01 SEPTEMBRE 2023

NOTE METHODOLOGIQUE

PROJET 7

XAVIER PARISOT

Table des matières

La méthodologie d'entraînement du modèle

Préparation des données

Avant de procéder à l'entraînement des modèles, les données sont préparées et divisées en ensembles d'entraînement et de validation. Le découpage est effectué de manière stratifiée pour s'assurer que la distribution des classes dans les ensembles d'entraînement et de validation reflète celle de l'ensemble de données complet.

Rééquilibrage des classes

Étant donné que les données sont déséquilibrées, la technique de suréchantillonnage SMOTE (Synthetic Minority Over-sampling Technique) est utilisée pour équilibrer les classes. Cette étape est intégrée dans un pipeline d'entraînement pour chaque modèle.

Sélection du modèle et optimisation des hyperparamètres

Plusieurs algorithmes de classification sont explorés :

- SVM linéaire (LinearSVC)
- Forêt aléatoire (Random Forest)
- XGBoost
- LightGBM

Pour chaque algorithme, une recherche d'hyperparamètres est effectuée en utilisant une validation croisée stratifiée à 3 plis. Le critère d'évaluation est la F2.

Entraînement du modèle

Chaque modèle est entraîné en utilisant les meilleures combinaisons d'hyperparamètres trouvées. L'entraînement est suivi par MLflow, qui enregistre les hyperparamètres, le temps d'entraînement, et les métriques de performance pour chaque modèle. Les importances des caractéristiques sont également enregistrées lorsque cela est possible.

Évaluation du modèle

Les modèles sont évalués sur un ensemble de validation à l'aide de plusieurs métriques, dont la précision, le rappel, le score F2 et l'AUC-ROC. Cela permet de tenir compte à la fois de la performance du modèle et de son aptitude à gérer les classes déséquilibrées.

Suivi et enregistrement

Toutes les informations, y compris les métriques, les paramètres et les importances des caractéristiques, sont enregistrées dans MLflow pour un suivi facile et une comparaison ultérieure des modèles.

Caractéristiques du modèle

Importance des caractéristiques : Les importances des caractéristiques sont calculées pour chaque modèle (lorsque cela est possible) et enregistrées. Cela fournit des indications sur les caractéristiques qui sont les plus informatives pour la tâche de classification.

Sauvegarde du modèle : Le modèle est sauvegardé sous forme d'artefact dans MLflow, permettant ainsi une réutilisation facile pour les prédictions futures ou pour des analyses plus approfondies.

Cette méthodologie offre une approche systématique et rigoureuse pour l'entraînement et l'évaluation de plusieurs modèles de classification sur un problème avec des classes déséquilibrées. Elle utilise des techniques de rééquilibrage des classes, optimise les hyperparamètres et évalue les modèles sur plusieurs métriques pour fournir une vue complète des performances.

Le traitement du déséquilibre des classes

Problématique

Le déséquilibre des classes est un défi majeur en apprentissage supervisé, en particulier dans les tâches de classification. Dans un contexte déséquilibré, la classe minoritaire est souvent celle qui présente le plus d'intérêt (par exemple, détection de fraudes, diagnostics médicaux), mais elle est également celle qui est la moins représentée dans les données. Les algorithmes de machine learning ont tendance à être biaisés en faveur de la classe majoritaire, ce qui peut entraîner des performances médiocres lors de la prédiction de la classe minoritaire.

Solution Adoptée : SMOTE (Synthetic Minority Over-sampling Technique)

Pour traiter ce déséquilibre, la technique de suréchantillonnage SMOTE est utilisée. SMOTE crée des échantillons synthétiques de la classe minoritaire en identifiant les k plus proches voisins d'un point de la classe minoritaire et en créant des points aléatoires entre le point et ses voisins. Cette technique permet d'équilibrer la classe minoritaire en augmentant sa taille jusqu'à ce qu'elle soit comparable à celle de la classe majoritaire.

Intégration dans le Pipeline d'Entraînement

Le suréchantillonnage SMOTE est intégré dans le pipeline d'entraînement pour chaque modèle de machine learning testé. Cela garantit que le rééquilibrage est effectué lors de chaque itération de la validation croisée, évitant ainsi tout risque de surapprentissage sur la classe minoritaire.

Validation Croisée Stratifiée

En plus de SMOTE, une validation croisée stratifiée est utilisée lors de la recherche d'hyperparamètres pour garantir que chaque "fold" de la validation croisée contient une répartition des classes similaire à celle de l'ensemble de données complet.

Évaluation des Performances

La performance des modèles est évaluée en utilisant plusieurs métriques qui prennent en compte le déséquilibre des classes, notamment le rappel, le score F1 et l'AUC-ROC. Le rappel est particulièrement important car il mesure la capacité du modèle à identifier correctement la classe minoritaire.

Le traitement du déséquilibre des classes est crucial pour améliorer les performances du modèle sur la classe minoritaire, qui est souvent la plus intéressante dans des contextes comme la détection de fraude ou le diagnostic médical. L'approche adoptée ici, combinant SMOTE et validation croisée stratifiée, offre une méthode robuste pour gérer le déséquilibre des classes tout en évitant le surapprentissage.

Class Weights

J’utilise le paramètre class_weight dans lgb.LGBMClassifier pour donner différents poids aux classes lors de l'entraînement du modèle. Les poids sont définis comme {0: 1, 1: score_train}. Cela signifie que la classe 0 a un poids de 1, tandis que la classe 1 a un poids de score_train.

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

La Fonction Coût Métier (Custom Score)

La fonction coût métier, également connue comme le score personnalisé, est une métrique conçue pour aligner étroitement l'évaluation du modèle sur les objectifs métier de l'organisation. Dans notre cas, la fonction est définie comme :

$$Score = \frac{5 * FN + FP}{FN}$$

où FN est le nombre de faux négatifs et FP est le nombre de faux positifs. Cette fonction attribue une pénalité 5 fois plus importante aux faux négatifs qu'aux faux positifs, ce qui est probablement en ligne avec un objectif métier où les faux négatifs ont des conséquences plus graves.

La pénalité est de 5 et non de 10 suite à l'utilisation de class_weight dans le pipeline, où la classe 1 (classe négative) a un poids égal à score_train, soit de 5. Par l'utilisation du class_weight nous focalisons le Custom Score sur la classe 1. Nous réappliquerons le Custom Score plus tard. Le poids de 10 serait alors beaucoup trop impactant.

L'Algorithme d'Optimisation (Grid Search avec LGBMClassifier)

L'optimisation des paramètres du modèle est réalisée via une recherche sur grille (GridSearchCV) appliquée au modèle LightGBM (LGBMClassifier). Les paramètres optimaux sont choisis en fonction d'une métrique d'évaluation.
De plus, nous utilisons un pipeline qui intègre SMOTE pour le suréchantillonnage de la classe minoritaire, ce qui aide à atténuer le déséquilibre de classe. Le modèle utilise également des pondérations de classe basées sur le score personnalisé calculé, ce qui aligne davantage le modèle sur les objectifs métier.

La Métrique d'Évaluation

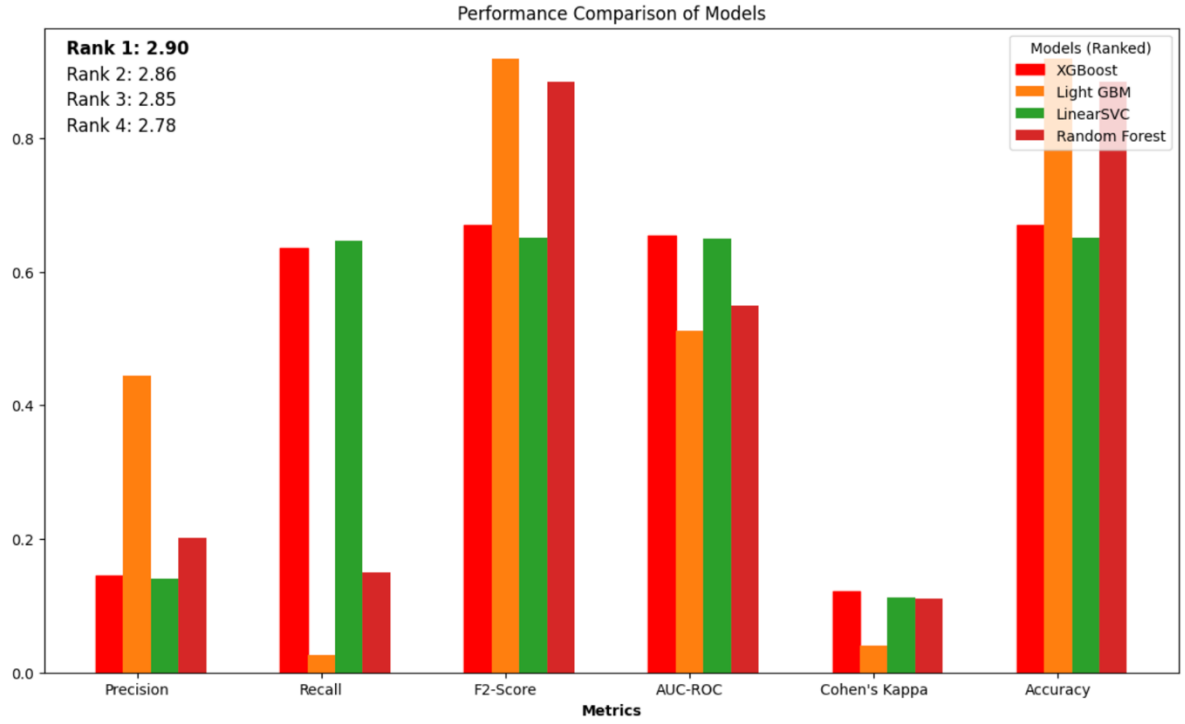
Bien que le modèle soit optimisé pour l'accuracy, le suivi des performances inclut également le score personnalisé, qui est central pour les objectifs métier. Ce score personnalisé est utilisé comme pondération de classe dans le modèle, et est également enregistré dans MLflow pour un suivi efficace et une éventuelle réutilisation.

En résumé, le workflow intègre une fonction coût métier sous la forme d'un score personnalisé pour aligner le modèle sur des objectifs métiers spécifiques. L'optimisation est réalisée via une recherche sur grille avec LightGBM, en tenant compte du score personnalisé pour pondérer les classes. Les performances sont suivies non seulement en termes de métriques standard comme l'accuracy, le rappel, et la précision, mais aussi en termes du score métier personnalisé, ce qui offre une évaluation plus complète alignée sur les objectifs métier.

Tableau de synthèse des résultats

Voici un tableau de synthèse qui résume les métriques de performance pour chaque modèle

	LinearSVC	RandomForest	XGBoost	Light GBM
Précision	0,1398	0,2020	0,1455	0,4448
Recall	0,6468	0,1492	0,6358	0,0260
F2-Score	0,6505	0,8838	0,6695	0,9188
AUC-ROC	0,6488	0,5487	0,6541	0,5116
Accuracy	0,6505	0,8838	0,6695	0,9188
Cohen's Kappa	0,1121	0,1107	0,1215	0,0406
Ranking	2,3418	1,6881	1,9325	1,9537




```
[('XGBoost', 2.895748041150186),
 ('Light GBM', 2.86062471506434),
 ('LinearSVC', 2.848640830353826),
 ('Random Forest', 2.778273872720025)]
```

Observations

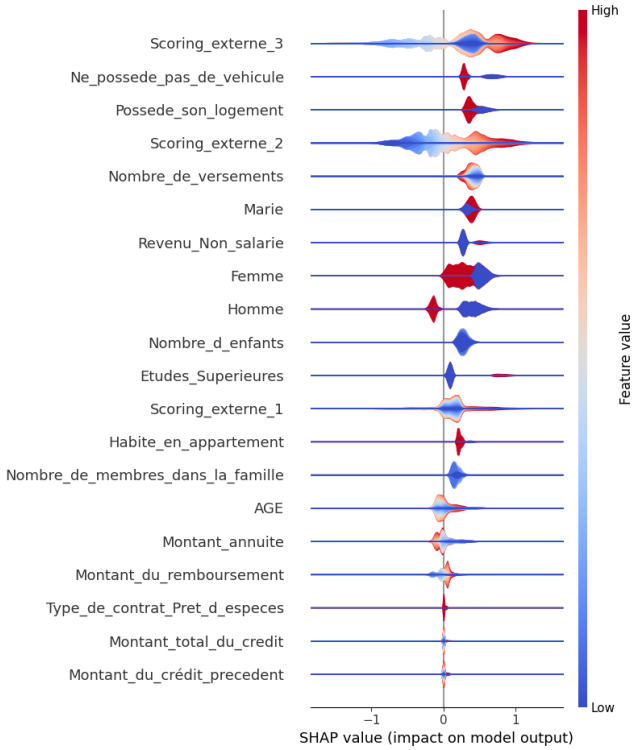
- **LinearSVC :** Ce modèle a un rappel élevé mais une précision très faible. Cela signifie qu'il est bon pour identifier les vrais positifs mais génère beaucoup de faux positifs.
- **Random Forest :** Ce modèle a une précision et un rappel relativement bas, mais une excellente exactitude.
- **XGBoost :** Ce modèle a un rappel élevé mais une précision faible, similaire à LinearSVC.
- **Light GBM :** Ce modèle a une excellente précision mais un très faible rappel. Son score F2 est le plus élevé, indiquant un bon équilibre entre rappel et précision.

Conclusion :

- L'exactitude est la plus importante, Light GBM est donc le meilleur modèle.

L’interprétabilité globale et locale du modèle

L'interprétabilité d'un modèle de machine learning est cruciale pour comprendre non seulement comment le modèle prend ses décisions, mais aussi pour gagner la confiance des parties prenantes et des utilisateurs finaux. L'interprétabilité peut être abordée à deux niveaux : global et local.

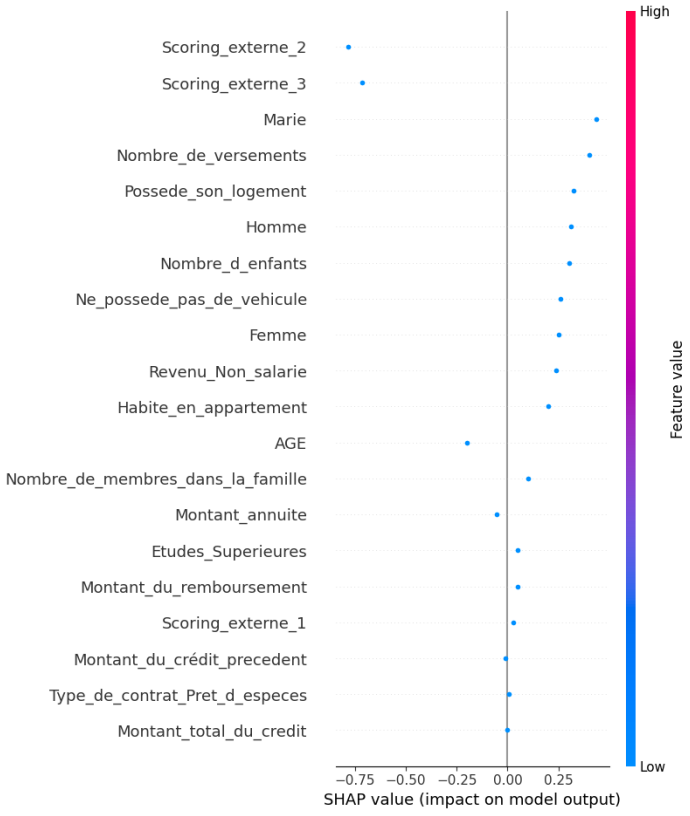


L’interprétabilité Globale

L'interprétabilité globale se réfère à notre compréhension de la manière dont le modèle prend des décisions sur l'ensemble du jeu de données. Dans les arbres de décision, par exemple, cette compréhension est intuitive parce que le modèle est essentiellement une série de questions posées sur les caractéristiques des données. Pour des modèles complexes comme LGBM, des techniques comme l'importance des caractéristiques peuvent être utiles. L'importance des caractéristiques quantifie la contribution de chaque caractéristique à la prédiction du modèle, permettant ainsi de comprendre quels facteurs sont les plus influents.

L’Interprétabilité Locale

L'interprétabilité locale concerne la justification des prédictions individuelles faites par le modèle. Cela est souvent réalisé via des techniques comme LIME (Local Interpretable Model-agnostic Explanations) ou SHAP (SHapley Additive exPlanations). Ces techniques construisent un modèle simple autour d'une prédiction spécifique pour expliquer comment le modèle complexe arrive à cette conclusion. Par exemple, si un modèle de classification prédit qu'un e-mail est un spam, une explication locale pourrait indiquer que la présence du mot "lotterie" dans le texte a été le principal contributeur à cette prédiction.



Les limites et les améliorations possibles

Limites

Mon apprentissage tardif de python (Décembre 2022) et ma méconnaissance des autres langages ainsi que leur logique a été pour moi un réel handicap et une perte de temps en débogage, et recherche de solution (l'api et le dashboard). Les api mêlent plusieurs langages, python et html, css… J'ai quasiment appris à 0 ces langages. Je savais exactement ce que je désirais, ce fut pour moi une chance.

Git et GitHub furent aussi un point d'achoppement, je me suis très vite retrouvé avec des ennuis lors des commit et des push. J'ai eu des déconnections lors de mes push et cela a causé des ennuis au début, entre autres des pertes de code, voire un notebook effacé. Time Machine (sous mac) m'a sauvé plus d'une fois. J'ai réussi à aller au-delà et à trouver les bonnes commandes afin de faire mes commit et mes push.

J'ai compris que la data engineering n'est pas le domaine dans lequel je souhaite évoluer professionnellement…

Les améliorations possibles

Je pense apprendre au plus vite un autre langage de programmation, de type java script, et travailler un peu plus le Html.

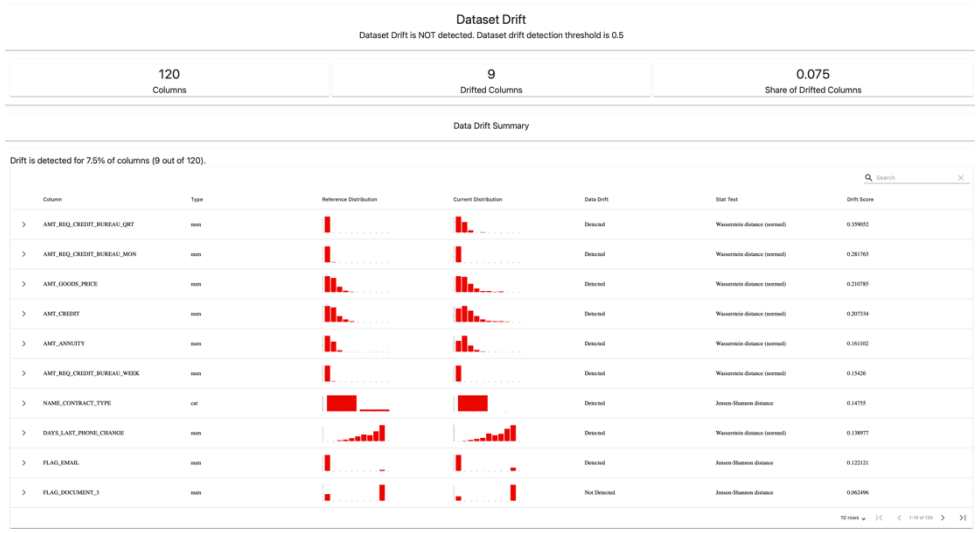
Une amélioration au niveau des hyperparamètres peut être envisagée, mais elle entraine une forte augmentation des poids des modèles.

Mon dashboard mériterait peut-être une amélioration cosmétique.

L’analyse du Data Drift

Qu'est-ce que le Data Drift ?

Le "data drift" fait référence à une modification ou un changement dans la distribution des données au fil du temps, ce qui peut affecter les performances d'un modèle de machine learning. Le modèle, en effet, est formé sur un ensemble de données spécifique, et si les données sur lesquelles il fait des prédictions changent, les performances du modèle peuvent en être affectées. Le data drift peut survenir pour diverses raisons, telles que les changements saisonniers, les modifications du comportement des utilisateurs, les changements de politique, etc.



Types de Data Drift

- Concept Drift : Le concept sous-jacent que le modèle tente de prédire change.

- Feature Drift: La distribution des caractéristiques ou des variables d'entrée change.

Analyse

Dans notre cas nous sommes confrontés à un Feature Drift, 9 des 120 colonnes ont drifté, soit environ 7,5 % des colonnes. Cela peut être considéré comme un niveau faible à modéré de data drift, selon le contexte et l'impact de ces colonnes sur le modèle.

Questions à Considérer

1. Importance des Feature : Ces 9 colonnes sont-elles cruciales pour les prédictions du modèle ? Si oui, même un faible taux de data drift peut avoir un impact significatif.

2. Contexte Temporel : Est-ce que le drift s'est produit sur une courte période ou s'est-il accumulé sur une longue période ? Un drift rapide peut nécessiter une action immédiate.

3. Impact sur les Performances : A-t-on observé une dégradation des performances du modèle ? Si oui, cela confirmerait que le drift a un impact significatif.

En résumé, bien que 7,5 % de colonnes ayant drifté puisse sembler faible, l'impact réel dépend de divers facteurs tels que l'importance de ces colonnes pour le modèle, le taux auquel le drift se produit, et les performances du modèle.