Hands-on Lab: Generative AI for Data Pipelines and ETL Workflows

Estimated effort: 30 mins

Introduction

Once you have the relevant data cleaned and prepared for processing, the next step for any data engineer is to set up a data pipeline such that the recorded data is extracted, transformed into a preferred format, and loaded to a destination as per requirement. This is called setting up an ETL workflow.

In this lab, you will learn how to use generative AI for creating Python scripts that can set up an entire ETL workflow pipeline for the given scenario.

Scenario

You are a data engineer for a healthcare company and have been tasked with creating a data pipeline that will capture the recorded information about liver disease parameters for patients, available in a CSV format in the hospital records, along with the prognosis of whether the said patient had a liver disease or not. You are further required to transform all attributes of the data set into numerical attributes and load the final data into an SQL database of the data science team to pick up the data for further processing.

Objectives

In this lab, you will learn how to use generative AI to create Python codes that can:

- Extract information from a CSV file available on a URL
- Transform the data into a preferred format
- Load the data as a table to an SQL database

Data set

For the purpose of this lab, we are making use of the Indian Liver Patient Dataset, publically available under the Creative Commons Attribution 4.0 International (CC BY 4.0) license. You may refer to the data set webpage for more details on the attributes.

The data set is available for use in this lab at the following URL.

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

Testing Interface

You will find a separate testing interface at the end of the lesson on the course page. Please keep that testing interface open as a separate lab on the side and follow the initial steps to be ready with the setup.

Data Extraction

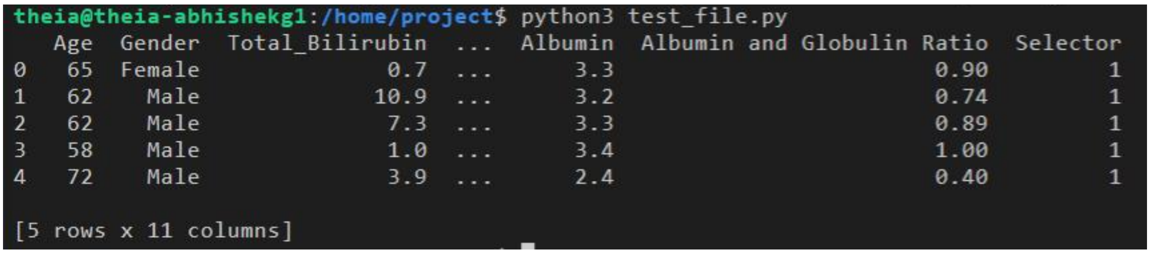
The first task is to extract the data, available as a CSV file on a URL, as a dataframe to your Python code. You can use the following prompt to generate your code.

```
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.
```

You can expect to see the following code as a response.

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

You can paste the code generated by the GenAI model to your file in the test environment. Make the required edits to it, like updating the URL, and note the final output. It should be as shown below.



```
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]
```

Transform Data

As was seen in the previous step, the data has some attributes which are categorical, and hence will not be usable by the data science team directly. Therefore, we must transform such attributes into numerical entities. The easiest way to do this task is using label encoding. You can use the GenAI classroom to create a code that will perform the label encoding on the categorical attributes.

Try the following prompt.

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

You can expect a code as shown below.

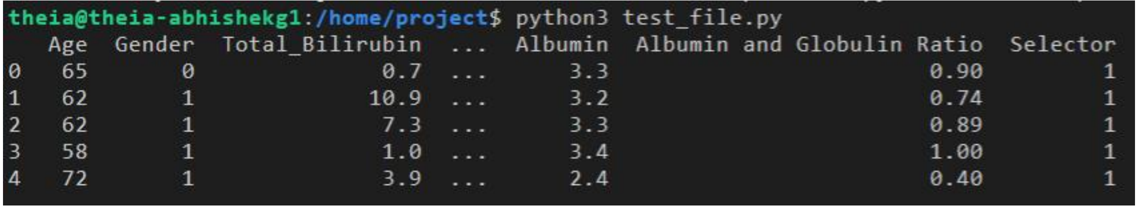
```
from sklearn.preprocessing import LabelEncoder
# Assuming df is your Pandas dataframe
categorical_cols = df.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le
# label_encoders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

Making the appropriate changes, you can add this snippet to your existing code.

As a sample, the final code can look like the one shown below.

```
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())
```

The output of this code would look like the code shown below.



```
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
```

Loading data

Now that your data is ready, you have to load it as a table to a SQL database. For this lab, we can simply choose to load the data to an SQLite3 database. Assume that the data needs to be loaded to a database called "Patient_record" as a table "Liver_patients". You can use the following prompt to create a code that creates this code for you.

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

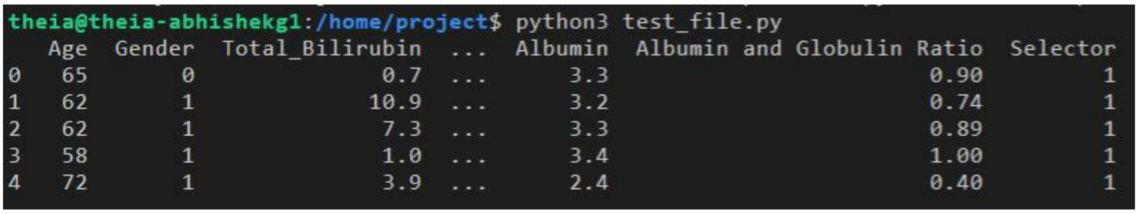
You can expect to see a code as shown below.

```
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)
```

Making appropriate updates to the code, you can update your code in the testing interface to the one shown below.

```
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/In
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)
```

The final result will be same as the previous result, but note in the code that the print statement now has the response of the query instead of the original dataframe.



```
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

Congratulations on completing this lab!

You now know how to use Generative AI to:

- Create a data extraction pipeline
- Transform the data into a preferred format
- Load the data to a database

Author(s)

Abhishek Gagneja