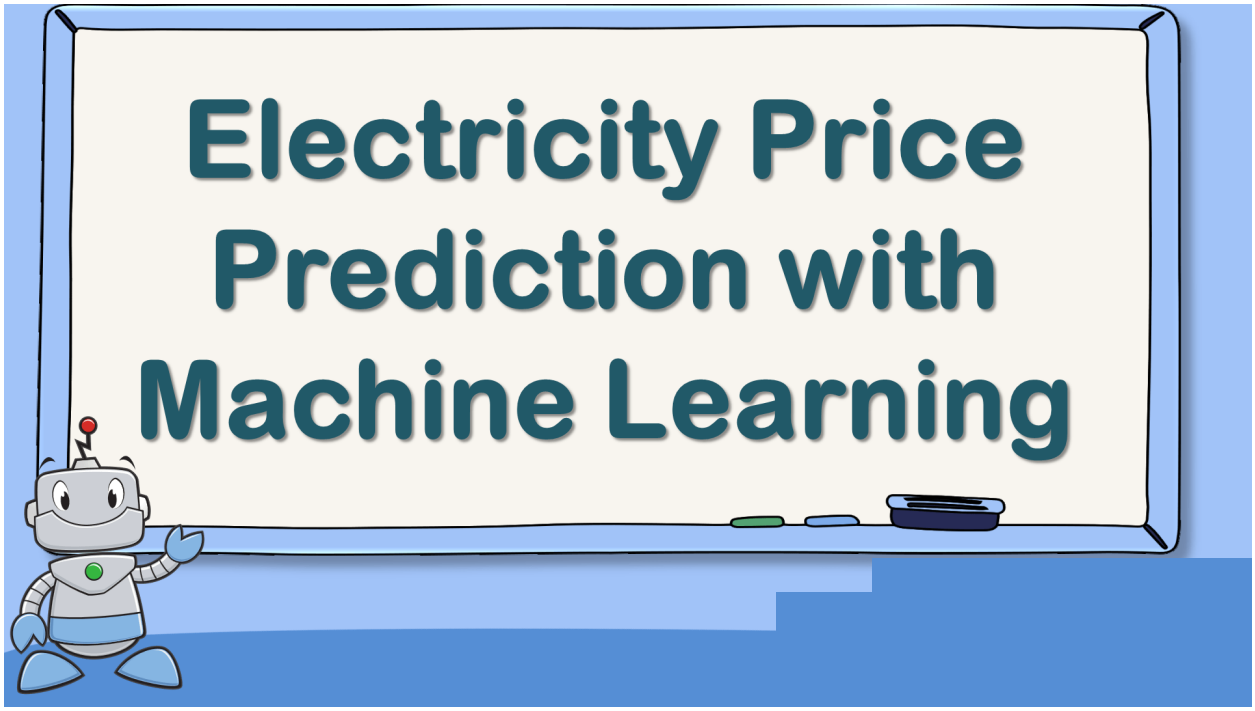
ELECTRICITY PRICES PREDICTION



Phase-2DocumentSubmission

OBJECTIVE:

The price of electricity depends on many factors. Predicting the price of electricity helps to understand how much electricity they have to pay each year. The Electricity Price Prediction task is based on a case study where you need to predict the daily price of electricity based on the daily consumption. We use the datascience with machine learning for the electricity price prediction.

1.DATASET INFORMATION:

- ✓ **A data source for electricity prices prediction using applied data science should be accurate and complete.**

Dataset Link:

https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction

Columns Information:

Information on columns

```
In [2]: data_frame = pd.read_csv("Electricity.csv")
```

```
C:\Users\linux6\AppData\Local\Temp\ipykernel_1440\4176279552.py:1: DtypeWarning: Columns (9,10,11,14,15,16,17) have mixed type
s. Specify dtype option on import or set low_memory=False.
  data_frame = pd.read_csv("Electricity.csv")
```

```
In [3]: data_frame.head()
```

Out[3]:

	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Intensity	ActualWindProduction	SystemLoadEP2	SMPEP2
0	0	315.31	3388.77	49.26	6.00	9.30	600.71	356.00	3159.60	54.32
1	1	321.80	3196.66	49.26	6.00	11.10	605.42	317.00	2973.01	54.23
2	2	328.57	3060.71	49.10	5.00	11.10	589.97	311.00	2834.00	54.23
3	3	335.60	2945.56	48.04	6.00	9.30	585.94	313.00	2725.99	53.47
4	4	342.90	2849.34	33.75	6.00	11.10	571.52	346.00	2655.64	39.87

```
In [5]: data_frame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38014 entries, 0 to 38013
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   DateTime               38014 non-null  object
 1   Holiday                1536 non-null   object
 2   HolidayFlag            38014 non-null  int64
 3   DayOfWeek              38014 non-null  int64
 4   WeekOfYear             38014 non-null  int64
 5   Day                    38014 non-null  int64
 6   Month                  38014 non-null  int64
 7   Year                   38014 non-null  int64
 8   PeriodOfDay            38014 non-null  int64
 9   ForecastWindProduction 38014 non-null  object
 10  SystemLoadEA           38014 non-null  object
 11  SMPEA                  38014 non-null  object
 12  ORKTemperature         38014 non-null  object
 13  ORKWindspeed           38014 non-null  object
 14  CO2Intensity           38014 non-null  object
 15  ActualWindProduction   38014 non-null  object
 16  SystemLoadEP2          38014 non-null  object
 17  SMPEP2                 38014 non-null  object
dtypes: int64(7), object(11)
memory usage: 5.2+ MB
```

Selecting Features And Target Variables:

Selecting features and Target Variables

Feature variables

```
In [ ]: x = data_[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",
                 "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
                "ActualWindProduction", "SystemLoadEP2"]]
```

Target variable

```
In [ ]: y = data["SMPEP2"]
```

2.DATA PREPROCESSING:

- ✓ **Clean the data by handling missing values, outliers, andcategorical variables. Standardize or normalize numerical features.**

Removing All The Missing Values:

Data preprocessing

1.Removing all the missing values

```
In [7]: #checking for missing values

        data_frame.isnull().sum()
```

```
Out[7]: DateTime                    0
        Holiday                 36478
        HolidayFlag                 0
        DayOfWeek                   0
        WeekOfYear                  0
        Day                         0
        Month                       0
        Year                        0
        PeriodOfDay                 0
        ForecastWindProduction      0
        SystemLoadEA                0
        SMPEA                       0
        ORKTemperature              0
        ORKWindspeed                0
        CO2Intensity                0
        ActualWindProduction        0
        SystemLoadEP2               0
        SMPEP2                      0
        dtype: int64
```

```
In [14]: #removing all missing values
         data_frame = data_frame.dropna()
```

Converting All Non-Numerical Values Into Numerical Values:

2.Converting all non numerical values into numerical values

In [15]:
```
data_frame["ForecastWindProduction"] = pd.to_numeric(data_frame["ForecastWindProduction"], errors= 'coerce')
data_frame["SystemLoadEA"] = pd.to_numeric(data_frame["SystemLoadEA"], errors= 'coerce')
data_frame["SMPEA"] = pd.to_numeric(data_frame["SMPEA"], errors= 'coerce')
data_frame["ORKTemperature"] = pd.to_numeric(data_frame["ORKTemperature"], errors= 'coerce')
data_frame["ORKWindspeed"] = pd.to_numeric(data_frame["ORKWindspeed"], errors= 'coerce')
data_frame["CO2Intensity"] = pd.to_numeric(data_frame["CO2Intensity"], errors= 'coerce')
data_frame["ActualWindProduction"] = pd.to_numeric(data_frame["ActualWindProduction"], errors= 'coerce')
data_frame["SystemLoadEP2"] = pd.to_numeric(data_frame["SystemLoadEP2"], errors= 'coerce')
data_frame["SMPEP2"] = pd.to_numeric(data_frame["SMPEP2"], errors= 'coerce')
```

Splitting Data Into Features And Labels:

3.Splitting our data into features and labels

In [16]:
```
x = data_frame[["Day", "Month", "ForecastWindProduction", "SystemLoadEA",
          "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity",
          "ActualWindProduction", "SystemLoadEP2"]]

y = data_frame["SMPEP2"]
```

Splitting Data Into Training And Test Set:

4.Splitting our data into training and test set

In [17]:
```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
```

3. SUPERVISED ALGORITHM:

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. Here we use the Random Forest Regression Algorithm.

RANDOM FOREST REGRESSION ALGORITHM:

Random forest regression is a supervised learning algorithm and bagging technique that uses an ensemble learning method for regression in machine learning. The trees in random forests run in parallel, meaning there is no interaction between these trees while building the trees.

4.TRAINING DATA SET USING SUPERVISED ALGORITHM:

Fitting Training Data Into Machine Learning Model:

Training our data using our machine learning model

1.Fitting our training data into machine learining model

```
In [18]: from sklearn.ensemble import RandomForestRegressor
         model = RandomForestRegressor()
         model.fit(xtrain, ytrain)

Out[18]: RandomForestRegressor
         RandomForestRegressor()
```

Testing Our Test Set Using Machine Learning Model:

2.Testing our test set using our machine learninig model

```
In [20]: #creating y preds
         ypreds = model.predict(xtest)

In [21]: #checking our machine learning models score

         model.score(xtest,ytest)

Out[21]: 0.6912176354192148
```

5.MODEL EVALUATION:

Finding R2 Score:

Evaluating our machne learning model

1.R2 score

In [26]:
```
#importing r2 library
from sklearn.metrics import r2_score

#checking r2 score
r2 = r2_score(ytest,ypreds)

print(f"The r2 score of our machine learning model is {r2}")
```

```
The r2 score of our machine learning model is 0.6912176354192148
```

Finding Mean Absolute Error:

2.Mean absolute error

In [27]:
```
#importing MAE library
from sklearn.metrics import mean_absolute_error

#checking mean absolute error
MAE = mean_absolute_error(ytest,ypreds)

print(f"The mean_absolute_error of our machine learning model is {MAE}")
```

```
The mean_absolute_error of our machine learning model is 8.516964788732393
```

Finding Mean Squared Error:

3.Mean squared error

In [28]:
```
#importing MSE library
from sklearn.metrics import mean_squared_error

#checking mean squared error
MSE = mean_squared_error(ytest,ypreds)

print(f"The mean_squared_error of our machine learning model is {MSE}")
```

```
The mean_squared_error of our machine learning model is 363.7967453705629
```