Doc Script Python

Sommaire

Objectif du script

Ce script python a pour but de traiter des données relatives au diabète à partir de fichiers CSV, de les nettoyer, de les enrichir avec de nouvelles colonnes, puis de les insérer dans une base de données MySQL. Plus précisément, il :

- Crée une base de données et des tables nécessaires si elles n'existent pas.
- Lit plusieurs fichiers CSV, les fusionne, et effectue des opérations de nettoyage et de transformation sur les données.
- Ajoute des colonnes dérivées, telles que "Diabete" et "Pregnant", en fonction de certaines conditions.
- Insère les données traitées dans des tables spécifiques de la base de données.

Fonctionnement du script

1 - Importation des bibliothèques Le script commence par importer plusieurs bibliothèques nécessaires :

```
import glob
import os
import mysql.connector
import csv
import numpy as np
import pandas as pd
import mysql.connector
import pandas as pd
```

- glob et os : Pour la gestion des fichiers et des répertoires.
- mysql.connector : Pour établir une connexion avec la base de données MySQL.
- csv : Pour lire et écrire des fichiers CSV.
- numpy et pandas : Pour la manipulation et l'analyse des données.

2 - Configuration de la connexion MySQL

Le script définit les paramètres de connexion à la base de données MySQL, y compris le nom de la base de données, l'hôte, l'utilisateur et le mot de passe.

```
# Configuration de la connexion MySQL
DB_NAME = "Diabete_MSPR"
DB_CONFIG = {
    "host": "localhost",  # Modifier si nécessaire
    "user": "root",  # Modifier selon vos identifiants
    "password": "",  # Modifier selon vos identifiants
    "database": DB_NAME,
}

# Connexion au serveur MySQL
conn = mysql.connector.connect(
    host=DB_CONFIG["host"], user=DB_CONFIG["user"], password=DB_CONFIG["password"]
)
```

3 - Création de la base de données

Il se connecte à MySQL et crée la base de données spécifiée si elle n'existe pas. Ensuite, il supprime toutes les tables existantes pour éviter les conflits de données.

```
# Connexion au serveur MySQL
conn = mysql.connector.connect(
    host=DB_CONFIG["host"], user=DB_CONFIG["user"], password=DB_CONFIG["password"]
)
cursor = conn.cursor()

# Création de la base de données si elle n'existe pas
cursor.execute(f"CREATE DATABASE IF NOT EXISTS {DB_NAME}")
conn.database = DB_NAME

# Supprimer toutes les tables existantes
cursor.execute("SET FOREIGN_KEY_CHECKS = 0;")
cursor.execute("SHOW TABLES;")
tables = cursor.fetchall()
for table in tables:
    cursor.execute(f"DROP TABLE IF EXISTS {table[0]}")
cursor.execute("SET FOREIGN_KEY_CHECKS = 1;")
```

4 - Lecture et exécution des requêtes SQL

Le script lit un fichier SQL contenant des instructions pour créer les tables nécessaires et exécute ces requêtes.

```
# Lire le fichier SQL et exécuter les requêtes
with open("Diabete_MSPR-1740578494.sql", "r") as sql_file:
    sql_script = sql_file.read()
    for statement in sql_script.split(";"):
        if statement.strip():
            cursor.execute(statement)
```

5 - Traitement des fichiers CSV

Le script utilise glob pour récupérer tous les fichiers CSV dans un répertoire spécifié.

```
csv_files = glob.glob(INPUT_DIRECTORY)
```

Il lit chaque fichier CSV, normalise les noms de colonnes en minuscules, et les fusionne en un seul DataFrame.

Il supprime également les doublons et remplace les valeurs manquantes par "Na".

```
# Supprimer les lignes en doublon
df_fusionne = df_fusionne.drop_duplicates()

# Remplacer les cases vides par "Na"
df_fusionne = df_fusionne.fillna("Na")
```

6 - Enrichissement des données

Le script ajoute plusieurs colonnes :

- Colonne "Diabete" : Déterminée par la valeur de "glyhb" (6.5 ou plus = Oui, moins de 6.5 = Non) et "outcome" (1 = Oui, 0 = Non).

diabete
1
0
1
0

```
if (
    pd.to_numeric(row["glyhb"], errors="coerce") >= 6.5
    or row["outcome"] == 1
)
else (
    "0"
    if (
        pd.to_numeric(row["glyhb"], errors="coerce") < 6.5
        or row["outcome"] == 0
    )
    else "Na"
```

- Colonne "Pregnant" : Indique si le nombre de grossesses est supérieur à zéro.

pregnant
1
1
1
1
0

- Colonne "bloodpressure" : Remplace les valeurs manquantes par une valeur calculée à partir d'autres colonnes.

```
df_fusionne["moyenne_bp"] = df_fusionne.apply(
    lambda row: (
        round(row["bp.1d"] + (1 / 3 * (row["bp.1s"] - row["bp.1d"])), 1)
        if pd.notna(row["bp.1s"]) and pd.notna(row["bp.1d"])
        else np.nan
```

7 - Insertion des données dans la base de données

Le script insère les données traitées dans les tables appropriées de la base de données MySQL.

```
def insert_data(table_name, columns, data):
    placeholders = ", ".join(["%s"] * len(columns))
    column_names = ", ".join(columns)
    query = f"INSERT INTO {table_name} ({column_names}) VALUES ({placeholders})"
    cursor.executemany(query, data)
    conn.commit()
```

Il vérifie également que toutes les colonnes nécessaires sont présentes avant d'effectuer l'insertion.

8 - Fermeture de la connexion

Enfin, le script ferme la connexion à la base de données et affiche un message de succès.

III. Liste des bibliothèques utilisées

mysql.connector : Utilisée pour établir une connexion avec la base de données MySQL et exécuter des requêtes SQL.

Nous avons privilégié mysql.connector pour sa simplicité et son efficacité dans la connexion à des bases de données MySQL, offrant une interface directe pour l'exécution de requêtes SQL.

pandas : Utilisée pour la manipulation des données, notamment pour lire les fichiers CSV, fusionner des DataFrames, et effectuer des opérations de nettoyage et de transformation.

Pandas est largement reconnu pour sa puissance dans la manipulation et l'analyse de données, facilitant la lecture de fichiers CSV, la fusion de DataFrames et le nettoyage des données grâce à des méthodes intuitives, ce qui en a fait le candidat idéal pour notre projet.

numpy : Utilisée pour des opérations numériques, notamment pour gérer les valeurs manquantes et effectuer des calculs, il est donc nécessaire pour la transformation de certaines données.

glob : Utilisée pour récupérer des fichiers correspondant à un motif spécifique dans un répertoire.

Nous avons choisi glob pour sa capacité à rechercher facilement des fichiers dans un répertoire, ce qui simplifie le traitement de plusieurs fichiers CSV.

IV. Conclusion

Ce script est un outil relativement puissant pour le traitement et l'analyse des données relatives au diabète.

Il automatise le processus de nettoyage, d'enrichissement et d'insertion des données dans une base de données, ce qui permet de gagner du temps et d'assurer l'intégrité des données.