Note Méthodologique

Projet 7 Parcours Data Scientist

Pouria Forouzesh - sep 2023

OpenClassrooms- Centrale Supélec

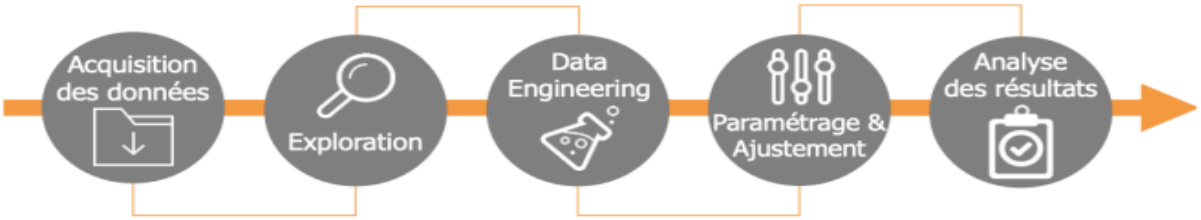
Synthèse

Approche Machine Learning :

Ce projet utilise trois algorithmes principaux : LightGBM, XGBoost et CatBoost. Ils ont été choisis pour leur facilité d'interprétation et leurs bonnes performances antérieures. Tout l'entraînement a été effectué sur Jupyter Notebook.

Étapes du Projet :

Un projet de Machine Learning suit plusieurs étapes clés. Pour ce projet, nous avons adopté une approche méthodique et couramment utilisée dans le déroulement de ces étapes.



Collecte des Données :

La première étape de notre projet était de collecter des données pertinentes. Nous avons choisi des données financières et comportementales de clients sur Kaggle pour former la base de notre apprentissage machine.

Prétraitement des Données :

Avant de faire des prédictions, il est crucial de traiter et de nettoyer les données, car elles peuvent souvent être bruitées ou incomplètes. Utiliser des données brutes peut conduire à des erreurs, d'où la nécessité de les ajuster et de les enrichir après l'étape exploratoire.

Modélisation et Évaluation :

L'objectif majeur est de créer un modèle prédictif en utilisant les données traitées. Les algorithmes mentionnés ont été configurés et optimisés sur cette base de données. Pour évaluer leur efficacité, nous mesurons l'erreur à l'aide de l'AUC (aire sous la courbe ROC). De plus, une fonction coût a été ajoutée pour prendre en compte les erreurs lors de la prise de décision en matière de crédit.

Table des matières

1- Prétraitement des données

Après le prétraitement des données, avant d'entrer dans la phase de modélisation, une étape intermédiaire est essentielle. Le modèle de prédiction fonctionne comme une fonction qui accepte des données et produit une décision. En apprentissage supervisé, où une variable de sortie est ciblée, les données sont généralement divisées en deux parties distinctes :

- **Échantillon d'Apprentissage** : C'est l'échantillon principal utilisé pour former et ajuster le modèle. Les algorithmes apprennent et s'ajustent sur cet échantillon. Pour notre projet, il constitue 80% de l'ensemble des données.

- **Échantillon de Test :** Utilisé pour évaluer la performance du modèle final, l'échantillon de test n'a pas été impliqué dans la phase d'apprentissage, garantissant ainsi son indépendance. Il simule la réception de nouvelles entrées et permet d'évaluer à quel point les prédictions du modèle correspondent aux valeurs réelles. Cet échantillon constitue 20% de l'ensemble total des données et offre une mesure objective de l'erreur du modèle.

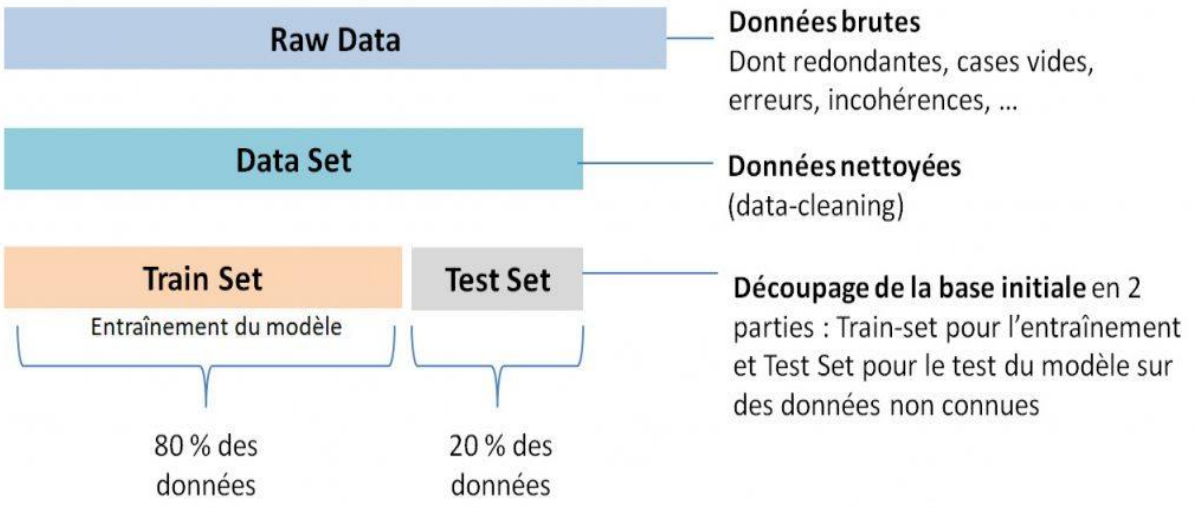


Figure 1 - Découpage des données Train/Test

Importance du Découpage des Données, Séparer correctement les données en échantillons d'apprentissage et de test est crucial dans un projet de Machine Learning. Une mauvaise séparation peut conduire à une surévaluation (over-fitting) où le modèle est trop adapté aux données d'entraînement, ou à une sous-évaluation (under-fitting) où il n'est pas assez adapté. Il est essentiel que le modèle s'ajuste bien, mais pas excessivement, à ses données d'entraînement.

Outil de Découpage des Données, La fonction train_test_split() fournie par Scikit-Learn facilite la division aléatoire des données en échantillons d'apprentissage et de test.

Problème de Classification et Équilibrage des Classes, Nous abordons un problème de classification binaire et il est vital d'avoir une distribution équilibrée des classes 0 et 1 dans les échantillons. L'analyse exploratoire a révélé un déséquilibre marqué : 92% des données appartiennent à la classe 0 et seulement 8% à la classe 1. Pour rectifier ce déséquilibre, nous avons utilisé la technique d'Oversampling, qui ajuste la distribution des classes pour atteindre une répartition plus homogène.

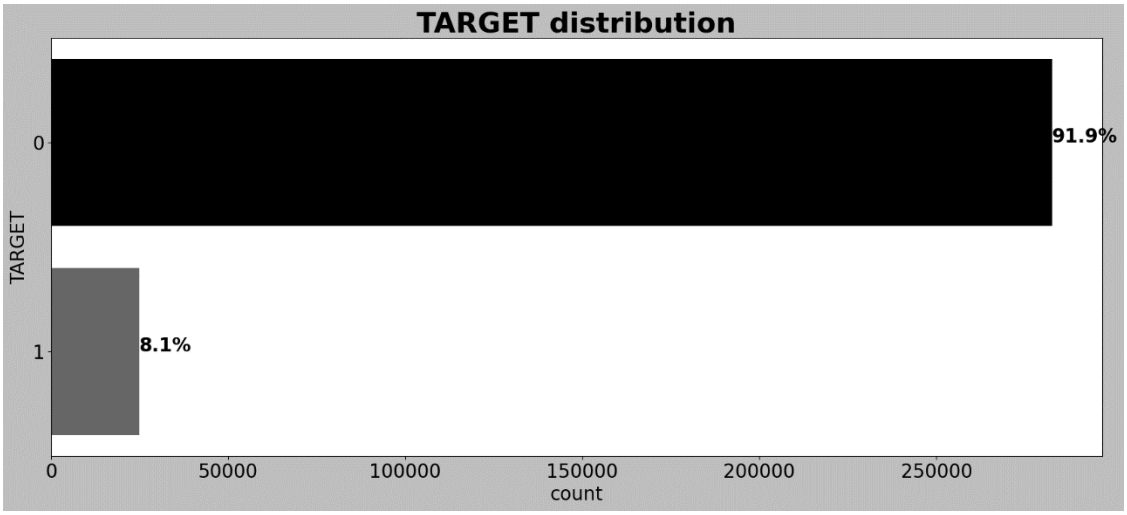


Figure 2 - Déséquilibre des classes

Suréchantillonnage avec SMOTE, Parmi les techniques de suréchantillonnage, SMOTE (Synthetic Minority Oversampling Technique) est particulièrement renommée dans le domaine du Machine Learning. Au lieu de simplement reproduire des données existantes, SMOTE génère des exemples "synthétiques" pour augmenter la classe minoritaire. Cela permet une meilleure équilibration des classes. Utilisation de :

from imblearn.over_sampling (imbalanced-learn Python library SMOTE class)

```
Label 1, Before using SMOTE: 17412
Label 0, Before using SMOTE: 197845
```

```
Label 1, After using SMOTE: 197845
Label 0, After using SMOTE: 197845
```

2 - Modélisation de la probabilité de défaut de paiement du client

2.1 - Notions de base en Machine Learning

Classification pour la Modélisation du Scoring, Le projet vise à établir un modèle de scoring qui prédit la probabilité de faillite d'un client. Cette prédiction est réalisée par le biais d'une classification, où l'objectif est de déterminer la catégorie d'appartenance d'un élément basé sur des exemples antérieurs. Dans notre cas, la classification est binaire, décidant entre l'acceptation ou le refus du crédit pour le client. Cette approche binaire se distingue des problèmes multi-classes où la variable cible peut appartenir à de multiples catégories.

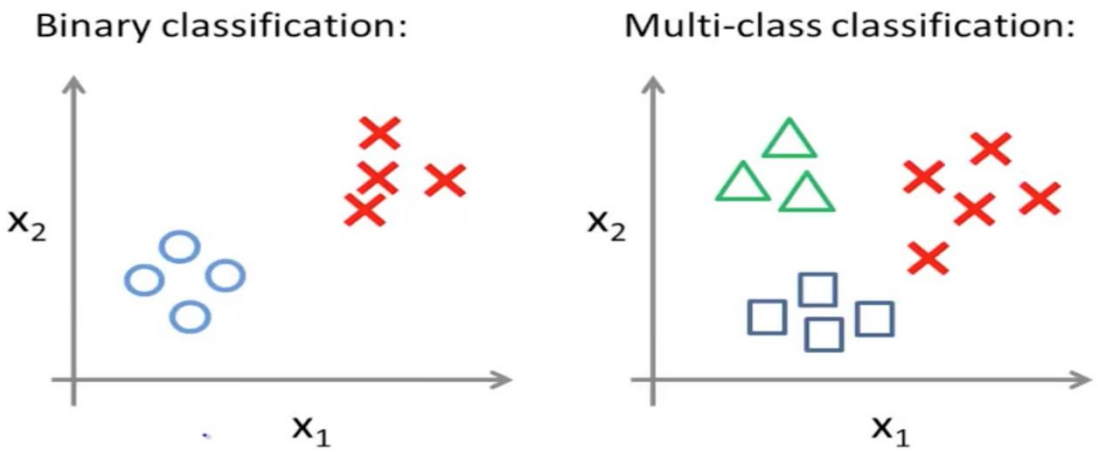


Figure 3 - Représentation des observations

Clustering vs Classification, Une autre technique, distincte de la classification, est le clustering. Au lieu de catégoriser les données selon des classes prédéfinies, le clustering regroupe les données selon leur similarité, sans connaissances préalables. Les groupes se forment à partir de la structure intrinsèque des données. Bien qu'il soit courant dans la réduction de dimensionnalité, le clustering n'a pas été employé dans notre projet.

La technique que nous avons adoptée relève de l'apprentissage supervisé, où l'objectif est de déduire à partir de données connues. En revanche, le clustering est une forme d'apprentissage non-supervisé qui vise à découvrir de nouvelles informations non manifestes initialement. Dans notre contexte, les décisions d'octroi de crédit sont déjà définies, justifiant ainsi l'utilisation de l'apprentissage supervisé.

2.2 - Algorithmes utilisés pour construire le modèle de scoring

Diversité des Approches en Machine Learning, L'avantage majeur du Machine Learning provient de la variété de ses méthodes. En testant de nombreuses approches, on augmente les chances de trouver l'algorithme le plus efficace pour une problématique donnée, comme notre modèle de scoring. Contrairement aux méthodes statistiques traditionnelles qui s'appuient sur certaines hypothèses, le Machine Learning est puissamment prédictif, grâce à son caractère non paramétrique et à la capacité de ses algorithmes d'apprendre directement à partir des données. Dans cette exploration, nous avons évalué trois algorithmes de Gradient Boosting, en comparant leurs performances à une baseline établie par la Régression Logistique, une technique statistique conventionnelle.

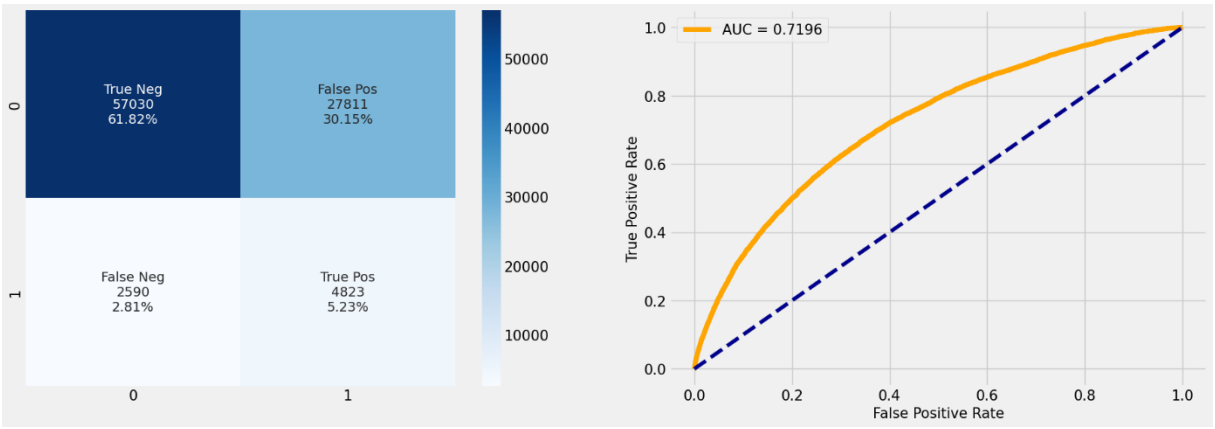


Figure 4 - Résultats Régression logistique : Matrice de confusion, AUC score

	Model	AUC	Accuracy	Precision	Recall	F1	Time
0	CatBoostClassifier	0.77064	0.919592	0.496887	0.053824	0.097128	293.658404
2	XGBClassifier	0.753275	0.918182	0.430341	0.056253	0.099499	20.994493
1	LGBMClassifier	0.75241	0.918865	0.398876	0.019156	0.036556	13.075452

LGBMClassifier reste le "plus performant" selon le couple métrique/temps

2.3 - Sélection des variables pertinentes

Élimination des Caractéristiques Récursives avec Validation Croisée (RFECV) :

La technique RFECV, disponible via from sklearn.feature_selection import RFECV, a été utilisée dans notre projet.

Terminologie :

Récursif : Répétition d'une action ou d'une méthode plusieurs fois pour obtenir un résultat souhaité.

Caractéristique : Une variable ou un attribut spécifique du jeu de données que nous souhaitons étudier ou mesurer.

Validation croisée : Un mécanisme permettant de tester l'efficacité des modèles de Machine Learning. Elle divise les données en plusieurs sous-ensembles, formant différents modèles sur ces sous-ensembles et les testant sur les parties restantes des données.

Ainsi, RFECV est une approche d'optimisation qui vise à sélectionner les caractéristiques les plus pertinentes en éliminant progressivement les moins importantes, tout en utilisant la validation croisée pour garantir la robustesse du modèle.

Élimination des Caractéristiques Récursives (RFE), La méthode RFE est une technique de sélection de prédicteurs par élimination :

Construction Initiale : RFE commence par bâtir un modèle en utilisant tous les prédicteurs disponibles.

Évaluation d'Importance : Après avoir formé le modèle, chaque prédicteur reçoit un score d'importance.

Élimination : Les prédicteurs ayant les scores les plus faibles (donc les moins importants) sont éliminés.

Réitération : Le modèle est reconstruit avec les prédicteurs restants et de nouveaux scores d'importance sont attribués. Cette procédure se répète jusqu'à ce qu'un nombre spécifié de prédicteurs soit atteint.

Réglage : L'analyste détermine le nombre et la taille des sous-ensembles de prédicteurs à évaluer. La meilleure taille de sous-ensemble, qui offre les meilleures performances, est choisie pour la formation du modèle final.

L'essence de RFE est donc d'éliminer progressivement les prédicteurs non essentiels, en se basant sur leur importance, pour améliorer la qualité et la précision du modèle.

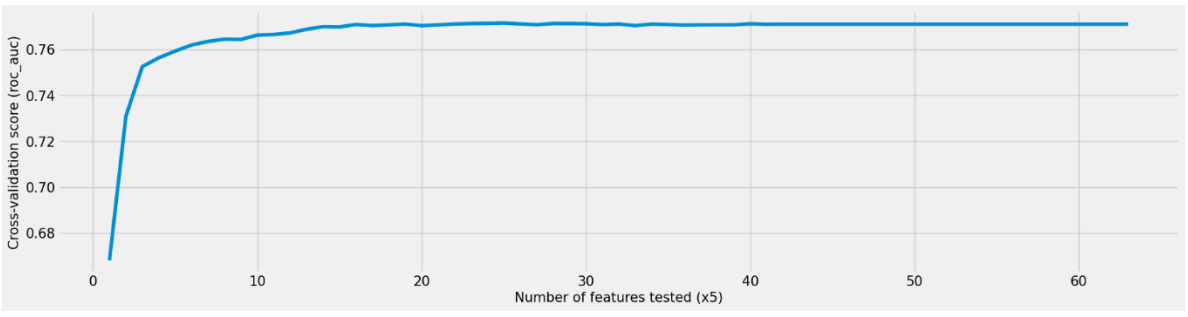


Figure 5 - Précision obtenue avec chaque nombre de features obtenues

2.4 - Fonction coût

Prévention des Défauts de Paiement chez "Prêt à dépenser", l'entreprise "Prêt à dépenser" cherche activement à réduire les défauts de paiement de ses clients pour éviter les pertes financières associées, telles que les coûts de recouvrement. Bien que le modèle de prédiction ne puisse éliminer complètement ce risque, l'impact des erreurs de prédiction peut être conséquent. Une erreur pourrait entraîner soit un défaut de paiement imprévu, soit le rejet d'un client solvable. Afin de minimiser ces erreurs, une fonction coût a été introduite, spécifiquement conçue pour pénaliser à la fois les Faux Positifs et les Faux Négatifs.

Terminologie :

1. **FP (Faux Positifs)** : Situations où le crédit est accordé à un client (prédiction positive) alors qu'il aurait dû être refusé (réalité négative). Il en résulte une perte d'opportunité, car un crédit est refusé à tort à un client solvable.
2. **FN (Faux Négatifs)** : Cas où le crédit est refusé à un client (prédiction négative) qui aurait dû être accepté (réalité positive). La conséquence directe est une perte financière, car un crédit accepté se solde par un défaut de paiement.
3. **TP (Vrais Positifs)** : Situations où la prédiction et la réalité s'alignent positivement, indiquant avec précision que le crédit sera remboursé.
4. **TN (Vrais Négatifs)** : Cas où le modèle prédit correctement que le client ne serait pas en mesure de rembourser le crédit, justifiant un refus.

Gestion du Risque via la Classification, les risques financiers associés à une mauvaise attribution de crédits sont directement liés aux probabilités des Faux Positifs (FP) et des Faux Négatifs (FN). L'objectif principal est de contourner les clients présentant un risque élevé de défaut. Ainsi, il est impératif de minimiser les FP et FN. Pour atténuer ce risque de perte, deux mesures clés, le Rappel (Recall) et la Précision (Precision), doivent être maximisées. Ces critères évaluent la performance du modèle à correctement identifier et classifier les clients selon leur solvabilité.

$$\text{Recall} = \frac{tp}{tp + fn} \qquad \text{Precision} = \frac{tp}{tp + fp}$$

Fonction d'optimisation Recall et Precision avec une importance plus forte pour le critère Precision:

$$\text{Fscore} = \frac{precision.\,recall}{\beta^2.\,precision + recall}$$

Métrique Métier - Importance Relative de Recall et Precision,l'utilisation d'une métrique métier pour évaluer la performance implique de déterminer la balance entre le Rappel (Recall) et la Précision (Precision). La quantification de cette importance relative est représentée par le coefficient Beta (β).

Estimation des Coûts liés aux Erreurs de Classification, Pour appliquer efficacement notre modèle, nous devons quantifier le coût potentiel associé à chaque erreur de classification :

- Coût du Défaut de Paiement : Si un client par défaut ne rembourse pas, cela se traduit par une perte de 30% du montant total du crédit, en raison des frais associés et des tentatives de recouvrement.
- Coût d'Opportunité d'un Refus Erroné : Si un client potentiel est refusé à tort, il existe une probabilité de 10% qu'il aurait obtenu et remboursé un crédit.

Ces estimations permettent d'ajuster le modèle en fonction des implications financières potentielles. Néanmoins, ces chiffres sont basés sur des hypothèses et peuvent être affinés ultérieurement en collaboration avec des experts métier.

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

Application de la Métrique Beta, dans le cadre de ce projet, un coefficient Beta (β) de 3 a été défini. Ceci a été déterminé après des tests pour évaluer l'adéquation et la cohérence des scores obtenus.

Interprétation du Score de Risque de Défaillance, le score évalué indique le risque de défaillance d'un client. Un score de 0% signifie un risque très faible que le client fasse défaut, tandis qu'un score de 100% indique un risque très élevé de défaillance. Ce gradient permet de prendre des décisions éclairées sur l'octroi de crédits basés sur le risque associé à chaque client.

2.5 - Algorithme d'optimisation HyperOpt

Optimisation des Hyperparamètres via Hyperopt, une technique avancée pour l'ajustement et l'optimisation des fonctions et des hyperparamètres dans les processus de Machine Learning est l'utilisation de l'optimisation bayésienne. Le module hyperopt offre des outils pour cette approche. Avec fmin, tpe, hp, STATUS_OK, et Trials issus de hyperopt, on peut chercher de manière efficace les meilleurs hyperparamètres pour un modèle donné. L'optimisation bayésienne vise à identifier la meilleure configuration d'hyperparamètres en se basant sur les performances passées, plutôt que par une recherche aléatoire ou une recherche sur grille.

Recherche d'Hyperparamètres Optimaux, la détermination de la meilleure configuration d'hyperparamètres ne devrait pas reposer uniquement sur l'intuition ou les expériences antérieures. Il est essentiel de se baser sur des méthodes systématiques qui assurent une optimalité robuste. Deux approches principales sont souvent utilisées pour cela :

1- **Recherches Exhaustives** : Ces méthodes, comme le Grid Search ou le Random Search, explorent un ensemble prédéfini d'hyperparamètres pour identifier la meilleure combinaison possible.
2- **Optimisation Bayésienne** : Cette approche, contrairement aux recherches exhaustives, utilise des informations sur les essais précédents pour choisir les combinaisons d'hyperparamètres qui sont les plus susceptibles d'améliorer les performances du modèle.

La sélection de l'approche dépend des besoins spécifiques du projet et des ressources disponibles.

Optimisation Bayésienne avec HyperOpt, l'optimisation bayésienne se base sur un modèle probabiliste qui estime P(score | configuration), où "score" est la performance du modèle pour une certaine "configuration" d'hyperparamètres. Le processus met continuellement à jour ce modèle en examinant un historique de scores et de configurations, dans le but d'optimiser le score pour de nouvelles configurations.

HyperOpt s'inscrit dans cette démarche d'optimisation bayésienne en introduisant certaines variations. Ces variations peuvent concerner la manière dont les échantillons sont tirés, comment l'espace de recherche est défini ou réduit, ainsi que les algorithmes utilisés pour maximiser le modèle probabiliste. L'objectif d'HyperOpt est d'améliorer l'efficacité de l'optimisation bayésienne, offrant ainsi une meilleure exploration de l'espace des hyperparamètres et une sélection plus précise des configurations optimales.

Utilisation de base d'HyperOpt, HyperOpt est un outil d'optimisation qui nécessite quatre éléments essentiels pour une mise en œuvre de base :

- **Fonction à optimiser** : C'est la fonction sur laquelle l'algorithme travaillera pour trouver le meilleur ensemble d'hyperparamètres. Elle est généralement construite pour évaluer la performance du modèle sur un ensemble spécifique d'hyperparamètres.
- **Espace de recherche** : C'est l'ensemble des hyperparamètres potentiels que l'algorithme explorera. L'espace délimite les possibles valeurs ou distributions pour chaque hyperparamètre.
- **Algorithme d'optimisation** : C'est l'algorithme spécifique qui sera utilisé pour naviguer dans l'espace de recherche. HyperOpt offre principalement 'tpe.suggest' pour l'optimisation bayésienne.
- **Nombre d'itérations** : C'est le nombre de fois que l'algorithme évaluera la fonction à optimiser. Plus il y a d'itérations, plus l'algorithme a de chances de trouver le meilleur ensemble d'hyperparamètres, mais cela augmente aussi le temps de calcul.

Dans ce contexte, StratifiedKFold(5) a été employé. Cela signifie que la validation croisée stratifiée a été effectuée avec 5 subdivisions (ou "folds") pour identifier la meilleure itération possible et assurer une distribution homogène des classes dans chaque fold. Cette méthode aide à obtenir des estimations plus robustes et fiables de la performance du modèle.

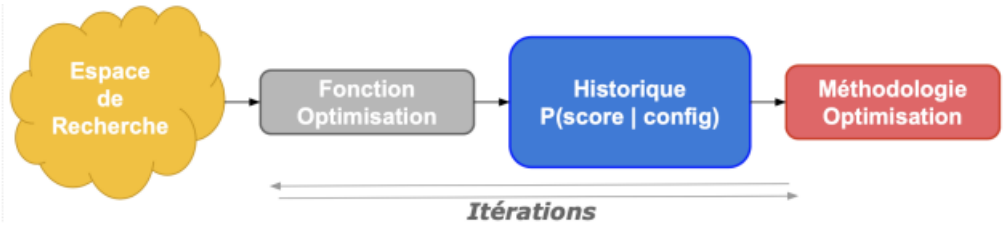


Figure 6 - Schéma représentatif de HyperOpt

Optimisation d'Hyperparamètres pour LightGBM avec HyperOpt, bien que l'implémentation de LightGBM soit relativement simple, le véritable défi réside dans le réglage de ses hyperparamètres. LightGBM offre une vaste gamme d'options avec plus de 100 hyperparamètres possibles. Ces hyperparamètres influencent la performance, la vitesse et la robustesse du modèle.

Dans le cadre de ce projet, l'objectif n'est pas de se plonger dans chaque hyperparamètre, mais de sélectionner et d'optimiser un sous-ensemble d'entre eux pour obtenir les meilleures performances possibles. L'outil choisi pour cette tâche d'optimisation est HyperOpt. Grâce à sa capacité à naviguer efficacement dans l'espace des hyperparamètres via l'optimisation bayésienne, HyperOpt peut aider à identifier rapidement et efficacement les meilleures configurations pour LightGBM.

En amont, il est nécessaire d'identifier des hyperparamètres pouvant avoir un impact dans l'amélioration de la métrique d'évaluation.

- `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 d'observations à sélectionner pour chaque arbre.
- `colsample_bytree` : fraction d'observations à sélectionner pour chaque arbre

```
#Parameter space
space = {
    'n_estimators': hp.quniform('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)
    }
```

```
%%time
best = fmin(fn=objective, space=space, max_evals=30,  rstate=np.random.seed(1), algo=tpe.suggest)
```

```
100%|██████████| 30/30 [27:35<00:00, 55.20s/trial, best loss: 0.9769959069739819]
CPU times: total: 2h 44min 5s
Wall time: 27min 36s
```

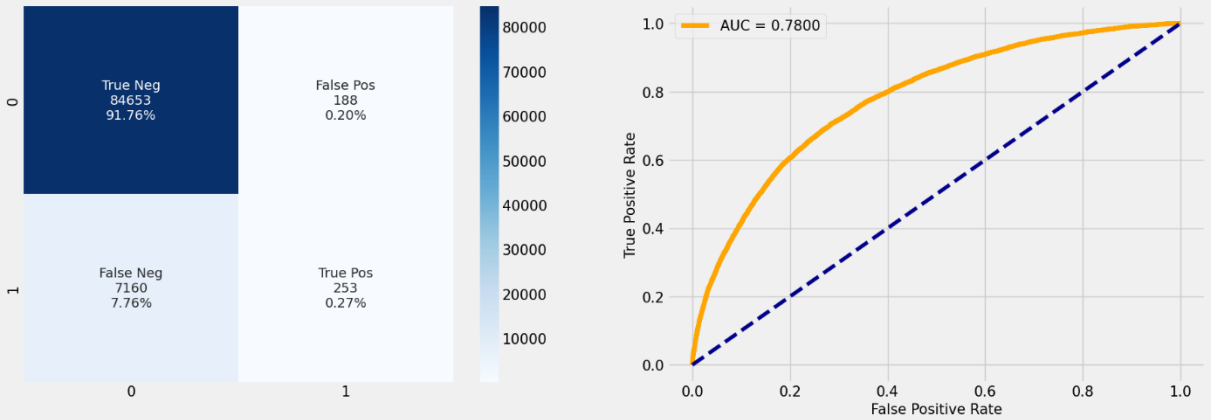


Figure 7 - Résultats obtenus avec le modèle optimisé

3 - Interprétabilité du modèle

Interprétabilité pour les Professionnels du Crédit, l'interprétabilité d'un modèle est d'une importance cruciale, surtout dans des contextes tels que l'octroi de crédits où les décisions prises ont un impact direct sur les individus et les entreprises. Dans ce scénario, le modèle ne s'adresse pas uniquement aux data scientists, mais surtout aux professionnels du crédit, comme les chargés de clientèle.

Ces professionnels ont besoin d'outils qui, non seulement prédisent avec précision, mais aussi expliquent clairement les raisons de ces prédictions. Lorsqu'ils discutent avec un client, ils devraient être en mesure d'expliquer la décision du modèle de manière transparente et compréhensible.

Le terme "interprétation" dans ce contexte fait référence à une compréhension approfondie de la manière dont le modèle prend ses décisions. Plus précisément, il s'agit de comprendre l'influence de chaque variable sur la prédiction finale.

Avec LightGBM, cela devient plus accessible. La classe 'lightgbm.LGBMClassifier' offre un attribut feature_importances_ qui donne un aperçu de l'importance relative de chaque caractéristique dans le modèle. Cela permet non seulement de comprendre les moteurs clés derrière une prédiction, mais aussi de les communiquer efficacement aux parties prenantes.

4 - Limites et améliorations

Améliorations Potentielles et Hypothèses sous-jacentes, les résultats actuels du modèle sont basés sur des suppositions qui n'ont pas encore été validées par les experts métier. Plus précisément, le coefficient Beta, qui joue un rôle crucial dans l'évaluation de notre modèle, est basé sur une hypothèse qui pourrait nécessiter des ajustements.

L'outil HyperOpt offre une flexibilité considérable dans l'optimisation des hyperparamètres. Bien que le choix actuel des hyperparamètres ait été déterminé selon certains critères, il existe toujours une possibilité d'explorer davantage. En élargissant l'espace de recherche ou en considérant d'autres hyperparamètres pertinents, on pourrait potentiellement améliorer davantage les performances du modèle.

En résumé, bien que le modèle actuel fournisse des résultats prometteurs, des ajustements basés sur des retours d'experts métier et une exploration approfondie des hyperparamètres pourraient mener à des améliorations supplémentaires.

5 - Analyse du datadrift

Une Problématique Essentielle, Le comportement des données, sur lesquelles un modèle de Machine Learning est formé, peut évoluer avec le temps. Ces changements peuvent être influencés par divers événements externes ou tendances internes. L'exemple de l'impact de la crise du COVID-19 sur les habitudes de voyage en avion illustre parfaitement cette situation. Les modèles qui se basaient sur les habitudes de voyage pré-COVID peuvent ne plus être pertinents dans le contexte post-COVID.

Ce phénomène est couramment appelé "Data Drift" ou dérive des données. Il s'agit d'un décalage entre les données sur lesquelles le modèle a été formé et les nouvelles données sur lesquelles il est appliqué. Si cette dérive n'est pas détectée et gérée, la performance du modèle peut se dégrader considérablement.

Dans de tels scénarios, il est crucial de surveiller régulièrement la performance du modèle en temps réel. Si une dérive est détectée, le modèle doit être réentraîné avec des données plus récentes et pertinentes. Cette démarche assure que le modèle reste fiable, précis et pertinent pour les décisions qu'il est censé soutenir.

Méthodologie d'Analyse du Data Drift :

Objectif, Analyser la dérive des données entre deux ensembles de données de nature similaire pour déceler toute variation dans leurs distributions.

Démarche :

Analyse de la Feature Cible, Pour évaluer l'impact du data drift sur les variables d'intérêt (features cibles), celles-ci peuvent être ajoutées aux colonnes à comparer :

- Si les labels cibles sont disponibles dans les deux jeux de données, ils doivent être inclus dans l'analyse.
- Si ce n'est pas le cas, une prédiction de la feature cible devrait être effectuée pour les deux ensembles de données afin de permettre une comparaison pertinente.

Gestion de Grands Datasets, Pour des jeux de données volumineux qui requièrent une durée d'analyse excessive :

- **Réduction des Features**, Si le nombre de variables est conséquent, envisagez de se concentrer uniquement sur les features les plus influentes. Cela peut être accompli en utilisant des méthodes telles que feature_importance_ ou des outils comme SHAP pour évaluer l'importance relative de chaque feature.

- **Échantillonnage**, Si le nombre d'observations est trop élevé, on peut envisager d'échantillonner les jeux de données de manière à obtenir un sous-ensemble représentatif mais plus gérable en termes de taille.

L'analyse du data drift est essentielle pour maintenir la fiabilité des modèles de Machine Learning au fil du temps. Une méthode systématique d'identification et de gestion de la dérive garantit que le modèle reste pertinent et précis face à des données en constante évolution.

Solution d'Analyse du Data Drift, Utilisation de la Librairie Evidently : Pour aborder le problème du data drift, nous avons choisi d'utiliser la librairie Evidently. Cette librairie offre une gamme d'outils pour l'analyse et le monitoring des performances des modèles de Machine Learning.

Caractéristiques principales :

Configuration DataDriftPreset, Evidently possède une configuration prédéfinie appelée DataDriftPreset. Elle est conçue spécifiquement pour détecter et analyser la dérive des données entre les datasets de référence et les nouveaux datasets. Elle compare automatiquement les distributions de caractéristiques dans les deux jeux de données pour identifier toute variation significative.

Scores Adaptatifs, L'un des atouts majeurs d'Evidently est sa capacité à adapter ses scores en fonction de la nature des datasets. Que les données soient numériques, catégorielles ou mixtes, la librairie peut calculer des métriques pertinentes pour chaque type.

Visualisation, Evidently offre également des visualisations intuitives, permettant aux utilisateurs de voir graphiquement où et comment les distributions des features ont changé.

En utilisant Evidently, nous avons un outil robuste et efficace pour surveiller et analyser le data drift. Il fournit non seulement des métriques quantitatives mais également des visualisations qualitatives pour une meilleure compréhension des changements survenus. Cette approche proactive permet de garantir que le modèle reste performant et précis malgré les évolutions potentielles des données.

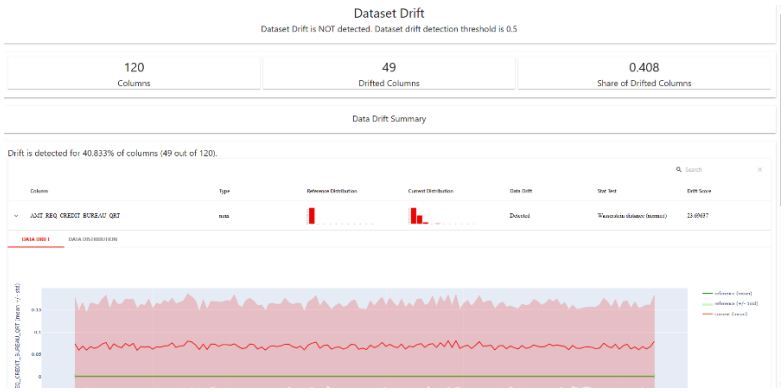


Figure 8 - Résultats obtenus pour datadrift