Rapport

Contexte et Objectif :

Le projet avait pour objectif de développer un modèle de prédiction capable d'optimiser les campagnes de marketing téléphonique menées par une banque portugaise. Ces campagnes, qui nécessitent souvent plusieurs appels, peuvent être chronophages et coûteuses. Le but principal était de réduire le nombre d'appels nécessaires en identifiant les clients les plus susceptibles de souscrire à une offre. La variable cible est binaire, avec les valeurs "yes" et "no", indiquant respectivement l'acceptation ou le refus de l'offre.

1. Exploration des Données :

Traitement des Valeurs Manquantes et des Variables Catégorielles :

Lors de l'analyse exploratoire des données, il a été observé qu'aucune donnée n'était réellement manquante, mais plusieurs variables catégorielles comportaient des valeurs codées sous "unknown". Ces valeurs ont été supprimées après application du One-Hot Encoding, qui permet de transformer les variables catégorielles en variables numériques.

Déséquilibre des Classes :

Une observation importante fut le déséquilibre des classes dans la variable cible. Cela peut poser un problème pour l'apprentissage automatique, car les modèles pourraient être biaisés en faveur de la classe majoritaire. Ce déséquilibre a nécessité l'utilisation de certaines techniques, telles que la stratification et l'augmentation de données, pour améliorer les performances des modèles.

2. Prétraitement des Données :

Encodage des Variables :

J'ai créé deux listes distinctes : une pour les variables catégorielles et une pour les variables numériques. Pour les variables catégorielles, j'ai utilisé un One-Hot Encoding, et pour les variables numériques, un StandardScaler a été appliqué pour normaliser les données.

Transformation de la Variable Cible :

La variable cible a été convertie à l'aide du `LabelEncoder` pour transformer les étiquettes "yes" et "no" en valeurs binaires (1 et 0, respectivement).

Split des Données :

Le jeu de données a été divisé en un ensemble d'apprentissage (train) et un ensemble de test (test), en utilisant une stratification basée sur la variable cible. La **stratification** consiste à garantir que chaque sous-ensemble (train et test) conserve la proportion de chaque classe

présente dans l'ensemble de données original, ce qui est essentiel lorsqu'il y a un déséquilibre des classes.

3. Choix des Modèles de Machine Learning :

Les modèles suivants ont été utilisés :

- **Logistic Regression** : Un modèle linéaire adapté pour la classification binaire. L'option `class_weight='balanced'` permet de traiter le déséquilibre des classes.
- **Random Forest** : Un modèle basé sur des arbres de décision, idéal pour les problèmes de classification avec un grand nombre de variables. Il est robuste aux données manquantes et au bruit.
- **XGBoost** : Un modèle de boosting très efficace, particulièrement adapté aux grandes bases de données avec des données déséquilibrées.
- **LightGBM** : Un autre modèle de boosting, optimisé pour les jeux de données volumineux et rapide à l'entraînement.
- **CatBoost** : Un modèle de boosting qui gère bien les variables catégorielles et est moins sensible à l'overfitting.

Tous ces modèles sont adaptés pour des problèmes de classification binaire, et la gestion des classes déséquilibrées a été réalisée à l'aide de paramètres tels que `class_weight='balanced'` et l'option de réglage spécifique pour chaque modèle.

4. Choix des Métriques d'Évaluation :

Les **métriques** choisies pour évaluer les performances des modèles sont :

- **F1-Score** : C'est la métrique de référence car elle est une moyenne harmonique de la précision et du rappel. Elle est particulièrement utile dans les cas de déséquilibre des classes, où une simple précision pourrait être trompeuse.
- **AUC-ROC** : Cette métrique permet de mesurer la performance du modèle indépendamment du seuil de décision, et est particulièrement adaptée pour les problèmes de classification binaire déséquilibrée.

5. Résultats et Optimisation :

Initialement, les résultats des modèles tournaient autour de 0.50 à 0.56 de F1-score, avec CatBoost et LightGBM en tête. Cependant, l'optimisation a permis de gagner quelques points :

Application de la Méthode SMOTE :

Le **SMOTE (Synthetic Minority Over-sampling Technique)** est une méthode d'augmentation de données qui génère des échantillons synthétiques pour la classe minoritaire en interpolant les points voisins. Cela permet d'équilibrer les classes, mais dans ce cas, l'ajout de données synthétiques a entraîné une baisse des performances. Cela pourrait être dû à la génération de points qui ne sont pas représentatifs des données réelles, ce qui pourrait introduire du bruit et dégrader la qualité du modèle.

Recherche des Meilleurs Hyperparamètres (Grid Search) :

Ensuite, une recherche exhaustive des hyperparamètres (Grid Search) a été effectuée pour les modèles CatBoost et LightGBM. Cela a permis d'améliorer les performances, passant à un F1-score de 0.60 et un AUC de 0.92.

Optimisation du Seuil de Décision :

Enfin, en cherchant à optimiser le **seuil de décision**, le F1-score est monté à 0.66 avec un AUC de 0.95 pour les deux modèles avec un seuil fixé à 0.3. Cela a permis d'améliorer les résultats par rapport à un F1-score de 0.56 initial.

6. Conclusion :

Les résultats finaux montrent une amélioration significative des performances du modèle, avec un F1-score de 0.66 et un AUC de 0.95 pour les deux meilleurs modèles (CatBoost et LightGBM). Toutefois, plusieurs points sont à prendre en compte :

- **Données supplémentaires** : Il est probable que l'ajout de plus de données, en particulier pour la classe minoritaire, puisse encore améliorer les performances. Le déséquilibre des classes reste un problème clé à résoudre pour obtenir des modèles plus robustes.
- **Ajustement du seuil de décision** : La recherche du seuil optimal a montré des gains importants, ce qui confirme l'importance de cette étape pour améliorer le F1-score sans nuire à l'AUC.
- **SMOTE** : L'augmentation des données avec SMOTE n'a pas été bénéfique dans ce cas particulier, et il serait intéressant de tester d'autres techniques d'augmentation ou de rééchantillonnage.

En conclusion, bien que des améliorations notables aient été obtenues, il est nécessaire de continuer à explorer des solutions pour équilibrer les classes, ajouter des données supplémentaires, et peut-être tester d'autres techniques d'augmentation pour améliorer encore les performances du modèle.