Rapport sur l'implémentation du projet de nettoyage et de fusion des données hospitalières

1. Introduction

Ce projet avait pour objectif de **nettoyer** et de **fusionner** deux ensembles de données hospitalières provenant de fichiers CSV distincts. Le nettoyage a inclus la gestion des **valeurs manquantes**, l'uniformisation des formats de données et l'ajout de nouvelles colonnes calculées. La **fusion** des données a permis de combiner les deux DataFrames en un fichier unique. En plus de cela, un fichier de **log** a été utilisé pour suivre toutes les étapes du processus.

2. Choix d'implémentation et gestion des problèmes rencontrés

2.1. Gestion des valeurs manquantes

Lors du nettoyage des données, une étape clé a été la gestion des **valeurs manquantes** dans certaines colonnes telles que **Diagnosis**, **TreatmentCost** et **LastVisit**. Pour les colonnes **Diagnosis** et **TreatmentCost**, les lignes contenant des valeurs manquantes ont été supprimées afin de garantir des données pertinentes. Cependant, pour des colonnes comme **Diagnosis**, où des valeurs manquantes ont été trouvées, il a été décidé de remplir les cellules vides par le terme **"Inconnu"**. Cela a permis de maintenir une certaine cohérence tout en évitant de perdre des informations importantes. Pour **TreatmentCost**, les valeurs manquantes ont été remplacées par la **moyenne** de la colonne, afin de minimiser l'impact de ces valeurs manquantes sur les analyses futures.

Problème rencontré : La gestion des **valeurs manquantes** a soulevé des questions sur l'intégrité des données. L'option choisie était de remplir certaines cellules avec des valeurs génériques tout en conservant un contrôle sur les données manquantes dans des colonnes sensibles comme **le coût du traitement**.

2.2. Uniformisation des formats de dates

Une autre étape importante a été l'**uniformisation des formats de dates** dans les colonnes **DOB** (Date de naissance) et **LastVisit**. Initialement, ces dates étaient dans des formats variés entre les deux fichiers CSV, ce qui pouvait entraîner des erreurs lors des calculs. J'ai utilisé la fonction **pd.to_datetime()** de pandas pour convertir ces colonnes en objets datetime, puis j'ai reformatté les dates au format **DD/MM/YYYY** pour garantir la cohérence.

Problème rencontré : La conversion des dates a provoqué des erreurs dans certains cas où des valeurs **non valides** ou mal formatées étaient présentes. Ces erreurs ont été gérées avec l'option **errors="coerce"**, qui transforme ces valeurs en **NaT** (Not a Time), permettant ainsi d'éviter les crashs du programme.

2.3. Calcul de l'âge à partir de la date de naissance

Dans l'Exercice 2, il était nécessaire de **calculer l'âge** des patients à partir de leur date de naissance. Le calcul a été effectué en utilisant la différence entre l'année actuelle et l'année de naissance, tout en gérant les cas où la **date de naissance** était manquante.

Problème rencontré : Lors du calcul de l'âge, certaines dates de naissance étaient manquantes ou mal formatées. En utilisant **pd.to_datetime()** et l'option **errors="coerce"**, ces valeurs ont été transformées en **NaT**, permettant de vérifier facilement si l'âge pouvait être calculé.

2.4. Ajout de la colonne "Statut"

Une autre transformation des données a consisté à ajouter une nouvelle colonne **Statut**, indiquant si le patient était **"Sain"** ou **"Malade"**, en fonction du diagnostic. Cela a été réalisé avec une fonction **lambda** appliquée à la colonne **Diagnosis**.

Problème rencontré : Un problème potentiel résidait dans les incohérences dans les valeurs de la colonne **Diagnosis**, notamment des fautes de frappe ou des valeurs non standardisées. Afin de garantir que le diagnostic **"Sain"** soit correctement identifié, une uniformisation préalable des données a été nécessaire.

2.5. Fusion des DataFrames

L'Exercice 3 impliquait la **fusion** des deux DataFrames **df** et **df2** sur la colonne commune **PatientID**. Cette opération a permis de combiner les données de manière cohérente. Cependant, des **doublons** sont apparus, créant des colonnes redondantes.

Problème rencontré : La fusion a généré des colonnes **redondantes** avec des suffixes **_df1** et **_df2** pour différencier les colonnes identiques des deux DataFrames. Afin d'éviter toute confusion lors de l'analyse, il a été nécessaire de choisir les valeurs les plus complètes de chaque colonne, en utilisant **fillna()** pour combiner les informations des deux DataFrames et supprimer les doublons.

2.6. Sauvegarde des fichiers et gestion des erreurs de lecture/écriture

Un fichier log, **modifications_log.txt**, a été maintenu pour consigner chaque modification apportée aux données, comme le nettoyage, la suppression des valeurs manquantes, l'ajout de colonnes et la fusion des DataFrames. Les logs ont été enregistrés après chaque étape pour assurer une traçabilité complète.

Problème rencontré : Des erreurs telles que l'impossibilité d'accéder aux fichiers ou des conflits d'encodage sont survenues. Ces problèmes ont été résolus en utilisant la méthode sécurisée with open() pour garantir une bonne gestion des fichiers et éviter les pertes de données.

3. Conclusion

Le projet a permis de créer des données nettoyées et fusionnées, prêtes pour une analyse approfondie. Les choix faits, comme le remplissage des valeurs manquantes et l'uniformisation des formats, ont visé à assurer la cohérence des données. Les problèmes rencontrés, tels que les valeurs manquantes, les erreurs de dates et les doublons, ont été résolus avec des méthodes adaptées. En fin de compte, le projet a permis de créer des fichiers CSV propres, prêts à être utilisés pour des analyses futures.