Rapport sur le script Python pour le traitement des données hospitalières

I. Introduction

Le script développé a pour objectif de manipuler et de nettoyer des données provenant de deux fichiers CSV relatifs à des patients hospitaliers. Les opérations réalisées incluent la conversion et le formatage des dates, la gestion des valeurs manquantes, l'ajout de nouvelles colonnes calculées (âge et statut), la fusion de ces deux fichiers et la gestion des doublons. Enfin, un log des modifications effectuées est généré pour assurer une traçabilité des changements.

II. Choix d'implémentation

a. Lecture des fichiers de données

Les deux fichiers CSV, hospital_data.csv et additional_hospital_data.csv, sont lus à l'aide de la bibliothèque pandas. Cette bibliothèque est particulièrement adaptée à la gestion des données tabulaires grâce à ses structures de données puissantes, telles que les DataFrames, qui facilitent la manipulation des données.

b. Conversion des dates et formatage

L'une des premières étapes de traitement des données concerne la conversion des dates de naissance (DOB) et des dates de dernière visite (LastVisit). Pour garantir l'uniformité des formats de date, les colonnes sont converties en objets datetime avec la fonction pd.to_datetime. Par la suite, les dates sont formatées au format DD/MM/YYYY grâce à la méthode strftime('%d/%m/%Y').

c. Gestion des valeurs manquantes

Dans cette étape, le script vérifie les valeurs manquantes dans la colonne Diagnosis. Lorsque des valeurs manquantes sont trouvées, elles sont remplacées par la valeur la plus fréquente dans la colonne, obtenue par la méthode mode(). Cette approche est

raisonnable lorsque les données manquantes sont peu nombreuses, permettant d'éviter de perdre des informations tout en garantissant une cohérence des valeurs.

d. Calcul de l'âge des patients

Une colonne Age a été ajoutée au fichier en calculant la différence entre l'année actuelle et l'année de naissance. Afin de préserver le format original de la colonne DOB, une conversion temporaire au format datetime a été effectuée avec pd.to_datetime()

e. Formatage de la colonne "Diagnosis"

Les noms de colonnes dans Diagnosis sont mises en majuscules grâce à la méthode str.upper(), ce qui permet de standardiser les diagnostics et d'éviter des erreurs lors de l'analyse ou de l'application de filtres.

f. Ajout de la colonne "Statut"

Une nouvelle colonne Statut est ajoutée, indiquant si le patient est sain ou malade. Cette valeur est attribuée en fonction de la valeur de la colonne Diagnosis, où "HEALTHY" est interprété comme un patient sain, et toute autre valeur comme un patient malade. Ce processus est effectué via une fonction lambda appliquée à chaque élément de la colonne Diagnosis.

g. Fusion des deux jeux de données

Les deux jeux de données sont fusionnés grâce à la fonction pd.concat(). Cette méthode permet de combiner les deux DataFrames tout en préservant l'intégrité des données. Ensuite, un contrôle des doublons est effectué en utilisant la méthode duplicated(), qui vérifie la présence de doublons sur la colonne PatientID. Les doublons sont supprimés grâce à la méthode drop_duplicates().

h. Log des modifications

Pour assurer une traçabilité des changements effectués, un fichier log (modification_log.txt) est généré. Ce fichier liste toutes les modifications réalisées durant le traitement des données, notamment la conversion des dates, le remplissage des valeurs manquantes, l'ajout de nouvelles colonnes et la suppression des doublons