**Lab Exercise 19- EDA Data Pipeline in Metaflow**

**Step-by-Step Solution**

1. **Check Installation of Seaborn**: First, ensure that Seaborn is installed correctly. You can reinstall it using:

pip install matplotlib pandas seaborn metaflow

1. **Load Iris Dataset from a CSV File**: If loading the dataset directly from Seaborn is not working, we can download the Iris dataset as a CSV file and load it using Pandas. Here’s how to modify the pipeline:

**Updated EDA Pipeline to Load Iris Dataset from CSV**

You can download the Iris dataset from [this link](https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data) and save it as iris.csv. Alternatively, you can also use the built-in load\_iris() function from the sklearn.datasets library to load the dataset directly without needing a CSV file.

Here's how to update the pipeline using Pandas to read from a CSV file, assuming you have the iris.csv file in the same directory as your script.

**File: eda\_pipeline.py**

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from metaflow import FlowSpec, step, current

class EDAPipelineFlow(FlowSpec):

@step

def start(self):

"""

Step 1: Load the dataset from a CSV file.

Ensure the CSV file is available in the working directory.

"""

print("Loading the Iris dataset from CSV...")

self.data = pd.read\_csv('iris.csv', header=None,

names=['sepal\_length', 'sepal\_width',

'petal\_length', 'petal\_width', 'species'])

print(f"Dataset loaded with shape: {self.data.shape}")

self.next(self.data\_summary)

@step

def data\_summary(self):

"""

Step 2: Perform basic data profiling (summary statistics).

"""

print("Generating summary statistics...")

self.summary = self.data.describe()

self.missing\_values = self.data.isnull().sum()

print("Summary statistics:\n", self.summary)

print("Missing values:\n", self.missing\_values)

self.next(self.visualize\_data)

@step

def visualize\_data(self):

"""

Step 3: Visualize the data with histograms and pair plots.

- Histogram for each numeric column.

- Pairplot to visualize relationships between features.

"""

print("Generating data visualizations...")

# Histogram for each numeric column

for column in self.data.select\_dtypes(include=['float64', 'int64']).columns:

plt.figure(figsize=(8, 6))

sns.histplot(self.data[column], kde=True, color='skyblue')

plt.title(f'Histogram of {column}')

plt.xlabel(column)

plt.ylabel('Frequency')

hist\_filename = f'{current.flow\_name}\_{current.run\_id}\_{column}\_histogram.png'

plt.savefig(hist\_filename)

plt.close() # Close the figure to free memory

print(f"Saved histogram: {hist\_filename}")

# Pairplot to visualize feature relationships

pairplot\_filename = f'{current.flow\_name}\_{current.run\_id}\_pairplot.png'

sns.pairplot(self.data, hue='species')

plt.savefig(pairplot\_filename)

plt.close() # Close the figure to free memory

print(f"Saved pair plot: {pairplot\_filename}")

self.next(self.end)

@step

def end(self):

"""

Step 4: End the flow.

"""

print("EDA flow finished successfully!")

print(f"Summary Statistics:\n{self.summary}")

print(f"Missing Values:\n{self.missing\_values}")

if \_\_name\_\_ == '\_\_main\_\_':

EDAPipelineFlow()

**Key Changes Made:**

* **Loading from CSV**: The pipeline now loads the Iris dataset from a CSV file using pd.read\_csv(). Ensure you have the iris.csv file in the same directory as this script.
* **Closing Plots**: Used plt.close() after saving each plot to free up memory.

**Running the Pipeline**

Once you have made these changes and ensured that the CSV file is in place, you can run the pipeline:

python eda\_pipeline.py run