**📥 Step 1: Importing Libraries**

First, the necessary libraries are imported:

from sklearn.model\_selection import train\_test\_split

import torch

from torch.utils.data import DataLoader, TensorDataset

import torch.nn as nn

import torch.optim as optim

import pandas as pd

from sklearn.preprocessing import MinMaxScaler, OneHotEncoder

**✅ Explanation:**

1. **train\_test\_split** → From scikit-learn, used to split data into training and testing sets.
2. **torch** → Main PyTorch library for handling tensors (multidimensional arrays) and neural network operations.
3. **DataLoader** → Handles batching and shuffling of data for training.
4. **TensorDataset** → Wraps tensors into a dataset object that can be loaded by DataLoader.
5. **nn** → Module for building neural networks.
6. **optim** → Contains optimization algorithms (like Adam).
7. **pandas** → Used to handle and process tabular data.
8. **MinMaxScaler** → Scales numeric data between 0 and 1.
9. **OneHotEncoder** → Encodes categorical data into a numerical format.

**📊 Step 2: Load and Explore the Dataset**

file\_path = 'resource/dataset.csv' # Update with the actual path

data = pd.read\_csv(file\_path)

# Display the first few rows to understand the structure

data.head()

# Display column names and data types

print(data.info())

# Show basic statistics

print(data.describe())

**✅ Explanation:**

* **pd.read\_csv** → Loads a CSV file into a pandas DataFrame.
* **data.head()** → Displays the first five rows of the dataset.
* **data.info()** → Shows column names, data types, and missing values.
* **data.describe()** → Provides summary statistics (mean, standard deviation, etc.) for numerical columns.

**✅ Example:**

| **Year** | **Month** | **Solar Energy** | **Wind Energy** | **Biomass Energy** | **Total Renewable Energy** |
| --- | --- | --- | --- | --- | --- |
| 2025 | 5 | 200 | 150 | 100 | 500 |

**✂️ Step 3: Handle Missing Values**

numeric\_cols = data.select\_dtypes(include=['number']).columns

data[numeric\_cols] = data[numeric\_cols].fillna(data[numeric\_cols].mean())

**✅ Explanation:**

* **select\_dtypes(include=['number'])** → Selects numeric columns.
* **fillna(mean())** → Fills missing values with the mean of each column.

**🏷️ Step 4: Scale and Encode Data**

scaler = MinMaxScaler()

data\_scaled = pd.DataFrame(scaler.fit\_transform(data[numeric\_cols]), columns=numeric\_cols)

encoder = OneHotEncoder(sparse=False)

encoded\_cols = encoder.fit\_transform(data[['Sector']])

data\_scaled = pd.concat([data\_scaled, pd.DataFrame(encoded\_cols)], axis=1)

**✅ Explanation:**

* **MinMaxScaler** → Scales numeric data between 0 and 1.
* **fit\_transform** → Fits the scaler to the data and transforms it.
* **OneHotEncoder** → Converts categorical data into numerical format.

**✅ Example:**

* Sector = 'Residential' → [0, 1, 0, 0]

**🏋️‍♂️ Step 5: Define Input and Output Variables**

X = data\_scaled[['Year', 'Month', 'Solar Energy', 'Wind Energy', 'Biomass Energy', 'Sector\_Electric Power', 'Sector\_Industrial', 'Sector\_Residential', 'Sector\_Transportation']]

y = data\_scaled['Total Renewable Energy']

**✅ Explanation:**

* X → Independent variables.
* y → Dependent variable (target).

**🔄 Step 6: Split Data into Training and Testing Sets**

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**✅ Explanation:**

* test\_size=0.2 → 20% data for testing.
* random\_state=42 → Ensures consistent splits.

**🔄 Step 7: Convert Data to PyTorch Tensors**

X\_train\_tensor = torch.FloatTensor(X\_train.values)

y\_train\_tensor = torch.FloatTensor(y\_train.values).reshape(-1, 1)

X\_test\_tensor = torch.FloatTensor(X\_test.values)

y\_test\_tensor = torch.FloatTensor(y\_test.values).reshape(-1, 1)

**✅ Explanation:**

* Converts data into tensor format.
* .reshape(-1, 1) ensures correct shape for regression.

**🏷️ Step 8: Create Tensor Dataset and DataLoader**

train\_dataset = TensorDataset(X\_train\_tensor, y\_train\_tensor)

train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)

**✅ Explanation:**

* batch\_size=32 → 32 samples per batch.
* shuffle=True → Shuffles data during training.

**🏋️‍♂️ Step 9: Define Neural Network Model**

class RenewableEnergyModel(nn.Module):

def \_\_init\_\_(self):

super(RenewableEnergyModel, self).\_\_init\_\_()

self.layer1 = nn.Linear(9, 64)

self.layer2 = nn.Linear(64, 32)

self.layer3 = nn.Linear(32, 1)

def forward(self, x):

x = torch.relu(self.layer1(x))

x = torch.relu(self.layer2(x))

x = self.layer3(x)

return x

**✅ Explanation:**

* First layer → 9 input features → 64 neurons.
* Second layer → 64 neurons → 32 neurons.
* Output layer → 32 neurons → 1 output.
* torch.relu → Activation function.

🔥 This model is now ready for training! 😎

### If u need more explanation refer this link :

https://chatgpt.com/share/67e28f88-e380-800a-bd61-e5a82ef456b9