ELECTRICITY PRICE PREDICTION

Phase-3 Document submission

Project: Electricity Price Prediction

Phase 3: Development part 1

Topic: In this part you will begin building your project by loading and preprocessing the dataset.

Begin building the electricity prices prediction model by loading and preprocessing the dataset.

Load and historical electricity Prices dataset and preprocess the data for analysis.

INTRODUCTION:

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** we have for the task of forecasting electricity prices:

1. Date Time: Date and time of the record
2. Holiday: contains the name of the holiday if the day is a national holiday
3. Holiday Flag: contains 1 if it's a bank holiday otherwise 0
4. Day Of Week: contains values between 0-6 where 0 is Monday
5. Week Of Year: week of the year
6. Day: Day of the date
7. Month: Month of the date
8. Year: Year of the date
9. Period Of Day: half-hour period of the day
10. SMPEA: forecasted price
11. ORK Temperature: actual temperature measured
12. ORK Windspeed: actual windspeed measured
13. CO2Intensity: actual C02 intensity for the electricity produced

14. Actual Wind Production: actual wind energy production

15. SystemLoadEP2: actual national system load

16. 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 data science using Python.

Importance of Electricity Price Prediction:

1. **Informed Decision Making:** Electricity price prediction enables consumers and producers to make calculated decisions about energy consumption and production.

2. **Risk Mitigation:** Accurate price prediction helps businesses and investors mitigate the financial risks associated with fluctuating electricity prices.

3. **Market Efficiency:** Predicting prices allows for efficient allocation of resources and ensures fair competition in the energy market.

Key Factors Influencing Electricity Prices:

Supply and Demand Dynamics: The balance between electricity supply and demand heavily influences market prices, with high demand periods often leading to price spikes.

Weather Conditions: Weather plays a significant role in changing electricity consumption patterns, leading to fluctuations in prices.

Energy Policies and Regulations: Policies and regulations impacting the energy sector can have direct effects on electricity prices, such as carbon pricing or subsidies.

Energy Policies and Regulations: Policies and regulations impacting the energy sector can have direct effects on electricity prices, such as carbon pricing or subsidies.

Generation Capacity and Availability: The availability of different energy sources and generation capacities directly affects electricity prices.

Data Collection and Preprocessing:

Sources of Data for Electricity Price Prediction: Data from energy markets, weather stations, and power plant operations are key sources for predicting electricity prices.

Handling Missing Data and Outliers: Implementing techniques like interpolation and outlier detection ensures the accuracy and reliability of prediction models.

Feature Engineering and Data Transformation: Creating meaningful input variables and transforming data through normalization or dimensionality reduction optimize model performance

Forecasting Electricity Prices:

Implementing the Selected Model for Price Forecasting:

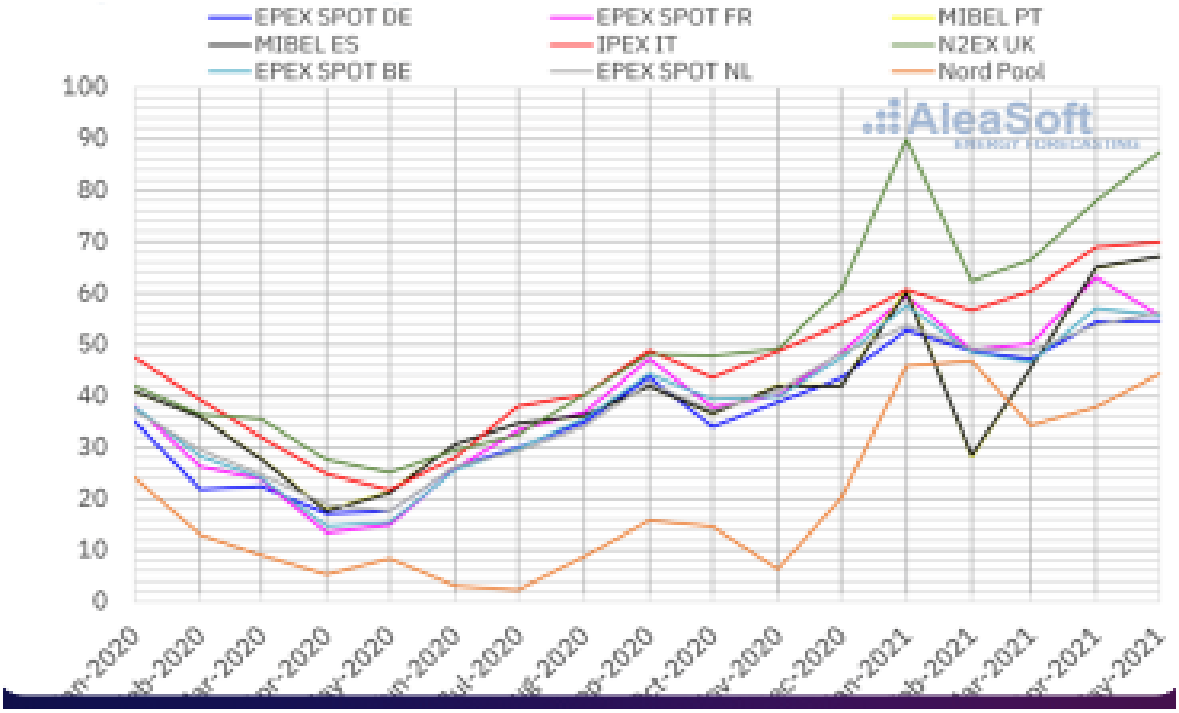
The chosen model is used with real-time or historical data to predict future electricity prices.

Assessing Model Accuracy and Reliability: The accuracy of the predictions is compared against actual prices to determine the reliability of the model.

Analyzing and Interpreting Predicted Prices: Predicted prices are studied, and insights are extracted to guide decision-making processes in the energy market.

Benefits and Applications:

Helping Consumers and Producers:



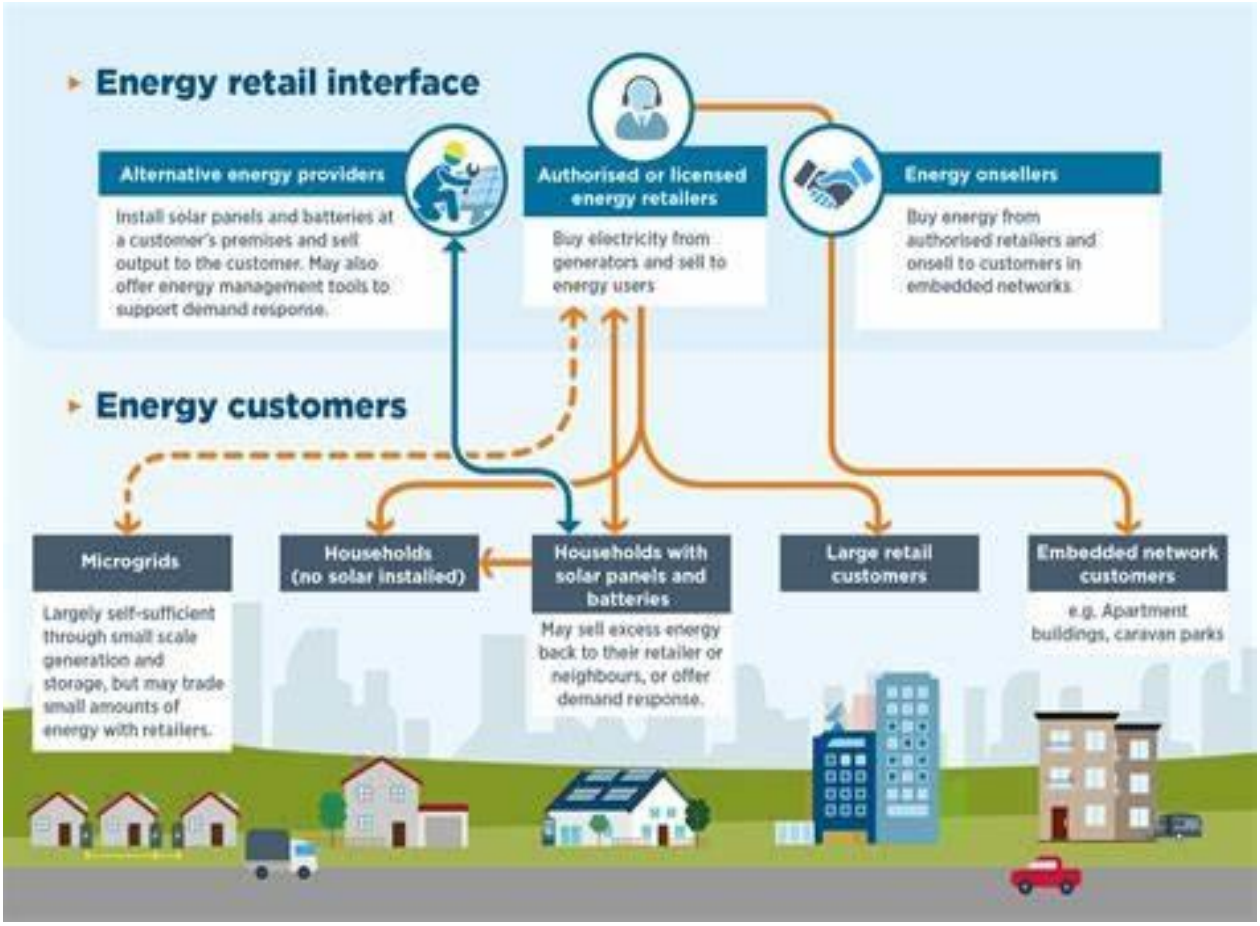
Price prediction enables informed choices, helping consumers save costs and producers optimize profit.

Optimizing Consumption and Production:

Predictive models aid in optimizing the use of energy resources and reducing waste.

Supporting Policy Making and Market Forecasting:

Predicting prices aids policymakers in creating effective energy policies and supports accurate market forecasting.



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:

IN[]:

```
import pandas as pd

import numpy as np
```

```
data=pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/electricity.csv")

print(data.head())
```

out[]:

```
         Date Time 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

[5 rows x 18 columns]
```

Let's have a look at all the columns of this dataset:

IN[]:

```
1. data.info()
2.
```

OUT[]:

```
<class 'pandas.core.frame.DataFrame'>
Range Index: 38014 entries, 0 to 38013
Data columns (total 18 columns):
 #   Column          Non-Null Count  dtypes
---  ------          --------------  -----
 0   Date Time       38014 non-null  object
 1   Holiday         38014 non-null  object
 2   Holiday Flag    38014 non-null  int64
 3   Day Of Week     38014 non-null  int64
 4   Week Of Year    38014 non-null  int64
 5   Day             38014 non-null  int64
 6   Month           38014 non-null  int64
```

7	Year	38014 non-null	int64
8	Period Of Day	38014 non-null	int64
9	Forecast Wind Production	38014 non-null	object
10	System Load EA	38014 non-null	object
11	SMPEA	38014 non-null	object
12	ORK Temperature	38014 non-null	object
13	ORK Windspeed	38014 non-null	object
14	CO2Intensity	38014 non-null	object
15	Actual Wind Production	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

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["Forecast Wind Production"] = pd.to
numeric(data["Forecast Wind Production"],
errors= 'coerce')

data["System Load EA"] = pd.to
numeric(data["System Load EA"], errors=
'coerce')

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

data["ORK Temperature"] = pd.to
numeric(data["ORK Temperature"], errors=
'coerce')
```

```
data["ORK Windspeed"] = pd.to numeric(data["ORK
Windspeed"], errors= 'coerce')

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

data["Actual Wind Production"] = pd.to
numeric(data["Actual Wind Production"], 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. **DateTime** **0**
2. **Holiday** **0**
3. **HolidayFlag** **0**
4. **DayOfWeek** **0**
5. **Week Of Year** **0**
6. **Day** **0**
7. **Month** **0**
8. **Year** **0**
9. **Period Of Day** **0**
10. **Forecast Wind Production** **5**

11. **System Load EA 2**
12. **SMPEA 2**
13. **ORK Temperature 295**
14. **ORKWindspeed 299**
15. **CO2Intensity 7**
16. **ActualWindProduction 5**
17. **SystemLoadEP2 2**
18. **SMPEP2 2**
19. **dtype: int64**

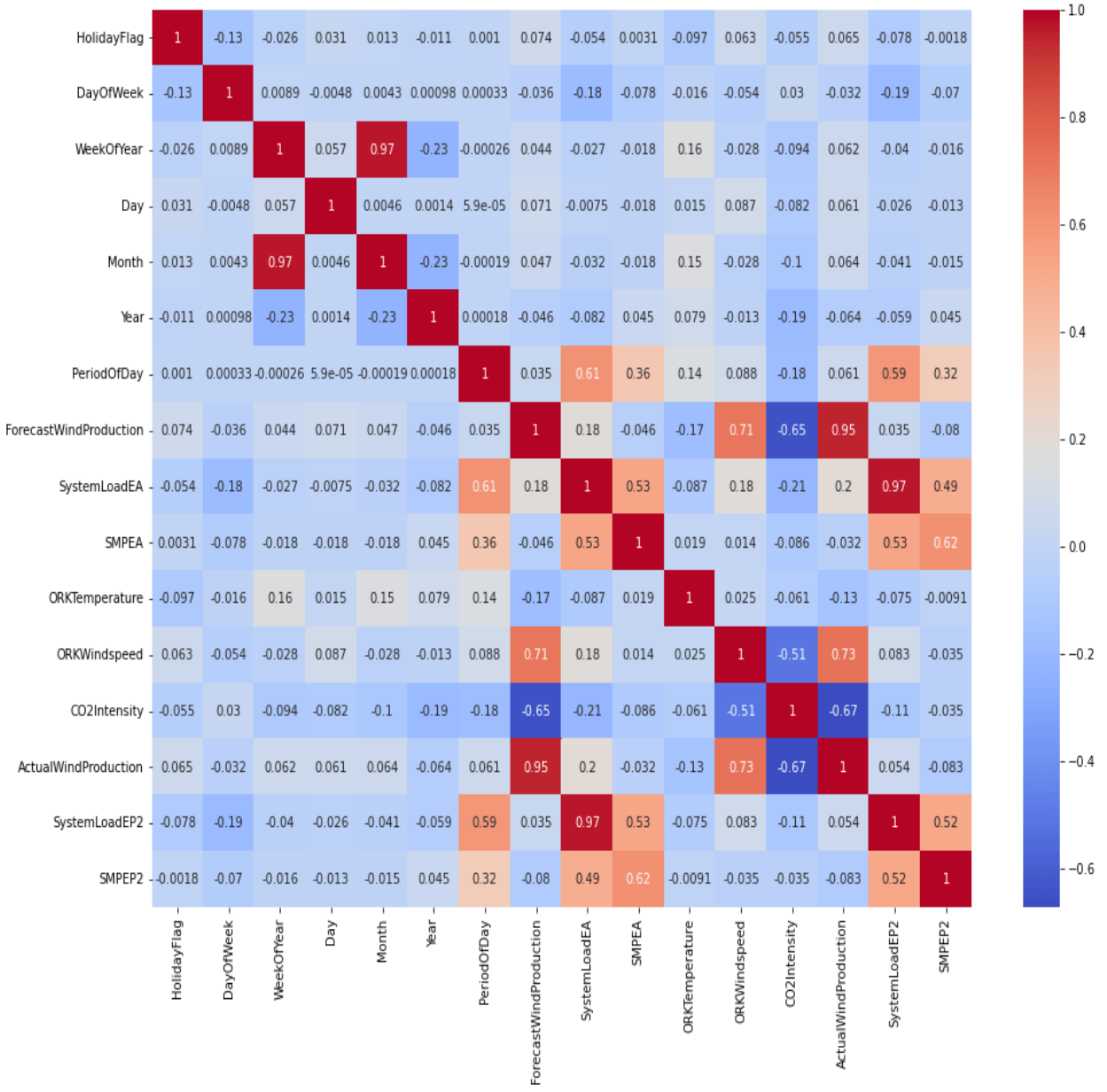
So there are some columns with null values, I will drop all these rows containing null values from the dataset:

IN[]:

```
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, test_size=0.2)
```

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

```
from sklearn.ensemble import RandomForestRegressor
model = RandomForestRegressor()
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)
```

OUT[]:

array([65.1696])

Conclusion:

1. Recap of Key Points:

   Electricity price prediction is essential for informed decision making, risk mitigation, and market efficiency.

2. Potential Future Developments:

Advancements in machine learning and big data analytics hold promise for even more accurate electricity price predictions