Mini projet : Nettoyage, Transformation et Déploiement d'un Modèle de Prédiction des Commandes en Retard (partie 1)

Introduction

L'objectif de ce mini projet est de développer une chaîne complète pour résoudre un problème de **classification supervisée** visant à prédire les commandes en retard (**Backorder Prediction**). Une commande en retard se produit lorsqu'un client potentiel tente de commander un produit, mais que sa commande ne peut être exécutée immédiatement parce que l'entreprise ne dispose pas du produit en question en stock. Le client est informé que le produit est en rupture de stock dans cette situation. Ce dernier a la possibilité de poursuivre la transaction, d'effectuer un achat, puis d'attendre le nouveau produit. Il peut également choisir de refuser la commande et de ne pas l'exécuter, ou de la poursuivre mais de l'annuler s'il trouve une solution plus rapide.

Vous serez amené à nettoyer et transformer un jeu de données, entraîner plusieurs modèles intelligents, et déployer le modèle final dans un environnement de production à l'aide de technologies modernes telles que Docker, Jenkins, Kubernetes et Kubeflow.

Le jeu de données (DS) de cette recherche est disponible sur les bases de données de Kaggle (https://www.kaggle.com/c/untadta/data). Le (DS) comprend deux ensembles ; l'ensemble d'entraînement (TR) et l'ensemble du test (TS).

Partie 1 : Prétraitement des Données

Étape 1 : Exploration initiale

1. Affichez les cinq premières lignes des ensembles d'entraînement (**TR**) et de test (**TS**).
2. Obtenez les dimensions des deux ensembles.
3. Identifiez les types de variables présentes dans les datasets.

Étape 2 : Gestion des valeurs manquantes

4. Identifiez les valeurs manquantes dans chaque colonne et calculez leur pourcentage. Interpréter les résultats.
5. Affichez des tableaux de statistiques descriptives pour TR et TS (moyennes, médianes, écarts-types).
6. Proposez des solutions pour traiter les valeurs manquantes.

Partie applicative :

- Utilisez Docker pour conteneuriser votre script Python de nettoyage des données afin de garantir une exécution reproductible.
- Créez un fichier Dockerfile contenant l'image Python avec les dépendances nécessaires (Pandas, NumPy, etc.).

Partie 2 : Sélection des Variables Pertinentes

Étape 3 : Analyse de corrélation

7. Calculer la corrélation entre les variables (Appliquer le test de khi deux pour les variables catégorielles). Si la valeur de corrélation dépasse 90% (les deux variables sont fortement corrélées) éliminer l'une de valeurs de DS. Laquelle qu'on doit éliminer et pourquoi ? C'est quoi la nouvelle dimensionnalité de notre DS ?
8. Appliquez la méthode **PCA** pour réduire la dimensionnalité.
9. Comparez les deux approches (corrélation et PCA) et interprétez les résultats.

Partie 3 : Rééchantillonnage des Données Déséquilibrées

10. Afficher la distribution de la valeur de sortie. Interprétation
11. Appliquer une méthode de suréchantillonnage (Oversampling) pour résoudre le problème de déséquilibre.
12. Afficher la distribution de la valeur de sortie après le rééchantillonnage.

Partie 4 : Modélisation et Évaluation

Étape 1 : Conception des modèles intelligents

13. Entraînez un classificateur basé sur l'un des modèles suivants :
    - **Random Forest**
    - **Decision Tree**
    - **Artificial Neural Network (ANN)**
14. Entraînez votre modèle sur :
    - Les données avant le rééchantillonnage.
    - Les données après le rééchantillonnage.

Étape 2 : Évaluation des modèles

1. Calculez la **matrice de confusion** pour chaque cas.
2. Calculez les métriques d'évaluation suivantes avant et après rééchantillonnage :
    - Exactitude, Précision, TPR, TNR, FNR, FPR, Score F1, et Spécificité.
3. Comparez les performances des modèles et discutez des résultats.

Partie applicative :

- Automatisez l'entraînement avec Jenkins pour permettre une exécution continue (CI/CD) lorsque le code ou les données changent.
- Créez un pipeline Jenkins qui :
    1. Télécharge les données.
    2. Nettoie les données avec le script Dockerisé.
    3. Entraîne le modèle et stocke les artefacts générés (modèle entraîné, logs, métriques).

Partie 5 : Déploiement en Production

Étape 1 : Conteneurisation avec Docker Compose

1. Utilisez **Docker Compose** pour orchestrer l'environnement, incluant :
    - Un conteneur pour le modèle déployé.
    - Un conteneur pour un serveur API (par exemple, Flask ou FastAPI).
    - Un conteneur pour une base de données de stockage des prédictions.