Practical-2

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.

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.

```
[ ] import numpy as np
    np.__version__

    '1.23.5'

[ ] import sklearn
    sklearn.__version__

    '1.2.2'

[ ] import matplotlib as plt
    plt.__version__

    '3.7.1'

[ ] import pandas as pd
    pd.__version__

    '1.5.3'
```

2. Write a python code to import the Sample.txt data. Further, apply the following processes on the imported data.
   a) Scale the dataset. Use the StandardScalar function of scikit learn to normalize the dataset. Ensure reproduciblity: Store the standard scalar object into your local file system, so that the same data normalization can be applied to the other data during the deployement.
   b) Split the data: Write the python code to split the normalized data into randomly selected proportion as per the constant ratio. For e.g., if the constant ratio is 0.8, then the code must randomly select the 80 % proportion of the data as the training dataset and remaining 20 % as the testing dataset.
   c) Store the snapshot of the data as the numpy file. Ensure the same dataset could be loaded into separate python code. (Extra question: how to ensure that the same random generation could be achieved on each execution.)

Normalization

```
[ ] from sklearn.preprocessing import StandardScaler
```

```
[ ] #converting pd series to numpy array
    x_train_reshape = np.asarray(x_train).reshape(-1,1)
    x_test_reshape = np.asarray(x_test).reshape(-1,1)
```

```
    scaler = StandardScaler()
    scaler.fit(x_train_reshape)
    x_train_normalized = scaler.transform(x_train_reshape)
    x_test_normalized = scaler.transform(x_test_reshape)
```

```
[ ] #store the scaler object
    import pickle
    with open(os.path.join(drivefolder,"scaler.pkl"),'wb') as f:
      pickle.dump(scaler,f)
```

```
x_train_normalized = scaler.transform(x_train_reshape)
x_train_normalized
```

```
array([[ 3.84168192e-01],
       [ 9.29180700e-01],
       [-6.07754573e-01],
```

```
    x_test_normalized = scaler.transform(x_test_reshape)
    x_test_normalized
```

```
    array([[ 0.33875048],
           [ 0.61125674],
```

```
[ ] np.mean(x_test_normalized)

    -0.008182479453515227
```

```
[ ] np.mean(x_train_normalized)

    1.2222638803212733e-16
```

```
[ ] np.std(x_test_normalized)

    0.9210591725069316
```

```
[ ] np.std(x_train_normalized)

    1.0000000000000002
```

Training

```
[ ] from sklearn.linear_model import LinearRegression
```

```
[ ] reg = LinearRegression()
    reg.fit(x_train_normalized, y_train)
    Score = reg.score(x_train_normalized, y_train)
    CoEfficient = reg.coef_
    Intercept = reg.intercept_
```

Testing

```
Predictions = reg.predict(x_test_normalized)
```

```
[ ] np.save(os.path.join(drivefolder,"Predictions"),Predictions)
```

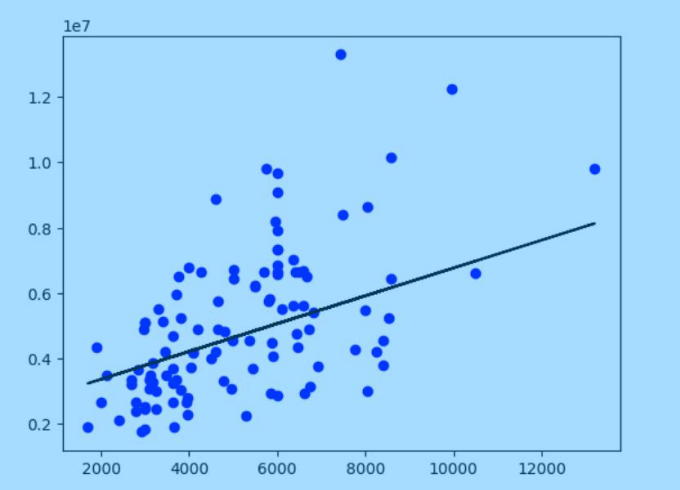
```
[ ] from sklearn.metrics import mean_absolute_error,mean_squared_error
```

```
[ ] mae = mean_absolute_error(y_test, Predictions)

    #squared True returns MSE value, False returns RMSE value.
    mse = mean_squared_error(y_test, Predictions) #default=True
    rmse = mean_squared_error(y_test, Predictions,squared=False)
    print("MAE:",mae)
    print("MSE:",mse)
    print("RMSE:",rmse)
```

```
MAE: 1474748.1337969352
MSE: 3675286604768.1855
RMSE: 1917103.7021424235
```

```
import matplotlib.pyplot as plt
plt.scatter(x_test, y_test, color ='b')
plt.plot(x_test, Predictions, color ='k')
plt.show()
```



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.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    import pickle
    import numpy as np
```

```
[ ] with open('/content/drive/MyDrive/SEM 7/ML - Ops/scaler.pkl','rb') as f:
      scaler = pickle.load(f)
```

```
    x_test = np.load('/content/drive/MyDrive/SEM 7/ML - Ops/x_test.npy', allow_pickle=True)
```

```
[ ] x_train_normalized_Main = np.load('/content/drive/MyDrive/SEM 7/ML - Ops/x_train_normalized.npy', allow_pickle=True)
    x_test_normalized_Main = np.load('/content/drive/MyDrive/SEM 7/ML - Ops/x_test_normalized.npy' ,allow_pickle=True)
```

```
[ ] '''Converting pd series to numpy array'''
    x_test = np.asarray(x_test).reshape(-1,1)
```

```
[ ] x_test_normalized = scaler.transform(x_test)
```

```
[ ] x_test_normalized
```

```
array([[ 0.33875048],
       [ 0.61125674],
       [-0.5060189 ],
       [-0.0700089 ],
       [-0.54235307],
       [ 0.7111757 ],
```

```
[ ] np.mean(x_test_normalized)
```

```
-0.008182479453515227
```

```
[ ] np.std(x_test_normalized)
```

```
0.9210591725069316
```

```
    x_test
```

```
array([[ 5900],
       [ 6500],
       [ 4040],
       [ 5000],
       [ 3960],
       [ 6720],
       [ 8520],
```

```
x_test_normalized_Main == x_test_normalized
```

```
array([[ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
```

```
predictions = np.load('/content/drive/MyDrive/SEM 7/ML - Ops/Predictions.npy', allow_pickle=True)
```

```
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

```
y_test = np.load('/content/drive/MyDrive/SEM 7/ML - Ops/y_test.npy', allow_pickle=True)
```

```
y_test.shape
```

```
(109,)
```

```
mae = mean_absolute_error(y_test, predictions)

#squared True returns MSE value, False returns RMSE value.
mse = mean_squared_error(y_test, predictions) #default=True
rmse = mean_squared_error(y_test, predictions,squared=False)
print("MAE:",mae)
print("MSE:",mse)
print("RMSE:",rmse)
```

```
MAE: 1474748.1337969352
MSE: 3675286604768.1855
RMSE: 1917103.7021424235
```