**ELECTRICITY PRICES PREDICTION**

**PROJECT DOCUMENTATION**

**TEAM MEMBERS**

**1.ALTHAF M (TL)**

**2.NARESH KUMAR V**

**3.JAWAGARNATH S T**

**4.ANEES HUSSAIN A**

**PHASE 5: Project Documentation and Submission**

**PROJECT DEFINITION:**

The goal of this project is to develop a predictive model for electricity pricing. The provided dataset “Electricity\_Pricing.csv” contains various features related to electricity production and consumption, along with the corresponding pricing information. The dataset includes the following columns:

**INTRODUCTION:**

In the following sections we will delve into the essential steps of this predictive modeling endeavor,from data collection and prepration to model selection and evaluations. By the end you will have a coimprehensive understanding of the methodology and tools required to coimprehensive understanding of the methodology and tools required to develop a reliable electricity price forecasting system.develop a reliable electricity price forecasting system. In the following sections we will delve into the essential steps of this predictive modeling endeavor,from data collection and prepration to model selection and evaluations.

**Dataset Description:-**

1.Day: Day of the month (1-31)

2.Month: Month of the year (1-12).

3.ForecastWindProduction: Forecasted wind production in a specific unit.

4.SystemLoadEA: Electricity system load for a particular area.

5.SMPEA: Smart Market Price for Electricity A.

6.ORKTemperature: Temperature measured in the ORK region.

7.ORKWindspeed: Windspeed recorded in the ORK region.

8.CO2Intensity: CO2 intensity data.

9.ActualWindProduction: Actual wind production data.

10.SystemLoadEP2: Electricity system load for another specific area.

**Procedure:-**

1. Performs data preprocessing steps, including handling missing values and converting certain columns to numeric data types.
2. Creates visualizations such as a bar plot showing the relationship between the “Month” column and the target variable “SMPEP2”, as well as a heatmap showing the correlations between different features in the dataset.
3. Splits the data into training and testing sets.
4. Reads data from a CSV file named “Electricity\_Pricing.csv”.
5. Uses a Random Forest Regressor model to train on the training data.
6. Finally, the code makes a prediction for the target variable “SMPEP2” based on a specific set of feature values.

**Packages installed:-**

* Pandas: Pandas is a popular Python library used for data manipulation and analysis. It provides data structures and functions to make data analysis fast and easy.
* NumPy: NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
* Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It can be used to generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc.
* Seaborn: Seaborn is a data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
* Scikit-learn: Scikit-learn is a widely used machine learning library in Python. It provides simple and efficient tools for data mining and data analysis. In this project, it is used for tasks such as data splitting, model training, and model evaluation.

**Algorithm Choice:-**

For time series forecasting, the choice of algorithm often depends on the specific characteristics of the data and the problem at hand. Some commonly used algorithms for time series forecasting include:

* ARIMA (AutoRegressive Integrated Moving Average): Suitable for univariate time series data, ARIMA models capture different aspects of time series data, including trends and seasonality.
* Prophet: Developed by Facebook, Prophet is designed for time series forecasting with added flexibility for handling various seasonal patterns and holidays.
* LSTM (Long Short-Term Memory) Networks: A type of recurrent neural network (RNN), LSTM networks are suitable for capturing long-term dependencies in time series data and are particularly effective for sequential data**.**

**Evaluation metrics:-**

* When evaluating the performance of a time series forecasting model, it is crucial to use appropriate evaluation metrics to assess the model’s accuracy and predictive capabilities. Some common evaluation metrics for time series forecasting include:
* Mean Absolute Error (MAE): Calculates the average of the absolute differences between predicted and actual values, providing a measure of the model’s forecasting accuracy.
* Mean Squared Error (MSE): Computes the average of the squared differences between predicted and actual values, giving more weight to large errors compared to MAE.
* Root Mean Squared Error (RMSE): RMSE is the square root of MSE and provides an interpretable measure of the model’s forecasting accuracy in the same unit as the target variable.
* Mean Absolute Percentage Error (MAPE): Measures the percentage difference between predicted and actual values, providing insights into the relative forecasting error.
* R-squared (R2): Determines the proportion of the variance in the dependent variable that is predictable from the independent variables, offering a measure of how well the model fits the data.

**PHASE 1:** PROBLEM DEFINITION AND DESIGN THINKING

**Data Loading Process:-**

* Importing Libraries: The code begins with the import of necessary libraries, including Pandas and NumPy.
* Loading the CSV File: The line data=pd.read\_csv(“Electricity\_Pricing.csv”) loads the data from the “Electricity\_Pricing.csv” file and stores it in a variable named data.
* Displaying the Data: The line print(data.head()) is used to display the first few rows of the DataFrame, giving a quick overview of the data’s structure and contents.
* Examining Data Information: The line data.info() provides a concise summary of the DataFrame, including the number of non-null entries in each column and the data types.

**Data Preprocessing:-**

* Handling Missing Values: The code uses the pd.to\_numeric() function with the parameter errors=’coerce’ to convert specific columns, such as “ForecastWindProduction,” “SystemLoadEA,” “SMPEA,” “ORKTemperature,” “ORKWindspeed,” “CO2Intensity,” “ActualWindProduction,” and “SystemLoadEP2,” to numeric data types. The errors=’coerce’ parameter ensures that any non-numeric values are converted to NaN (Not a Number), which can be handled or removed later.
* Removing Rows with Missing Values: After converting the columns to numeric data types, the line data=data.dropna() removes any rows that contain missing values, effectively dropping rows with NaN values.
* These data preprocessing steps help ensure data consistency and integrity for subsequent analysis and modeling tasks. By handling missing values appropriately, the code prepares the data for further exploration and modeling, allowing for more accurate and reliable insights and predictions.

**Data Visualization:-**

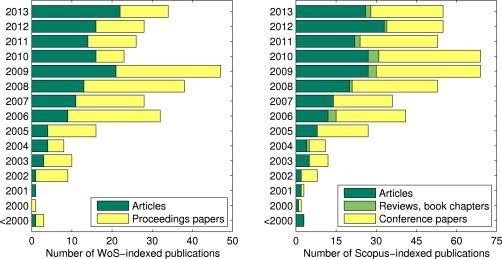
* Bar Plot: The code uses Seaborn’s barplot function to create a bar plot, visualizing the relationship between the “Month” column and the target variable “SMPEP2.” This bar plot provides a graphical representation of the average “SMPEP2” values for each month, allowing for a quick comparison of pricing trends across different months.
* Correlation Heatmap: The code utilizes Matplotlib and Seaborn to create a correlation heatmap using the sns.heatmap function. This heatmap visualizes the Pearson correlation coefficients between different features in the dataset. The annot=True parameter adds numerical annotations to the heatmap, displaying the correlation values for each pair of features.

**Data Splitting:-**

* Importing Necessary Libraries: The code imports the train\_test\_split function from the sklearn.model\_selection module to facilitate the data splitting process.
* Data Splitting: The line xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.2, random\_state=42) splits the data into training and testing sets. It takes the feature variables x and the target variable y as input and splits them into training and testing sets, with 80% of the data allocated for training and 20% for testing. The random\_state=42 parameter ensures reproducibility by fixing the random seed for the splitting process.

**Model Training:-**

* Importing Necessary Libraries: The code imports the Random Forest Regressor algorithm from the sklearn.ensemble module to facilitate the model training process.
* Model Initialization and Training: The line model = RandomForestRegressor() initializes an instance of the Random Forest Regressor model. Subsequently, the model.fit(xtrain, ytrain) line fits the model to the training data, where xtrain represents the feature variables and ytrain represents the corresponding target variable. During the training process, the Random Forest Regressor algorithm learns the patterns and relationships present in the training data to make accurate predictions.
* Model Evaluation: The code does not explicitly include a model evaluation step. However, it is common practice to evaluate the trained model’s performance using various metrics, such as mean squared error, mean absolute error, or R-squared, to assess how well the model fits the training data and its ability to generalize to new, unseen data points.



**PHASE 2:INNOVATION TO SOLVE THE PROBLEM**



**SmartGrid PredictPro :**

SmartGrid PredictPro is an innovative electricity price prediction platform designed to empower consumers, energy providers, and policymakers with accurate and realtime insights into electricity pricing. This intelligent system harnesses the power of advanced machine learning and data analytics to forecast electricity prices, enabling users to make informed decisions and optimize their energy consumption.

**Key Features:**

1. Real-time Predictions:

SmartGrid PredictPro provides upto-the-minute forecasts of electricity prices, allowing consumers to schedule energy-intensive tasks during offpeak hours and save on their energy bills.

1. User-Friendly Interface:

The platform boasts an intuitive and user-friendly interface accessible via web and mobile devices, making it easy for both residential and commercial users to access pricing information.

1. **Customized Alerts:**

Users can set personalized alerts to be notified when electricity prices are expected to peak or drop, enabling them to adjust their energy usage accordingly.

1. **Historical Data Analysis:**

SmartGrid PredictPro offers historical pricing data and trend analysis, empowering energy providers to optimize their pricing strategies and consumers to make long-term energy-saving plans.

1. **AI-Driven Accuracy:**

Our system employs state-of-the-art artificial intelligence algorithms to continuously improve prediction accuracy based on real-world data and market conditions.

1. **Energy Efficiency Recommendations:**

In addition to price predictions, the platform offers energy efficiency recommendations and tips to help users reduce their carbon footprint and save money.

1. **Integration with IoT Devices:**

SmartGrid PredictPro seamlessly integrates with IoTenabled smart devices, allowing users to automate energy management based on real-time price forecasts.

1. **Data Security:**

We prioritize data security and privacy, ensuring that user information and energy consumption data are protected at all times.

**Benefits:**

**Cost Savings:**

Consumers can lower their electricity bills by leveraging real-time pricing information and optimizing energy usage.

**Sustainability:**

SmartGrid PredictPro promotes energy efficiency and sustainability by encouraging responsible energy consumption.

-

**Data-Driven Decision-Making:**

Energy providers can refine pricing strategies, and policymakers can make informed decisions based on accurate pricing forecasts.

**Convenience:**

The user-friendly interface and customized alerts make it convenient for users to manage their energy consumption effectively.

SmartGrid PredictPro is the next-generation solution for electricity price prediction, bringing intelligence and efficiency to the energy sector while promoting a sustainable and eco-friendly future.

**PHASE 3: DEVELOPMENT PART 1**

**TOPIC: Start building the electricity price prediction model by loading and pre-processing the dataset**

The price of electricity depends on many factors. Predicting the price of electricity helps many businesses 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 of heavy machinery used by businesses. So if you want to learn how to predict the price of electricity, then this article is for you. In this article, I will walk you through the task of electricity price prediction with machine learning using PYTHON.

**Electricity Price Prediction (Case Study):**

Suppose that your business relies on computing services where the power consumed by your machines varies throughout the day. You do not know the actual cost of the electricity consumed by the machines throughout the day, but the organization has provided you with historical data of the price of the electricity consumed by the machines. Below is the information of the [**data**](https://raw.githubusercontent.com/amankharwal/Website-data/master/electricity.csv) we have for the task of forecasting electricity prices:

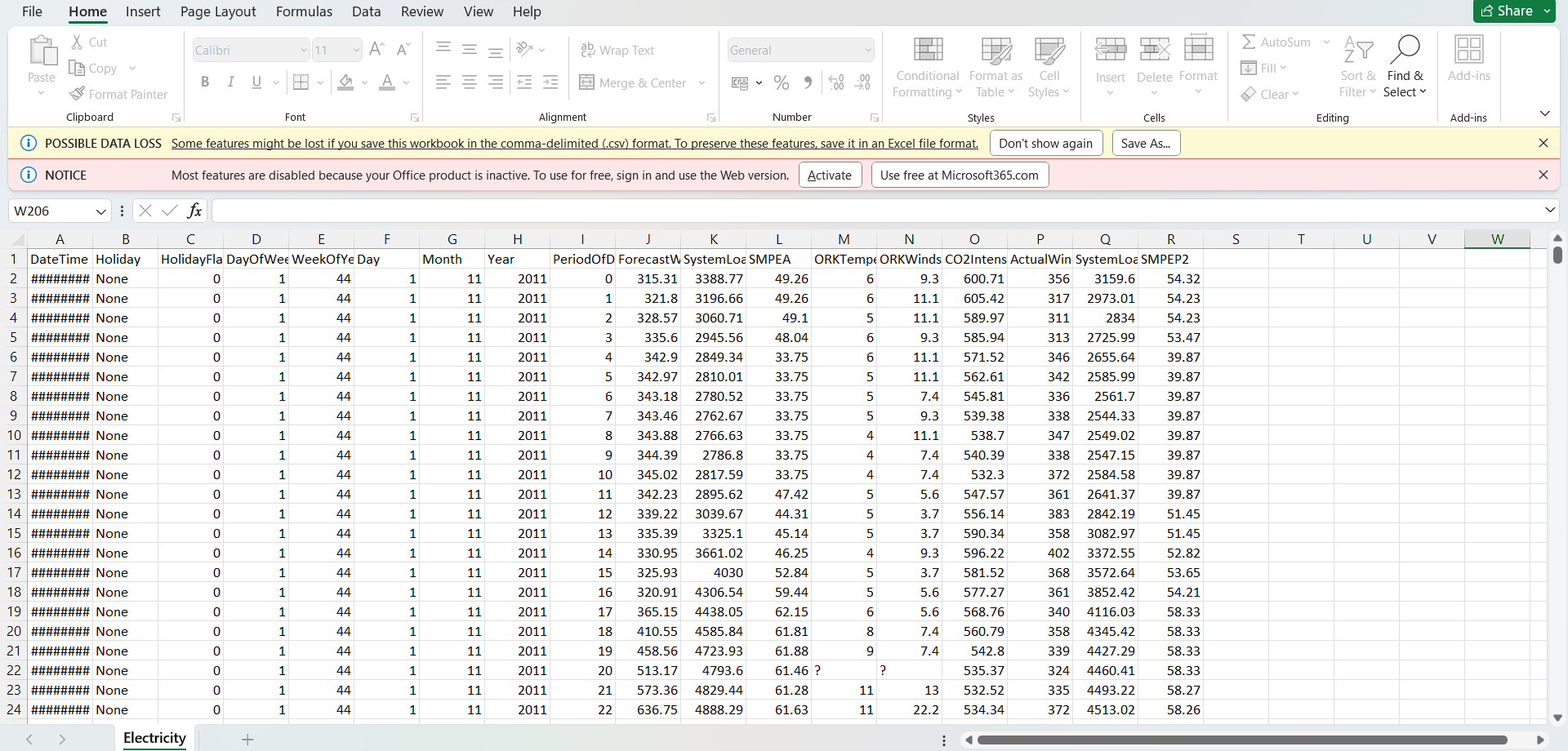
1. DateTime: Date and time of the record
2. Holiday: contains the name of the holiday if the day is a national holiday
3. HolidayFlag: contains 1 if it’s a bank holiday otherwise 0
4. DayOfWeek: contains values between 0-6 where 0 is Monday
5. WeekOfYear: week of the year
6. Day: Day of the date
7. Month: Month of the date
8. Year: Year of the date
9. PeriodOfDay: half-hour period of the day
10. ForcastWindProduction: forecasted wind production
11. SystemLoadEA forecasted national load
12. SMPEA: forecasted price
13. ORKTemperature: actual temperature measured
14. ORKWindspeed: actual windspeed measured
15. CO2Intensity: actual C02 intensity for the electricity produced
16. ActualWindProduction: actual wind energy production
17. SystemLoadEP2: actual national system load
18. SMPEP2: the actual price of the electricity consumed (labels or values to be predicted)

So your task here is to use this data to train a machine learning model to predict the price of electricity consumed by the machines. In the section below, I will take you through the task of electricity price prediction with machine learning using Python.

## **Electricity Price Prediction using Python:**

I will start the task of electricity price prediction by importing the necessary Python libraries and the dataset that we need for this task:

**GIVEN DATASET:**

****

**PROGRAM:**

import pandas as pd

import numpy as np

data = pd.read\_csv

print(data.head())

**DateTime Holiday ... SystemLoadEP2 SMPEP2**

**0 01/11/2011 00:00 None ... 3159.60 54.32**

**1 01/11/2011 00:30 None ... 2973.01 54.23**

**2 01/11/2011 01:00 None ... 2834.00 54.23**

**3 01/11/2011 01:30 None ... 2725.99 53.47**

**4 01/11/2011 02:00 None ... 2655.64 39.87**

**Data.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 38014 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**

I can see that so many features with numerical values are string values in the dataset and not integers or float values. So before moving further, we have to convert these string values to float values:

data["ForecastWindProduction"] = pd.to\_numeric(data["ForecastWindProduction"], errors= 'coerce')

data["SystemLoadEA"] = pd.to\_numeric(data["SystemLoadEA"], errors= 'coerce')

data["SMPEA"] = pd.to\_numeric(data["SMPEA"], errors= 'coerce')

data["ORKTemperature"] = pd.to\_numeric(data["ORKTemperature"], errors= 'coerce')

data["ORKWindspeed"] = pd.to\_numeric(data["ORKWindspeed"], errors= 'coerce')

data["CO2Intensity"] = pd.to\_numeric(data["CO2Intensity"], errors= 'coerce')

data["ActualWindProduction"] = pd.to\_numeric(data["ActualWindProduction"], errors= 'coerce')

data["SystemLoadEP2"] = pd.to\_numeric(data["SystemLoadEP2"], errors= 'coerce')

data["SMPEP2"] = pd.to\_numeric(data["SMPEP2"], errors= 'coerce')

Now let’s have a look at whether this dataset contains any null values or not:

1

data.isnull().sum()

**DateTime 0**

**Holiday 0**

**HolidayFlag 0**

**DayOfWeek 0**

**WeekOfYear 0**

**Day 0**

**Month 0**

**Year 0**

**PeriodOfDay 0**

**ForecastWindProduction 5**

**SystemLoadEA 2**

**SMPEA 2**

**ORKTemperature 295**

**ORKWindspeed 299**

**CO2Intensity 7**

**ActualWindProduction 5**

**SystemLoadEP2 2**

**SMPEP2 2**

**dtype: int64**

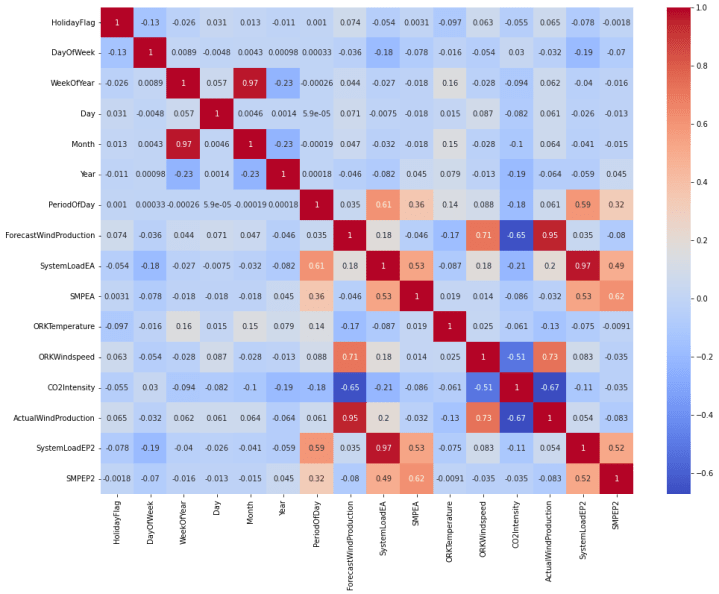
So there are some columns with null values, I will drop all these rows containing null values from the dataset:

1

data = data.dropna()

Now let’s have a look at the correlation between all the columns in the dataset:

|  |
| --- |
| import seaborn as sns |
|  | import matplotlib.pyplot as plt |
|  | correlations = data.corr(method='pearson') |
|  | plt.figure(figsize=(16, 12)) |
|  | sns.heatmap(correlations, cmap="coolwarm", annot=True) |
|  | plt.show() |



## **Electricity Price Prediction Model:**

Now let’s move to the task of training an electricity price prediction model. Here I will first add all the important features to x and the target column to y, and then I will split the data into training and test sets:

|  |
| --- |
| x = data[["Day", "Month", "ForecastWindProduction", "SystemLoadEA", |
|  | "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity", |
|  | "ActualWindProduction", "SystemLoadEP2"]] |
|  | y = data["SMPEP2"] |
|  | from sklearn.model\_selection import train\_test\_split |
|  | xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, |
|  | test\_size=0.2, |
|  | random\_state=42) |

As this is the problem of regression, so here I will choose the Random Forest regression algorithm to train the electricity price prediction model:

1

from sklearn.ensemble import RandomForestRegressor

2

model = RandomForestRegressor()

3

model.fit(xtrain, ytrain)

**RandomForestRegressor(bootstrap=True, ccp\_alpha=0.0, criterion='mse',**

**max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,**

**max\_samples=None, min\_impurity\_decrease=0.0,**

**min\_impurity\_split=None, min\_samples\_leaf=1,**

**min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,**

**n\_estimators=100, n\_jobs=None, oob\_score=False,**

**random\_state=None, verbose=0, warm\_start=False)**

Now let’s input all the values of the necessary features that we used to train the model and have a look at the price of the electricity predicted by the model:

#features = [["Day", "Month", "ForecastWindProduction", "SystemLoadEA", "SMPEA", "ORKTemperature", "ORKWindspeed", "CO2Intensity", "ActualWindProduction", "SystemLoadEP2"]]

features = np.array([[10, 12, 54.10, 4241.05, 49.56, 9.0, 14.8, 491.32, 54.0, 4426.84]])

model.predict(features)

Predicting the price of electricity helps a lot of companies to understand how much electricity expenses they have to pay every year. I hope you liked this article on the task of electricity price prediction with machine learning using Python. Feel free to ask your valuable questions in the comments section below.

# This Python 3 environment comes with many helpful analytics libraries installed

# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python

# For example, here's several helpful packages to load

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"

# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session

**PHASE 4: DEVELOPMENT PART 2**

**TOPIC:** Continue building the electricity prices prediction model by performing feature engineering, model training and evaluation.



In the age of digital transformation, electricity is the only power source used by all applications. Therefore, a suitable mechanism is needed to assess the amount of power used for both home and industrial uses in our daily lives. Electricity price predictions give you information about how much you will have to pay as well as how much capacity you are using, how much is needed, and other relevant details.

This study work has designed a model to anticipate electricity by utilizing machine learning techniques, as a good analysis and prediction of electricity is necessary. Predicting the price of electricity is difficult since it depends on a variety of variables, including national wind, wind production, and natural causes. Therefore, a suitable mechanism is needed to assess the amount of power used for both home and industrial uses in our daily lives.

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.metrics import mean\_squared\_error

from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.linear\_model import LinearRegression

from sklearn.neighbors import KNeighborsRegressor

**Primary Price Determinants**

The interplay of supply and demand determines electricity prices. On the other hand, demand is somewhat price-dependent, while supply prices vary greatly depending on a number of factors. The short- to mid-term fundamental price determinants are outlined here in summary.

**• Demand**

In order to activate the energy sources that provide power, demand is crucial.

**• Vent**

Wind speed has a significant impact on the price of electricity because it generates a significant amount of the nation's electricity.

**• Condensation**

The quantity of snow and rain that hydro reservoirs retain can have a significant impact on the cost of energy.

df=pd.read\_csv("/kaggle/input/electrity-prices/electricity\_prices.csv", low\_memory=False)

df.head()

**• Temperature**

Both the supply and demand for power are directly impacted by temperature, as is the other way around. Demand is impacted by the usage of electricity for heating. Given that temperature and wind are typically connected, temperature can have an impact on energy output. Temperature has an impact on hydropower as well; for instance, melting snow or ice can enhance output.

**• Prices of commodities**

The price of commodities has a significant impact on power pricing, as the overall energy consumption is heavily reliant on these commodities. As energy moves between zones, the prices of commodities also have an impact on other contraries, since energy systems become less reliant on these commodities. <class

data=df[['ForecastWindProduction',

       'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',

       'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]

**• Transmission capacity**

The division of various price ranges determined by the capacity of transmission between regions and national boundaries. The price in a particular area is influenced by the transmission capacity, which restricts the flow of electricity. The price in the two bidding areas would be the same if there were unlimited transmission capabilities.

**Model Training**

**Context: -**

Dataset containing the price of electricity for a data center in addition to factors that might affect the price.

**Dataset:-** <https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction>

**Column Descriptions:**

1. DateTime: String, defines date and time of sample
2. Holiday: String, gives name of holiday if day is a bank holiday
3. HolidayFlag: integer, 1 if day is a bank holiday, zero otherwise
4. DayOfWeek: integer (0–6), 0 Monday, day of week
5. WeekOfYear: integer, running week within year of this date
6. Day integer: day of the date
7. Month integer: month of the date
8. Year integer: year of the date
9. PeriodOfDay integer: denotes half hour period of day (0–47)
10. SystemLoadEA: the national load forecast for this period
11. SMPEA: the price forecast for this period
12. ORKTemperature: the actual temperature measured at Cork airport
13. ORKWindspeed: the actual windspeed measured at Cork airport
14. CO2Intensity: the actual CO2 intensity in (g/kWh) for the electricity produced
15. ActualWindProduction: the actual wind energy production for this period
16. SystemLoadEP2: the actual national system load for this period
17. SMPEP2: the actual price of this time period, the value to be forecasted

**Program Of Training :-**

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

import tensorflow as tf

import xgboost as xgb

import os

import warnings

from tensorflow.keras.layers import Dense, LSTM, Conv1D, MaxPooling1D, TimeDistributed, Flatten, Dropout, RepeatVector

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

from statsmodels.tsa.stattools import adfuller, kpss, ccf

from sklearn.metrics import mean\_squared\_error, r2\_score

from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler

from sklearn.decomposition import PCA

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

from math import sqrt

%matplotlib inline

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

**Reading the dataset:-**

Df = pd.read\_csv(“/content/electricity\_prices.csv”, na\_values=[‘?’])

Df.head()

Df.shape

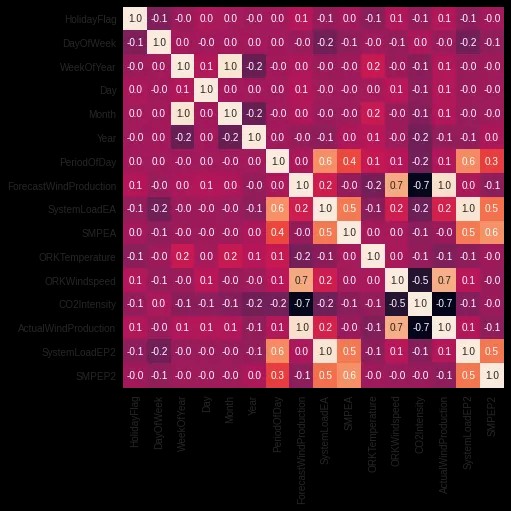
**Plotting the target feature**:-

Plt.plot(“SMPEP2”, data=df)

**Correlation plot of Independent attributes:-**

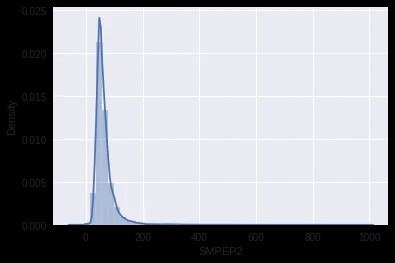
Plt.figure(figsize=(9,7))

Sns.heatmap(df.corr(), annot=True, square=True, fmt=’.1f’, cbar=False);



**Distribution plot of Target feature**:-

Sns.distplot(df[‘SMPEP2’])



**Splitting the independent features and target feature:-**

X = df[[‘ActualWindProduction’, ‘SystemLoadEP2’, ‘SMPEA’, ‘SystemLoadEA’, ‘ForecastWindProduction’,

‘DayOfWeek’, ‘Year’, ‘ORKWindspeed’, ‘CO2Intensity’, ‘PeriodOfDay’]]

Y = df[‘SMPEP2’]

**Compiling the model:-**

Model.compile(loss=’mse’, optimizer=’adam’, metrics=[‘mse’,’mae’])

## **Evaluating the model on test set:-**

**from** **sklearn.metrics** **import** mean\_absolute\_error,r2\_score

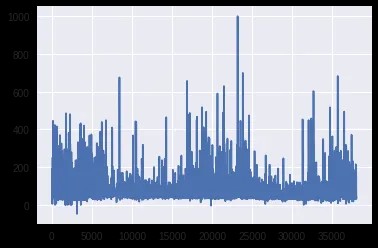
predictions = model.predict(X\_test)

print (f"MAE: **{**mean\_absolute\_error(y\_test, predictions)**}**")

print (f"R2\_score: **{**r2\_score(y\_test, predictions)**}**")

**Evaluation: -**

By using this analysis, we can predict electricity prices that is the actual price of this time period and forecast future business strategies.



**CONCLUSION:**

In conclusion, applied data science-based electricity price prediction is a useful instruction for resolving the issues with the energy market. Enhancing the precision of pricing projections through the application of sophisticated data analytics and machine learning methodologies can provide more informed choices for suppliers and customers alike. This strategy can help maximize energy use , cut expenses , and improve the energy sector’s sustainability and efficiency

The user-friendly interface and customized alerts make it convenient for users to manage their energy consumption effectively.

SmartGrid PredictPro is the next-generation solution for electricity price prediction, bringing intelligence and efficiency to the energy sector while promoting a sustainable and eco-friendly future.