Projet MLOps

Deuxième année Master SIAD - Parcours DS

Année Universitaire 2024-2025

Introduction

Le **MLOps** (Machine Learning Operations) est une pratique qui combine les principes du **DevOps** avec les spécificités du **Machine Learning**. Elle vise à automatiser et optimiser le développement, le déploiement et la gestion des modèles de machine learning en production. L'objectif est d'améliorer la collaboration entre les équipes de data science et les opérations tout en garantissant des modèles fiables et performants à long terme.

Notre projet consiste à prédire si un citoyen des États-Unis gagne plus de 50 000 $ par an en fonction de diverses caractéristiques socio-économiques (âge, niveau d'éducation, profession, etc.). En utilisant des techniques de machine learning, nous entraînons plusieurs modèles de classification pour prédire cette variable cible binaire : `<=50K` ou `>50K`.

L'objectif est de préparer les données, entraîner et évaluer les modèles, tout en suivant des pratiques MLOps pour assurer une gestion efficace des données, du déploiement et de la performance des modèles en production.

Sommaire

1. L’organisation du travail

Pour mener au mieux ce projet, nous avons découpé le projet en plusieurs tâches. Grâce à cette répartition, nous avons pu nous répartir les différentes tâches comme ci-dessous :

- Import et analyse des variables - Thomas Rodrigues
- Recodage des variables - Klervi Dagorne
- Traitement des données manquantes et aberrantes - Elena Spindler
- Modélisation - Lucas Meganck

2. Les fonctions

1. Fonction import des données

La fonction load_and_describe_data charge un fichier CSV et renvoie son contenu sous forme de DataFrame. Il s'agit d'une étape clé pour importer les données brutes dans le pipeline de traitement.

Entrée

- file_path (str) : Le chemin du fichier CSV à charger.

Sortie

- df (DataFrame) : Un DataFrame pandas contenant les données chargées depuis le fichier CSV.

Fonctionnement

La fonction utilise pandas.read_csv pour lire le fichier spécifié par file_path et retourne un DataFrame contenant les données prêtes pour une inspection ou un traitement ultérieur.

2. Fonction statistiques descriptives

La fonction descriptive_statistics fournit un aperçu des données contenues dans un DataFrame en affichant :

- Les statistiques descriptives des variables numériques.
- Le nombre de valeurs uniques pour chaque variable catégorielle.
- Les fréquences et proportions des modalités pour chaque variable catégorielle.

Entrée

- df (DataFrame) : Un DataFrame pandas contenant les données à analyser.

Sortie

Cette fonction n'a pas de sortie explicite. Elle affiche directement les informations suivantes :

- Les statistiques descriptives des colonnes numériques (via df.describe()).
- Le nombre de valeurs uniques par variable catégorielle.
- Les fréquences et proportions (%) des modalités pour chaque variable catégorielle.

Fonctionnement

- Identifie les colonnes numériques pour afficher leurs statistiques descriptives (moyenne, écart-type, min, max, etc.).

- Compte les valeurs uniques pour chaque colonne catégorielle.
- Calcule les fréquences et proportions des modalités pour chaque colonne catégorielle et les affiche sous forme de tableau.

3. Fonction distribution des variables numériques

La fonction `plot_categorical_distributions` visualise la distribution des variables catégorielles d'un DataFrame en affichant des diagrammes en barres. Elle aide à comprendre la répartition des observations dans chaque modalité.

Entrée

- **df (DataFrame)** : Un DataFrame `pandas` contenant les données à visualiser.

Sortie

Cette fonction ne retourne rien. Elle affiche des diagrammes en barres pour chaque variable catégorielle dans le DataFrame.

Fonctionnement

- Identifie les colonnes catégorielles du DataFrame.
- Pour chaque colonne catégorielle :
    - Crée un diagramme en barres avec les modalités triées par fréquence.
    - Ajoute un titre et des axes pour améliorer la lisibilité.
- Affiche chaque graphique en utilisant `matplotlib` et `seaborn`.

4. Fonction distribution des variables catégorielles

La fonction plot_categorical_distributions visualise la distribution des variables catégorielles d'un DataFrame en affichant des diagrammes en barres. Elle aide à comprendre la répartition des observations dans chaque modalité.

Entrée

- df (DataFrame) : Un DataFrame pandas contenant les données à visualiser.

Sortie

Cette fonction ne retourne rien. Elle affiche des diagrammes en barres pour chaque variable catégorielle dans le DataFrame.

Fonctionnement

- Identifie les colonnes catégorielles du DataFrame.
- Pour chaque colonne catégorielle :
    - Crée un diagramme en barres avec les modalités triées par fréquence.
    - Ajoute un titre et des axes pour améliorer la lisibilité.
- Affiche chaque graphique en utilisant matplotlib et seaborn.

5. Fonction matrice de corrélation

La fonction `plot_correlation_matrix` visualise la matrice de corrélation des variables numériques d'un DataFrame sous forme de heatmap. Elle permet d'identifier les relations linéaires entre les variables.

Entrée

- **df (DataFrame)** : Un DataFrame `pandas` contenant les données à analyser.

Sortie

Cette fonction ne retourne rien. Elle affiche une heatmap représentant les coefficients de corrélation entre les colonnes numériques.

Fonctionnement

- Sélectionne les colonnes numériques du DataFrame.
- Calcule la matrice de corrélation en utilisant `DataFrame.corr()`.
- Affiche une heatmap de la matrice de corrélation avec :
    - Les coefficients annotés.
    - Une palette de couleurs pour indiquer l'intensité des corrélations.

6. Fonction suppression de valeur manquantes

La fonction drop_data permet de nettoyer un DataFrame en supprimant toutes les lignes contenant le caractère '?'. Cette opération est utilisée pour ne plus avoir de valeurs manquantes ou indésirables représentées par '?'.

Entrée

- df (DataFrame) : Un DataFrame pandas contenant les données à nettoyer.

Sortie

- df_cleaned (DataFrame) : Un DataFrame où toutes les lignes contenant au moins un '?' ont été supprimées.

Fonctionnement

La fonction supprime toutes les lignes du DataFrame où une valeur de '?' est présente dans n'importe quelle colonne. Elle utilise la méthode isin() pour identifier ces lignes et les exclure.

7. Fonction de recodage

La fonction `numeriser_toutes_colonnes` transforme un DataFrame contenant des données socio-économiques en un format numérique, prêt pour l'analyse ou la modélisation. Elle supprime la colonne `education` (si présente), car cette variable est la même qu'une variable déjà présente et numérisée : education_num. Puis elle numérise plusieurs colonnes catégorielles spécifiques.

Entrée

- **df (DataFrame)** : Un DataFrame `pandas` contenant les données à transformer.

Sortie

- **df (DataFrame)** : Un DataFrame où :
    - La colonne `education` a été supprimée (si elle existe).
    - Les colonnes catégorielles (`workclass`, `marital-status`, `occupation`, etc.) sont numérisées selon des mappings prédéfinis.

Fonctionnement

- **Suppression de la colonne `education`** : Si la colonne `education` existe, elle est supprimée.
- **Numérisation** : Les colonnes catégorielles sont converties en valeurs numériques à l'aide de mappings définis pour chaque colonne.
- **Nettoyage des espaces** : Les espaces dans les colonnes de texte sont supprimés avant la numérisation pour garantir la cohérence des données.

Mappings

Les valeurs de certaines colonnes sont remplacées par des entiers :

- **workclass :** 'Private' : 1, 'Self-emp-not-inc' : 2, etc.
- **marital-status :** 'Never-married' : 1, 'Married-civ-spouse' : 2, etc.
- **occupation :** 'Machine-op-inspct' : 1, 'Farming-fishing' : 2, etc.
- **income :** '<=50K' : 0, '>50K' : 1, etc.

8. Fonction Analyse Exploratoire des données

La fonction `full_data_analysis` effectue une analyse complète des données d'un fichier CSV. Elle couvre plusieurs étapes : chargement, nettoyage, numérisation des colonnes, et génération de statistiques descriptives et de visualisations pour mieux comprendre les données.

Entrée

- **file_path (str)** : Le chemin du fichier CSV à analyser.

Sortie

Cette fonction ne retourne rien. Elle exécute plusieurs opérations et affiche les résultats suivants :

- Statistiques descriptives des données nettoyées et numérisées.
- Visualisations des distributions des variables numériques et catégorielles.
- La matrice de corrélation des variables numériques.

Fonctionnement

- **Chargement des données** : Le fichier CSV est chargé à l'aide de la fonction `load_and_describe_data`.
- **Nettoyage des données** : Les lignes contenant des valeurs `'?'` sont supprimées avec `drop_data`.
- **Vérification de l'intégrité des données** : Si le DataFrame nettoyé est vide après la suppression des lignes contenant `'?'`, un message d'avertissement est affiché.
- **Numérisation des données** : Les colonnes catégorielles sont numérisées avec la fonction `numeriser_toutes_colonnes`.
- **Analyse exploratoire** :
    - **Statistiques descriptives** sont générées pour les variables numériques et catégorielles via `descriptive_statistics`.
    - **Visualisations** sont créées pour les variables numériques (histogrammes) et catégorielles (diagrammes en barres).
    - **Matrice de corrélation** des variables numériques est générée et affichée sous forme de heatmap.

9. Fonction Modélisation

La fonction `model` entraîne et évalue plusieurs modèles de classification sur un DataFrame d'entrée. Elle effectue une recherche d'hyperparamètres pour chaque modèle et affiche les résultats d'évaluation sur un ensemble de test.

Entrée

- **df (DataFrame)** : Un DataFrame `pandas` contenant les données à analyser. La fonction suppose que la colonne cible est `income` (variable à prédire).

Sortie

- **pd.DataFrame** : Un DataFrame contenant les résultats de l'évaluation des modèles. Cela inclut :
    - Les meilleurs hyperparamètres pour chaque modèle.
    - La précision (`accuracy`), le score du modèle, et le rapport de classification pour chaque modèle.

Fonctionnement

- **Séparation des données** : Le DataFrame est divisé en variables explicatives (`X`) et la variable cible (`y`, `income`).-
- **Diviser les données** : Les données sont divisées en ensembles d'entraînement et de test avec `train_test_split`.
- **Modèles de classification** :
    - Les modèles utilisés sont : `DecisionTreeClassifier`, `KNeighborsClassifier`, `LogisticRegression`, `RandomForestClassifier`.
    - Pour chaque modèle, un pipeline est défini, avec ou sans mise à l'échelle des données en fonction du modèle.
- **Recherche d'hyperparamètres** : Pour chaque modèle, une recherche de grille (`GridSearchCV`) est effectuée pour trouver les meilleurs hyperparamètres.
- **Évaluation des modèles** :
    - Les meilleurs paramètres sont affichés.
    - La précision, le score et le rapport de classification sont calculés et affichés pour chaque modèle sur l'ensemble de test.