Practical-2

AIM : Introduction to reproducible Machine Learning Operations

The aim of the practial is to get the hands-on experience of reproducing the machine learning operations at each stage. Student needs to apply the following steps in the practical.

Step 1 :

Ensure that the numpy, scikit learn, and matplotlib libraries are available in your system. Create the requirements.txt file and make a note of the versions of these libraries.

To check the Versions of Libraries :
pip show numpy
pip show scikit-learn
pip show matplotlib

```
PS E:\7sem\MLOPS\practicals> pip show numpy
Name: numpy
Version: 1.25.1
Summary: Fundamental package for array computing in Python
Home-page: https://www.numpy.org
Author: Travis E. Oliphant et al.
Author-email:
License: BSD-3-Clause
PS E:\7sem\MLOPS\practicals> pip show scikit-learn
Name: scikit-learn
Version: 1.3.0
Summary: A set of python modules for machine learning and data mining
Home-page: http://scikit-learn.org
Author:
Author-email:
License: new BSD
PS E:\7sem\MLOPS\practicals> pip show matplotlib
Name: matplotlib
Version: 3.7.2
Summary: Python plotting package
Home-page: https://matplotlib.org
Author: John D. Hunter, Michael Droettboom
Author-email: matplotlib-users@python.org
License: PSF
```

Create requirements.txt:

requirements.txt

```
python==3.10.9
numpy==1.25.0
pandas==1.5.3
scikit-learn==1.3.0
matplotlib==3.7.1
pickle==4.0
```

Step 2 :

a) **Import the data and scale it using StandardScaler:**

```
import numpy as np
from sklearn.preprocessing import StandardScaler
import joblib

# Load the data from sample.csv using np.genfromtxt
data = np.genfromtxt('sample.csv', delimiter=',')
# # Load the data from Sample.txt
# data = np.loadtxt('sample.csv')

# Create a StandardScaler object and fit_transform the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(data)

# Store the scaler object for reproducibility
joblib.dump(scaler, 'scaler_object.joblib')

# Now scaled_data contains the normalized dataset
```

b) Splitting the normalized data:

```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
train_data, test_data = train_test_split(scaled_data, test_size=0.2,
random_state=42)

# train_data and test_data now contain the training and testing datasets
```

c) Storing the snapshot of the data as a numpy file:

```
# Save the datasets as numpy files
np.save('train_data.npy', train_data)
np.save('test_data.npy', test_data)
```

Step 3 :

Apply the linear regression algorithm on the dataset and assess the prediction on the test dataset.

a) Store the trained model into the local file system to ensure the reproducibility of the
prediction. Import the model and the test dataset into other python file. Check whether the same prediction is obtained in the latter case.

```
from sklearn.linear_model import LinearRegression
import joblib

# Assume you already have the train_data and test_data loaded

# Separate features and target variable
X_train = train_data[:, :-1]
```

```
y_train = train_data[:, -1]

# Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Assess the prediction on the test dataset
X_test = test_data[:, :-1]
y_test = test_data[:, -1]

# Make predictions
predictions = model.predict(X_test)

# Store the trained model for reproducibility
joblib.dump(model, 'linear_regression_model.joblib')

# Now, in another Python file, you can load the model and test data for
prediction
# Load the model
loaded_model = joblib.load('linear_regression_model.joblib')

# Make predictions on the test data
loaded_predictions = loaded_model.predict(X_test)

# Check if the predictions are the same
if np.array_equal(predictions, loaded_predictions):
    print("Output : Predictions match!")
else:
    print("Output : Predictions differ!")
```

Output :

```
[9]   ✓ 0.1s

...   Output : Predictions match!
```