Laboratoire pratique : IA générative pour les pipelines de données et les flux de travail ETL

Effort estimé : 30 minutes

Introduction

Une fois que vous avez nettoyé et préparé les données pertinentes pour le traitement, l'étape suivante pour tout ingénieur de données consiste à mettre en place un pipeline de données de sorte que les données enregistrées soient extraites, transformées dans un format préféré et chargées vers une destination selon les besoins. C'est ce qu'on appelle la mise en place d'un flux de travail ETL.

Dans ce laboratoire, vous apprendrez à utiliser l'IA générative pour créer des scripts Python capables de configurer un pipeline de workflow ETL complet pour le scénario donné.

Scénario

Vous êtes ingénieur de données pour une entreprise de soins de santé et vous avez été chargé de créer un pipeline de données qui capturera les informations enregistrées sur les paramètres de la maladie hépatique des patients, disponibles au format CSV dans les dossiers hospitaliers, ainsi que le pronostic de la maladie hépatique ou non du patient en question. Vous devez en outre transformer tous les attributs de l'ensemble de données en attributs numériques et charger les données finales dans une base de données SQL de l'équipe de science des données pour récupérer les données en vue d'un traitement ultérieur.

Objectifs

Dans ce laboratoire, vous apprendrez à utiliser l'IA générative pour créer des codes Python capables de :

- Extraire des informations d'un fichier CSV disponible sur une URL
- Transformer les données dans un format préféré
- Charger les données sous forme de table dans une base de données SQL

Ensemble de données

Pour les besoins de ce laboratoire, nous utilisons l' ensemble de données des patients indiens souffrant de troubles hépatiques , accessible au public sous la licence Creative Commons Attribution 4.0 International (CC BY 4.0) . Vous pouvez vous référer à la page Web de l'ensemble de données pour plus de détails sur les attributs.

L'ensemble de données est disponible pour une utilisation dans ce laboratoire à l'URL suivante.

```
URL = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m2/data/Ir
```

Interface de test

Vous trouverez une interface de test distincte à la fin de la leçon sur la page du cours. Veuillez garder cette interface de test ouverte en tant que laboratoire séparé sur le côté et suivre les étapes initiales pour être prêt avec la configuration.

Extraction de données

La première tâche consiste à extraire les données, disponibles sous forme de fichier CSV sur une URL, sous forme de trame de données dans votre code Python. Vous pouvez utiliser l'invite suivante pour générer votre code.

```
Write a Python code that reads a CSV file from a URL into a Pandas dataframe, and prints the first 5 entries to confirm it.
```

Vous pouvez vous attendre à voir le code suivant en guise de réponse.

```
import pandas as pd
url = "https://example.com/data.csv"
df = pd.read_csv(url)
print(df.head())
```

Vous pouvez coller le code généré par le modèle GenAI dans votre fichier dans l'environnement de test. Apportez-y les modifications nécessaires, comme la mise à jour de l'URL, et notez le résultat final. Il doit être comme indiqué ci-dessous.

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
   Age  Gender  Total_Bilirubin  ...  Albumin  Albumin and Globulin Ratio  Selector
0   65  Female              0.7  ...      3.3                        0.90         1
1   62    Male             10.9  ...      3.2                        0.74         1
2   62    Male              7.3  ...      3.3                        0.89         1
3   58    Male              1.0  ...      3.4                        1.00         1
4   72    Male              3.9  ...      2.4                        0.40         1

[5 rows x 11 columns]
```

Transformer les données

Comme nous l'avons vu à l'étape précédente, les données ont des attributs qui sont catégoriques et ne seront donc pas utilisables directement par l'équipe de science des données. Par conséquent, nous devons transformer ces attributs en entités numériques. Le moyen le plus simple d'effectuer cette tâche est d'utiliser le codage d'étiquettes. Vous pouvez utiliser la classe GenAI pour créer un code qui effectuera le codage d'étiquettes sur les attributs catégoriels.

Essayez l'invite suivante.

```
In a pandas dataframe, identify the categorical variables and perform label encoding on them
```

Vous pouvez vous attendre à un code comme indiqué ci-dessous.

```
from sklearn.preprocessing import LabelEncoder
# Assuming df is your Pandas dataframe
categorical_cols = df.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
# label_encoders dictionary stores the LabelEncoders for each categorical column
```

En apportant les modifications appropriées, vous pouvez ajouter cet extrait à votre code existant.

À titre d'exemple, le code final peut ressembler à celui présenté ci-dessous.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m2/data/Ir
df = pd.read_csv(url)
# Assuming df is your Pandas dataframe
categorical_cols = df.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
# label_encoders dictionary stores the LabelEncoders for each categorical column
print(df.head())
```

La sortie de ce code ressemblerait au code indiqué ci-dessous.

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
   Age  Gender  Total_Bilirubin  ...  Albumin  Albumin and Globulin Ratio  Selector
0   65       0              0.7  ...      3.3                        0.90         1
1   62       1             10.9  ...      3.2                        0.74         1
2   62       1              7.3  ...      3.3                        0.89         1
3   58       1              1.0  ...      3.4                        1.00         1
4   72       1              3.9  ...      2.4                        0.40         1
```

Chargement des données

Maintenant que vos données sont prêtes, vous devez les charger sous forme de table dans une base de données SQL. Pour ce laboratoire, nous pouvons simplement choisir de charger les données dans une base de données SQLite3. Supposons que les données doivent être chargées dans une base de données appelée « Patient_record » sous forme de table « Liver_patients ». Vous pouvez utiliser l'invite suivante pour créer un code qui crée ce code pour vous.

```
Write a python code that loads a python dataframe to an SQLite3 database named "Patient_record" as a table "Liver_patients". To conf
```

Vous pouvez vous attendre à voir un code comme indiqué ci-dessous.

```
import sqlite3
import pandas as pd
# Assuming df is your Pandas dataframe
conn = sqlite3.connect('Patient_record.db')
df.to_sql('Liver_patients', conn, index=False)
# Query the database to print the first entries in the table
query = "SELECT * FROM Liver_patients LIMIT 5"
result = pd.read_sql(query, conn)
print(result)
```

En apportant les mises à jour appropriées au code, vous pouvez mettre à jour votre code dans l'interface de test vers celui indiqué ci-dessous.

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
import sqlite3
url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m2/data/Ir
df = pd.read_csv(url)
# Assuming df is your Pandas dataframe
categorical_cols = df.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
#print(df.head())
# Assuming df is your Pandas dataframe
conn = sqlite3.connect('Patient_record.db')
df.to_sql('Liver_patients', conn, index=False)
# Query the database to print the first entries in the table
query = "SELECT * FROM Liver_patients LIMIT 5"
result = pd.read_sql(query, conn)
print(result)
```

Le résultat final sera le même que le résultat précédent, mais notez dans le code que l'instruction d'impression a maintenant la réponse de la requête au lieu du dataframe d'origine.

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
   Age  Gender  Total_Bilirubin  ...  Albumin  Albumin and Globulin Ratio  Selector
0   65       0              0.7  ...      3.3                        0.90         1
1   62       1             10.9  ...      3.2                        0.74         1
2   62       1              7.3  ...      3.3                        0.89         1
3   58       1              1.0  ...      3.4                        1.00         1
4   72       1              3.9  ...      2.4                        0.40         1
```

Conclusion

Félicitations pour avoir terminé ce laboratoire !

Vous savez maintenant comment utiliser l'IA générative pour :

- Créer un pipeline d'extraction de données
- Transformer les données dans un format préféré
- Charger les données dans une base de données

Auteur(s)

Abhishek Gagneja