Note méthodologique

Méthodologie d'entraînement du modèle

La méthodologie mise en place pour l'entraînement du modèle est la suivante :

1. Oversampling: Les données d'entraînement sont déséquilibrées. Afin de pallier ce problème, une technique d'oversampling appelée SMOTE est appliquée. Cela consiste à générer des données synthétiques pour les classes minoritaires en utilisant des techniques de réplication.

2. Standardisation : Les données sont ensuite standardisées en utilisant le StandardScaler de scikit-learn. La standardisation est effectuée sur toutes les variables pour qu'elles soient centrées sur zéro et aient une variance unitaire.

3. Modèle : Le modèle utilisé est un modèle de classification. Dans le cadre de notre projet ce sera Dummy Classifier (en tant que Baseline), Logistic Regression, SGD Classifier, LGBM / Random Forest / XGBoost Classifier.

4. Validation croisée : La validation croisée est utilisée pour évaluer la performance du modèle. La méthode de validation croisée utilisée est StratifiedKFold avec 5 plis.

5. Paramétrage : Le modèle est entraîné sur une grille de paramètres définie par des valeurs selon le modèle de classification.

6. Optimisation : L'optimisation est effectuée à l'aide de la méthode de recherche de grille HalvingGridSearchCV de scikit-learn. Cette méthode permet de réduire le temps d'entraînement en éliminant les combinaisons de paramètres sous-performantes de manière itérative.

7. Évaluation : La performance du modèle est évaluée en utilisant la fonction coût-métier 'loan_score'. Cette fonction prend en compte les coûts associés à la classification incorrecte des clients et est utilisée pour optimiser le modèle.

En résumé, la méthodologie mise en place pour l'entraînement du modèle comprend l'utilisation de techniques d'oversampling, la standardisation des données, la validation

croisée, le paramétrage et l'optimisation du modèle, ainsi que l'utilisation d'une fonction coût-métier pour évaluer la performance du modèle.

Traitement du déséquilibre des classes

Le déséquilibre des classes dans les données est un problème courant dans les tâches de classification. Dans notre cas, nous avons observé que la classe 0 était sur-représentée par rapport à la classe 1, c'est-à-dire que la majorité des clients avaient été approuvés pour un prêt. Cette disproportion peut affecter les performances du modèle de classification, car le modèle aura tendance à favoriser la classe majoritaire, en négligeant la classe minoritaire.

Pour résoudre ce problème, nous avons utilisé la méthode SMOTE (Synthetic Minority Over-sampling Technique), qui est une technique d'oversampling qui permet de créer de nouvelles observations synthétiques de la classe minoritaire. Cela permet de rendre les classes plus équilibrées, ce qui peut améliorer les performances du modèle de classification.

En utilisant SMOTE, nous avons créé de nouvelles observations pour la classe minoritaire, en interpolant les caractéristiques des observations existantes de cette classe. Ainsi, le nombre d'observations pour la classe minoritaire est augmenté, et les classes sont équilibrées. Cela a permis à notre modèle de ne pas biaiser en faveur de la classe majoritaire, et d'être plus performant dans la classification des clients qui ne remplissent pas les critères de prêt.

Il est important de noter que l'utilisation de SMOTE doit être effectuée avec précaution, car cela peut entraîner un surapprentissage (overfitting) du modèle si le nombre de nouvelles observations générées est trop important. Il est donc recommandé de trouver un équilibre entre le nombre d'observations générées et le risque de surapprentissage.

Par ailleurs, il est important de comprendre que l'utilisation de SMOTE ne résout pas tous les problèmes liés au déséquilibre de classe. En effet, certaines classes minoritaires peuvent être très différentes de leurs homologues majoritaires en termes de caractéristiques, ce qui peut rendre leur interpolation difficile. Dans ces cas, il peut être nécessaire d'adopter une approche différente, telle que le sous-échantillonnage de la classe majoritaire ou l'utilisation d'autres méthodes de suréchantillonnage.

Enfin, il est important de souligner que la résolution du problème de déséquilibre des classes est souvent spécifique à chaque jeu de données et à chaque modèle. Il est donc essentiel de tester différentes approches pour déterminer celle qui fonctionne le mieux pour un problème donné.

Fonction coût métier, algorithme d'optimisation et métrique d'évaluation

La problématique "métier" dans ce projet est de minimiser le coût des erreurs de prédiction, en prenant en compte que les faux négatifs sont environ 10 fois plus coûteux que les faux positifs. Dans cette optique, le score F1 ne serait pas suffisant pour évaluer la performance du modèle.

Afin de répondre à cette problématique, nous avons créé un score "métier" qui calcule le coût des erreurs de prédiction. Ce score est créé à l'aide de la fonction "make_scorer" de la bibliothèque scikit-learn. Nous avons utilisé cette fonction pour calculer une fonction de coût métier de type 10*FN + FP, où FN représente le nombre de faux négatifs dans la matrice de confusion pour un seuil donné, et FP le nombre de faux positifs. Le but est de minimiser cette fonction pour un seuil donné afin d'optimiser le modèle.

Pour évaluer la performance des modèles, nous avons choisi d'utiliser l'AUC score, qui mesure la capacité du modèle à classer correctement les exemples positifs et négatifs. Cependant, le score "métier" que nous avons créé est plus adapté à la problématique "métier" de minimiser le coût des erreurs de prédiction en prenant en compte le déséquilibre entre les faux négatifs et les faux positifs. Ainsi, l'utilisation de cette fonction de coût métier dans le processus d'optimisation via GridSearchCV nous permet de sélectionner les modèles qui minimisent le coût des erreurs de prédiction pour un seuil donné, ce qui correspond mieux aux besoins de la banque.

Tableau de synthèse des résultats

Model Classifier	Fit Time (s)	Test Time (s)	AUC score
Dummy	10.17	0.01	0.500
Logistic Regression	13.43	0.01	0.723
SGD	6.41	0.01	0.712
LGBM	311.26	1.05	0.675
Random Forest	688.48	1.73	0.686
XGBoost	713.69	0.23	0.684

En se basant sur le tableau, on peut observer que les performances des modèles varient considérablement en termes de temps de formation, de temps de test et de score AUC. Le modèle le plus rapide à entraîner est SGD avec un temps de formation de seulement 6,41 secondes, suivi de près par le modèle Dummy avec un temps de formation de 10,17 secondes.

En termes de performances de prédiction, le modèle de régression logistique obtient le meilleur score AUC de 0.723. Il est suivi de près par SGD et Random Forest avec des scores AUC respectifs de 0.712 et 0.686. Les modèles LGBM et XGBoost ont des scores AUC légèrement plus faibles de 0.675 et 0.684 respectivement.

En ce qui concerne le temps de test, les modèle SGD, Logistic Regression et Dummy sont les plus rapides, ne prenant que 0.01 seconde. Le modèle Random Forest a le temps de test le plus long avec 1.73 seconde.

En résumé, nous pouvons dire que le modèle de régression logistique a la meilleure performance de prédiction globale, bien qu'il y ait des modèles plus rapides en termes de temps de formation et de test.

Interprétabilité globale et locale du modèle

L’interprétabilité globale du modèle se base sur les coefficients attribués à chaque variable explicative. Ces coefficients peuvent être obtenus à partir de la méthode gridmodel.best_estimator_.named_steps['classifier'].coef_[0]. Ils permettent de comprendre l'influence de chaque variable sur la prédiction et donc de déterminer quelles variables sont les plus importantes dans la prise de décision du modèle.

L’interprétabilité locale, quant à elle, se base sur la méthode shap.Explainer. Cette méthode permet d'expliquer la prédiction d'un modèle pour une observation spécifique en quantifiant l'importance de chaque variable dans la prédiction pour cette observation. Ainsi, l'interprétabilité locale permet de comprendre comment le modèle prend la décision pour une observation spécifique et peut aider à identifier les variables qui ont le plus d'impact sur la prédiction pour cette observation.

Il y a plusieurs limites et améliorations possibles pour le modèle et la méthodologie que nous avons utilisés :

Limites :

- La performance du modèle peut être limitée par la qualité des données. Il est important d'avoir des données de qualité pour obtenir des résultats précis et fiables.
- Bien que nous ayons utilisé la méthode SMOTE pour équilibrer les classes, il est possible qu'il y ait d'autres méthodes d'équilibrage de classe qui pourraient mieux fonctionner pour ce problème spécifique.
- Les modèles de Machine Learning ont souvent une capacité limitée à la généralisation, c'est-à-dire à leur capacité à faire des prédictions précises sur des données qu'ils n'ont jamais vues auparavant. Par conséquent, il est important de surveiller le modèle pour s'assurer qu'il continue à bien fonctionner au fil du temps.

Améliorations possibles :

- Nous pourrions explorer d'autres méthodes de prétraitement des données, telles que la normalisation ou le codage one-hot pour les variables catégorielles, pour voir si cela améliore la performance du modèle.
- Nous pourrions également essayer d'autres algorithmes de classification tels que les réseaux de neurones ou les SVM pour voir s'ils sont capables de mieux gérer les données et d'obtenir de meilleures performances.
- Pour améliorer l'interprétabilité locale, nous pourrions utiliser des techniques de visualisation plus avancées telles que des cartes de chaleur pour comprendre comment les caractéristiques individuelles influencent les prédictions.
- Pour améliorer l'interprétabilité globale, nous pourrions explorer d'autres techniques d'interprétation telles que LIME ou Partial Dependence Plots pour comprendre comment les caractéristiques interagissent entre elles pour influencer les prédictions du modèle.

Analyse drift

L'analyse de dérive de données a été effectuée sur 8 caractéristiques. Le résultat montre que la dérive a été détectée pour 1 des 8 caractéristiques, soit une proportion de 12.5%. Cependant, le dataset en lui-même n'a pas subi de dérive.

La caractéristique pour laquelle la dérive a été détectée est EXT_SOURCE_1. La distance de Wasserstein normée pour cette caractéristique est de 0.159398. Cela indique qu'il y a eu un changement significatif dans la distribution des données pour cette caractéristique entre les deux ensembles de données.

Les autres caractéristiques n'ont pas montré de dérive de données. La distance de Wasserstein normée qui est une mesure de la distance entre deux distributions de probabilité pour chacune de ces caractéristiques est inférieure à 0.1, ce qui indique une petite différence entre les deux ensembles de données.

Il est important de noter que la détection de dérive de données n'indique pas nécessairement un problème. Cela signifie simplement que la distribution des données pour une caractéristique a changé entre les deux ensembles de données. Il est important d'examiner les raisons de ce changement pour déterminer si cela a un impact sur les résultats de l'analyse et si des mesures doivent être prises pour y remédier.