APPL_MEAN_DAYS_TERMINATION	num			Detected	K-S p_value	0.113665
> PREV_APPL_MEAN_SELLERPLACE_AREA	num			Not Detected	K-S p_value	0.23649
> PREV_APPL_MEAN_DAYS_LAST_DUE	num			Not Detected	K-S p_value	0.254765
> PREV_APPL_MEAN_INSTALL_MEAN_DAYS_ENTRY_PAYMENT	num			Not Detected	K-S p_value	0.274059

Le tableau montre en premier les entités dérivantes. On peut également choisir de trier les lignes par nom ou type de fonctionnalité.

3. Data Drift by feature



Pour les features numériques, on peut également explorer les valeurs dans un tracé. La ligne vert foncé représente la moyenne, comme on le voit dans l'ensemble de données de référence. La zone verte couvre un écart type par rapport à la moyenne.