IMPORTING THE REQUIRED LIBRARIES:

Additionally, if you plan to use advanced machine learning techniques, you might consider importing libraries like scikit-learn, TensorFlow, or Porch for building more complex models.

```
Import pandas as pd

Import numpy as np

Import matplotlib.pyplot as plt

Import seaborn as sns

From sklearn.model_selection import train_test_split

From sklearn.preprocessing import StandardScaler

From sklearn.linear_model import LinearRegression

From sklearn.metrics import mean_squared_error, r2_score
```

IMPORTING THE DATASET (READ DATA SET; CREATE MATRIX):

Make sure you have the file ‘your_dataset.csv’ in the same directory as your Python script or provide the full path if it’s located elsewhere. This will load the data into a pandas DataFrame, which you can then use for data manipulation and analysis

```
Import pandas as pd
```

```
# Replace ‘your_dataset.csv’ with your actual dataset’s filename

Df = pd.read_csv(‘your_dataset.csv’)

# Now you can use df to perform further data preprocessing and analysis
```

HANDLING THE MISSING DATA:

The sklearn.preprocessing library in Python includes a class called Simple Imputer that can be used to handle missing data. It is used for completing missing values in a dataset. Here is a basic example of how you can use it:

```
From sklearn.impute import Simple Imputer

Import pandas as pd
```

# Load your dataset

Data = pd.read_csv('path_to_your_dataset.csv')

# INITIALIZE THE IMPUTER

Imputer = Simple Imputer (missing_values=np.nan, strategy='mean') # You can use 'median', 'most_frequent' or 'constant' as strategy

# FIT AND TRANSFORM THE DATA

Data_imputed = imputer.fit_transform(data)

In this case, strategy can be set to 'mean', 'median', 'most_frequent', or 'constant' depending on how you want to impute the missing values.

ENCODING THE CATEGORICAL DATA:

To encode categorical data in Python, you can use techniques like Label Encoding and One-Hot Encoding. Here's an example using the pandas library:

Import pandas as pd

Load the dataset

Data = pd.read_csv("path_to_your_dataset.csv")

Perform Label Encoding for a single categorical column 'red'

From sklearn.preprocessing import Label Encoder

Label_encoder = LabelEncoder()

Data['red'] = label_encoder.fit_transform(data['red'])

Alternatively, for One-Hot Encoding

# data = pd.get_dummies(data, columns=['red'])

Now 'red' column has been encoded. You can use the updated dataframe for further analysis or modelling.

Make sure to replace "path_to_your_dataset.csv" with the actual path to your dataset file.

FEATURE SCALING AND IMPORT STANDARDSCALER:

To perform feature scaling and import the StandardScaler in Python, you can use the following code:

Python

Import pandas as pd

From sklearn.preprocessing import StandardScaler

```
# Load your dataset using pandas

Data = pd.read_csv(“path_to_your_dataset.csv”)
```

```
# Perform feature scaling

Scaler = StandardScaler()

Scaled_data = scaler.fit_transform(data)
```

```
# THE SCALED_DATA WILL BE YOUR SCALED DATASET
```

Make sure to replace “path_to_your_dataset.csv” with the actual path to your dataset. This code will import the necessary libraries, perform feature scaling using the StandardScaler, and then transform your data accordingly.

SPLITTING THE DATA SET:

To split a dataset into a training set and a test set, you can use Python libraries such as scikit-learn. Here’s an example of how to do it:

Python

Copy code

```
Import pandas as pd

From sklearn.model_selection import train_test_split

# Load your dataset (replace ‘your_dataset.csv’ with the actual file path)

Data = pd.read_csv(‘your_dataset.csv’)

# Specify the features (X) and the target variable (y)

X = data.drop(columns=[‘target_column_name’]) # Replace ‘target_column_name’ with the actual name
of the target variable
```

Y = data[‘target_column_name’]

# Split the dataset into a training set and a test set (adjust the test_size as needed)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In this code:

Replace ‘your_dataset.csv’ with the path to your dataset file.

Replace ‘target_column_name’ with the actual name of the column you want to predict (target variable).

You can adjust the test_size parameter to determine the proportion of data you want to allocate to the test set.

This code will split your dataset into training and test sets, and you can then use them for building and evaluating your predictive models

IMPORTING THE DATASET (READ DATA SET; CREATE MATRIX):

It seems like you are trying to import a dataset for electricity price prediction from Kaggle. To read the dataset and create a matrix, you can use Python and libraries such as pandas and numpy. Here's an example of how you can do it:

```
Import pandas as pd
```

Assuming you have downloaded the dataset and placed it in the current working directory

```
File_path = 'path_to_your_dataset.csv' # Replace 'path_to_your_dataset.csv' with the actual path of your dataset
```

Read the dataset

```
Data = pd.read_csv(file_path)
```

Creating a matrix

```
Matrix = data.values
```

Make sure to replace ‘path_to_your_dataset.csv’ with the actual path of your dataset. Also, ensure you have the necessary libraries installed.