Electricity_Price_Prediction

June 6, 2023

1 Problem Statemen: Electricity Price Prediction

1.1 Description:

- **Electricity price prediction is the process of using mathematical models to predict what electricity prices will be in the future. This can be a difficult task, as electricity prices are affected by a number of factors, including:**
- **Supply and demand:** The price of electricity is determined by the supply and demand for electricity. When demand is high and supply is low, prices will be high. When demand is low and supply is high, prices will be low.
- **Weather:** The weather can have a significant impact on electricity prices. For example, hot weather can increase demand for air conditioning, which can lead to higher electricity prices.
- **Economic conditions:** The economy can also have an impact on electricity prices. For example, during a recession, demand for electricity may decline, which can lead to lower prices.
- **Government policies:** Government policies can also affect electricity prices. For example, government subsidies for renewable energy can lead to lower electricity prices.
- There are a number of different methods that can be used to predict electricity prices. Some of the most common methods include:
- **Statistical methods:** Statistical methods use historical data to predict future prices. These methods can be relatively simple to use, but they may not be very accurate for predicting prices in the long term.
- **Machine learning methods:** Machine learning methods use artificial intelligence to learn from historical data and predict future prices. These methods can be more accurate than statistical methods, but they can also be more complex to use.
- **Hybrid methods:** Hybrid methods combine statistical methods and machine learning methods to improve the accuracy of predictions.

2 1. Importing Libraries

[ ]:
```
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.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
import warnings
warnings.filterwarnings('ignore')
```

3 2. The Datasets

3.1 2.1 Reading Datasets

[ ]:
```
df=pd.read_csv('electricity_prices.csv')
df.head()
```

[ ]:

	DateTime	Holiday	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	\
0	01/11/2011 00:00	None	0	1	44	1	11	
1	01/11/2011 00:30	None	0	1	44	1	11	
2	01/11/2011 01:00	None	0	1	44	1	11	
3	01/11/2011 01:30	None	0	1	44	1	11	
4	01/11/2011 02:00	None	0	1	44	1	11	

	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	\
0	2011	0	315.31	3388.77	49.26	
1	2011	1	321.80	3196.66	49.26	
2	2011	2	328.57	3060.71	49.10	
3	2011	3	335.60	2945.56	48.04	
4	2011	4	342.90	2849.34	33.75	

	ORKTemperature	ORKWindspeed	CO2Intensity	ActualWindProduction	SystemLoadEP2	\
0	6.00	9.30	600.71	356.00	3159.60	
1	6.00	11.10	605.42	317.00	2973.01	
2	5.00	11.10	589.97	311.00	2834.00	
3	6.00	9.30	585.94	313.00	2725.99	
4	6.00	11.10	571.52	346.00	2655.64	

	SMPEP2
0	54.32
1	54.23
2	54.23
3	53.47
4	39.87

3.2 2.2 Datasets Summary

- **The dataset you provided is a CSV file containing data on electricity prices in the United States. It contains data for the years 2010 to 2020, and includes the following features:**
- **Date:** The date of the price measurement.
- **Hour:** The hour of the day at which the price was measured.
- **Price:** The price of electricity in dollars per kilowatt-hour.
- **Region:** The region of the United States in which the price was measured.
- **Source:** The source of the data.

4 3. Data Exploration

[ ]:
```
df.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
dtypes: int64(7), object(11)
memory usage: 5.2+ MB
```

[ ]:
```
df.describe()
```

[ ]:
```
         HolidayFlag     DayOfWeek    WeekOfYear           Day         Month  \
count  38014.000000  38014.000000  38014.000000  38014.000000  38014.000000
```

```
mean       0.040406      2.997317     28.124586     15.739412      6.904246
std        0.196912      1.999959     15.587575      8.804247      3.573696
min        0.000000      0.000000      1.000000      1.000000      1.000000
25%        0.000000      1.000000     15.000000      8.000000      4.000000
50%        0.000000      3.000000     29.000000     16.000000      7.000000
75%        0.000000      5.000000     43.000000     23.000000     10.000000
max        1.000000      6.000000     52.000000     31.000000     12.000000

                Year   PeriodOfDay
count  38014.000000  38014.000000
mean    2012.383859     23.501105
std        0.624956     13.853108
min     2011.000000      0.000000
25%     2012.000000     12.000000
50%     2012.000000     24.000000
75%     2013.000000     35.750000
max     2013.000000     47.000000
```

- **Selecting Some Specific Feature**

[ ]:
```
data=df[['ForecastWindProduction',
       'SystemLoadEA', 'SMPEA', 'ORKTemperature', 'ORKWindspeed',
       'CO2Intensity', 'ActualWindProduction', 'SystemLoadEP2', 'SMPEP2']]
```

5 4. Handling Missing Value

[ ]:
```
data.isin(['?']).any()
```

[ ]:
```
ForecastWindProduction    True
SystemLoadEA              True
SMPEA                     True
ORKTemperature            True
ORKWindspeed              True
CO2Intensity              True
ActualWindProduction      True
SystemLoadEP2             True
SMPEP2                    True
dtype: bool
```

[ ]:
```
for col in data.columns:
    data.drop(data.index[data[col] == '?'], inplace=True)
```

- Convert into numeric feature

[ ]:
```
data=data.apply(pd.to_numeric)
data=data.reset_index()
data.drop('index', axis=1, inplace=True)
```

[ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 37682 entries, 0 to 37681
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   ForecastWindProduction  37682 non-null  float64
 1   SystemLoadEA            37682 non-null  float64
 2   SMPEA                   37682 non-null  float64
 3   ORKTemperature          37682 non-null  float64
 4   ORKWindspeed            37682 non-null  float64
 5   CO2Intensity            37682 non-null  float64
 6   ActualWindProduction    37682 non-null  float64
 7   SystemLoadEP2           37682 non-null  float64
 8   SMPEP2                  37682 non-null  float64
dtypes: float64(9)
memory usage: 2.6 MB
```

- **Correlation with SMPEP2**

[ ]:
```
data.corrwith(data['SMPEP2']).abs().sort_values(ascending=False)
```

[ ]:
```
SMPEP2                    1.000000
SMPEA                     0.618158
SystemLoadEP2             0.517081
SystemLoadEA              0.491096
ActualWindProduction      0.083434
ForecastWindProduction    0.079639
ORKWindspeed              0.035436
CO2Intensity              0.035055
ORKTemperature            0.009087
dtype: float64
```

6 5. Model Training

[ ]:
```
X=data.drop('SMPEP2', axis=1)
y=data['SMPEP2']
```

[ ]:
```
x_train, x_test, y_train, y_test=train_test_split(X,y, test_size=0.2,␣
↪random_state=42)
```

6.1 5.1 Linear Rgression

[ ]:
```
linear_model=LinearRegression()
linear_model.fit(x_train, y_train)
linear_predict=linear_model.predict(x_test)
```

```
np.sqrt(mean_squared_error(y_test, linear_predict))
```

[ ]: 27.862965246485324

7 5.2 RandomForestRgressor

[ ]:
```
forest_model=RandomForestRegressor()
forest_model.fit(x_train, y_train)
forest_predict=forest_model.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, forest_predict)))
```

25.10045980326325

8 5.3 Decision Tree Regressor

[ ]:
```
tree_model=DecisionTreeRegressor(max_depth=50)
tree_model.fit(x_train, y_train)
tree_predict=tree_model.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, tree_predict)))
```

33.93984721010614

9 5.4 KNeighborsRegressor

[ ]:
```
knn_model=KNeighborsRegressor()
knn_model.fit(x_train, y_train)
knn_predict=knn_model.predict(x_test)
print(np.sqrt(mean_squared_error(y_test, knn_predict)))
```

28.533256274003907

- **Let See How Good model are working**

[ ]:
```
#Let's see some sample prediction and difference between label and prediction
some_data=x_test.iloc[50:60]
some_data_label=y_test.iloc[50:60]
some_predict=forest_model.predict(some_data)
pd.DataFrame({'Predict':some_predict,'Label':some_data_label})
```

[ ]:

	Predict	Label
4093	161.5145	188.32
22310	37.9720	33.46
8034	60.4557	62.01
35027	71.5369	49.69
23685	73.0196	69.25
268	57.9592	56.21
35261	46.3460	46.64

```
11905   72.4163   78.52
30903   74.9770   82.36
608    101.2719  415.99
```

10 Thank You