l'école d'ingénierie informatique

Rapport

TP Conception-Pipeline

Étudiants:

Deboub Maher

maher.deboub@ecoles.epsi.net

Naghmouchi Med Aziz

m.naghmouchi@ecoles-epsi.net

Intervenant

Marie Amina

22/05/2025

Explication du code ETL

Ce script met en place un **pipeline ETL (Extract - Transform - Load)** complet pour nettoyer et enrichir un fichier CSV de ventes.

Importations

```
import pandas as pd
import numpy as np
import logging
from sklearn.preprocessing import MinMaxScaler
```

- `pandas` : gestion des données tabulaires.
- `numpy` : opérations numériques.
- `logging` : journalisation des événements dans un fichier log.
- `MinMaxScaler` : normalisation des valeurs numériques entre 0 et 1.

Configuration des logs

```
logging.basicConfig(filename='etl.log', level=logging.INFO,
format='%(asctime)s - %(levelname)s - %(message)s')
```

- Configure un fichier de log nommé `etl.log` pour suivre les étapes de traitement.

Étape 1 : Chargement des données

```
def charger_donnees(fichier_csv):
```

- Charge le fichier CSV.
- Vérifie la présence des colonnes obligatoires.
- Gère les erreurs : fichier manquant, format invalide, parsing…

Erreurs gérées :

- `FileNotFoundError`
- `pd.errors.ParserError`
- Colonnes manquantes
- Toute autre exception

Étape 2 : Analyse exploratoire (EDA)

```
def analyse_donnees(df):
```

- Affiche un aperçu des données :
    - `head()`, `info()`, `isnull()`, `describe()` pour comprendre la structure.

Utilisé pour valider manuellement les données avant traitement.

Étape 3 : Traitement des valeurs manquantes

```
def traitement_valeurs_manquantes(df):
```

- Supprime les lignes sans valeurs dans les colonnes critiques (`dropna(subset=...)`).
- Remplir les quantités manquantes avec `0`.

- Remplir les prix manquants avec la **médiane**.

But : éviter les erreurs dans les calculs ultérieurs.

Étape 4 : Gestion des valeurs aberrantes

```
def gestion_valeurs_aberrantes(df):
```

- Supprime les lignes où `Quantite_vendue <= 0` ou `Prix_unitaire <= 0`
- Convertit `Date_vente` en format `datetime`, supprime les dates invalides (`NaT`)

Étape 5 : Suppression des doublons

```
def suppression_doublons(df):
```

- Supprime les lignes identiques pour éviter de fausser les statistiques.

Étape 6 : Transformations

```
def transformations(df):
```

1. Montant total de la vente

```
df['Montant_total'] = df['Quantite_vendue'] * df['Prix_unitaire']
```

2. Normalisation du prix unitaire

```
 scaler = MinMaxScaler()
    df['Prix_unitaire_normalisé'] =
scaler.fit_transform(df[['Prix_unitaire']]).round(3)
```

- Ramène toutes les valeurs entre 0 et 1 (arrondi à 3 décimales).

3. Extraction mois/année

```
df['Mois_vente'] = df['Date_vente'].dt.month
df['Annee_vente'] = df['Date_vente'].dt.year
```

- Pour analyse temporelle.

4. Nettoyage du nom produit

```
df['Nom_produit'] = df['Nom_produit'].str.strip().str.lower()
```

- Nettoie les espaces, harmonise la casse.

5. Catégorisation des montants

```
df['Categorie_montant'] = pd.cut(
        df['Montant_total'],
        bins=[-1, 50, 200, np.inf],
        labels=['faible', 'moyen', 'élevé']
    )
```

- Classe les montants en 3 niveaux.

6. Catégorisation des produits

```
def categorie_produit(nom):
    ...
df['Categorie_produit'] = df['Nom_produit'].apply(categorie_produit)
```

- Classe les produits en `habillement`, `électronique`, `autre` selon des mots-clés.

Étape 7 : Validation croisée

```
def validation_croisée(df):
```

- Vérifie que :
    - Aucune valeur manquante
    - Quantité > 0
    - `Date_vente` est bien de type `datetime`

Étape 8 : Sauvegarde des données

```
def sauvegarde(df, chemin_sortie):
```

- Sauvegarde au format CSV avec l'encodage `utf-8-sig` (utile pour bien gérer les accents dans Excel).

Fonction principale : `pipeline_etl`

```
def pipeline_etl(fichier_entree, fichier_sortie):
```

- Exécute toutes les étapes ci-dessus dans l'ordre logique.
- Journalise chaque étape.
- Gère les erreurs globales avec un bloc `try/except`.

Lancement du script

```
if __name__ == "__main__":
    pipeline_etl("ventes.csv", "vente_clean.csv")
```

- Point d'entrée du programme.

Conclusion

Ce pipeline :

- Nettoie les données
- Ajoute des colonnes utiles
- Catégorise et normalise
- Valide la qualité des données
- Gère toutes les erreurs critiques
- Documente tout dans un fichier `etl.log`
