ELECTRICITY PRICES PREDICTION

TEAM MEMBER

DEVIKA NAVAKUMAR

Phase 4 Submission Document

Project : ELECTRICITY PRICES PREDICTION

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

Introduction:

In a world that thrives on energy as the lifeblood of modern society, the dynamics of electricity pricing have a profound impact on both consumers and producers. The ability to accurately predict electricity prices holds immense significance for various stakeholders, ranging from individual households seeking to manage their energy costs to utility companies striving to optimize *resource allocation and policy makers working towards a sustainable energy future.*

Electricity price prediction is not merely a matter of financial prudence; it's a critical element in the broader landscape of energy management and sustainability. It empowers us to make informed decisions, reduce energy waste, and align our consumption patterns with fluctuating supply and demand dynamics.

This discussion or project aims to delve into the intricate world of electricity price prediction. We will explore the multifaceted factors that influence pricing, from supply and demand patterns to environmental conditions and regulatory policies. Through the lens of data-driven approaches, machine learning, and statistical modeling, we will uncover the methodologies and tools used to forecast electricity prices with increasing precision.

Throughout our journey, we will address the real-world implications of electricity price prediction. From enabling cost-efficient strategies for businesses to encouraging renewable energy adoption and grid optimization, the ability to foresee price trends stands as a linchpin in the pursuit of an efficient, sustainable, and equitable energy ecosystem.

As we embark on this exploration of electricity price prediction, we invite you to discover the intricate interplay between data, technology, and the future of energy management. Join us as we uncover the valuable insights hidden within the numbers and explore the potential to make more informed, economically sound, and environmentally responsible decisions in an electrified world.

Given Dataset

1	DateTime	Holiday	HolidayFla	DayOfWe	WeekOfY	Day	Month	Year	PeriodOfD	ForecastW	SystemLo	SMPEA	ORKTemp	ORKWind	CO2Intens	ActualWir	SystemLo	SMPEP2
2	########	None	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6	9.3	600.71	356	3159.6	54.32
3	########	None	0	1	44	1	11	2011	1	321.8	3196.66	49.26	6	11.1	605.42	317	2973.01	54.23
4	########	None	0	1	44	1	11	2011	2	328.57	3060.71	49.1	5	11.1	589.97	311	2834	54.23
5	########	None	0	1	44	1	11	2011	3	335.6	2945.56	48.04	6	9.3	585.94	313	2725.99	53.47
6	########	None	0	1	44	1	11	2011	4	342.9	2849.34	33.75	6	11.1	571.52	346	2655.64	39.87
7	########	None	0	1	44	1	11	2011	5	342.97	2810.01	33.75	5	11.1	562.61	342	2585.99	39.87
8	########	None	0	1	44	1	11	2011	6	343.18	2780.52	33.75	5	7.4	545.81	336	2561.7	39.87
9	########	None	0	1	44	1	11	2011	7	343.46	2762.67	33.75	5	9.3	539.38	338	2544.33	39.87
10	########	None	0	1	44	1	11	2011	8	343.88	2766.63	33.75	4	11.1	538.7	347	2549.02	39.87
11	########	None	0	1	44	1	11	2011	9	344.39	2786.8	33.75	4	7.4	540.39	338	2547.15	39.87
12	########	None	0	1	44	1	11	2011	10	345.02	2817.59	33.75	4	7.4	532.3	372	2584.58	39.87
13	########	None	0	1	44	1	11	2011	11	342.23	2895.62	47.42	5	5.6	547.57	361	2641.37	39.87
14	########	None	0	1	44	1	11	2011	12	339.22	3039.67	44.31	5	3.7	556.14	383	2842.19	51.45
15	########	None	0	1	44	1	11	2011	13	335.39	3325.1	45.14	5	3.7	590.34	358	3082.97	51.45
16	########	None	0	1	44	1	11	2011	14	330.95	3661.02	46.25	4	9.3	596.22	402	3372.55	52.82
17	########	None	0	1	44	1	11	2011	15	325.93	4030	52.84	5	3.7	581.52	368	3572.64	53.65
18	########	None	0	1	44	1	11	2011	16	320.91	4306.54	59.44	5	5.6	577.27	361	3852.42	54.21
19	########	None	0	1	44	1	11	2011	17	365.15	4438.05	62.15	6	5.6	568.76	340	4116.03	58.33
20	########	None	0	1	44	1	11	2011	18	410.55	4585.84	61.81	8	7.4	560.79	358	4345.42	58.33
21	########	None	0	1	44	1	11	2011	19	458.56	4723.93	61.88	9	7.4	542.8	339	4427.29	58.33
22	########	None	0	1	44	1	11	2011	20	513.17	4793.6	61.46	?	?	535.37	324	4460.41	58.33
23	########	None	0	1	44	1	11	2011	21	573.36	4829.44	61.28	11	13	532.52	335	4493.22	58.27

PROCEDURE:

Feature Engineering

Feature engineering is a critical step in building accurate electricity price prediction models. Effective feature engineering can help capture relevant patterns and relationships in the data. Here are some feature engineering techniques and considerations for electricity price prediction:

1. Time-Based Features:

- *Time of Day: Create features to represent the time of day, such as hour of the day or minute of the hour. Electricity prices often exhibit daily and hourly patterns.*
- *Day of the Week: Include features for the day of the week to capture weekly seasonality.*
- *Month and Season: Incorporate features for the month and season to capture monthly and seasonal patterns.*
- *Holidays: Add binary features to indicate holidays or special events that may affect electricity prices.*

2. Lagged Features:

- *Lagged Prices: Include lagged electricity prices as features. Lagged values can capture autocorrelation and previous price trends.*
- *-Lagged Demand: Consider lagged electricity demand as a feature, as demand patterns can influence prices.*

3. Rolling Statistics:

- *Rolling Mean and Rolling Standard Deviation: Calculate rolling statistics over a certain window (e.g., 7 days) to capture short-term trends and volatility.*

4. Weather Data:

- *Incorporate weather data, such as temperature, humidity, or wind speed, as these factors can impact electricity consumption and prices.*

5. Demand Data:

- *Include features related to electricity demand, such as historical demand levels and peak demand periods.*

6. Market Data:

- *Consider variables related to the energy market, such as fuel prices, electricity generation capacity, or the state of the grid.*

7. Feature Scaling:

- *Normalize or scale features as needed to ensure that they have the same magnitude. This is important for models like linear regression or neural networks.*

8. Categorical Variables:

- *If you have categorical variables (e.g., region or market type), use one-hot encoding or other categorical encoding techniques to convert them into numerical features.*

9. Special Events:

- *Include features that indicate special events or anomalies, such as power outages or significant market changes.*

10. Price Differencing:

- *Calculate differences between consecutive price values to create features that capture price changes.*

11. Calendar Events:

- *Incorporate calendar-related features, such as the number of days until the next holiday or the number of days remaining in the billing cycle.*

12. Feature Selection:

- *Use feature selection techniques to identify the most relevant features for your model. Eliminate redundant or unimportant features to reduce model complexity.*

13. Domain-Specific Features:

- *Consult with domain experts in the energy industry to identify domain-specific features that might influence electricity prices.*

14. Time Series Decomposition:

- *Decompose the time series data into trend, seasonality, and residual components using methods like seasonal decomposition of time series (STL) and use these components as features.*

15. External Data Sources:

- *Consider incorporating external data sources, such as economic indicators, news sentiment, or energy market reports, to enhance the model's predictive power.*

Model Training

Training a model for electricity price prediction involves several key steps. Here's a high-level overview of the process:

1. Data Collection: *Gather historical data on electricity prices. This data may include information such as time of day, season, weather conditions, demand, and more. High-quality and comprehensive data are crucial for accurate predictions.*

2. Data Preprocessing: *Clean and preprocess the data. This includes handling missing values, outliers, and encoding categorical variables. Time series data may also require specific preprocessing steps like resampling, differencing, or decomposing.*

3. Feature Engineering: *Create relevant features that can help the model capture patterns and trends in the data. Feature engineering can include lag features, moving averages, and seasonality indicators.*

4. Splitting the Data: *Divide your dataset into training, validation, and test sets. The training set is used to train the model, the validation set helps with hyperparameter tuning, and the test set is reserved for final model evaluation.*

5. Model Selection: *Choose an appropriate machine learning or statistical model for electricity price prediction. Common choices include regression models (e.g., linear regression, random forest, or gradient boosting), time series models (e.g., ARIMA, SARIMA, or Prophet), or deep learning models (e.g., recurrent neural networks or LSTM).*

6. Model Training: *Train the selected model using the training dataset. Ensure that the model optimizes a relevant loss function, such as mean squared error (MSE) for regression tasks. Adjust hyperparameters as needed to improve model performance.*

7. Hyperparameter Tuning: *Use techniques like grid search or random search to fine-tune hyperparameters. This process helps you find the best configuration for your model.*

8. Model Validation: *Evaluate the model's performance on the validation dataset using appropriate evaluation metrics. Adjust the model and repeat training if necessary.*

9. Model Testing: *Once you're satisfied with the model's performance on the validation set, test it on the reserved test set to assess how well it generalizes to new, unseen data.*

10. Model Deployment: *If the model meets your performance requirements, deploy it to make real-time predictions on new electricity price data. Ensure that the deployment environment is scalable and reliable.*

11. Monitoring and Maintenance: *Continuously monitor the model's performance in a production environment and update it as needed. Electricity prices can be influenced by various factors that may change over time, so model maintenance is crucial.*

12. Interpretability and Visualization: *Provide clear explanations of the model's predictions, and use visualization techniques to communicate insights to stakeholders.*

Model Evaluation

1. Mean Absolute Error (MAE): *Calculate the absolute differences between predicted and actual prices, and then take the mean. It measures the average magnitude of errors.*

2. Mean Squared Error (MSE): *Square the differences between predicted and actual prices, and then take the mean. MSE gives more weight to larger errors.*

3. Root Mean Squared Error (RMSE): *Take the square root of the MSE. It's in the same unit as the target variable and provides a clearer interpretation.*

4. R-squared (R^2): *This measures the proportion of variance in the target variable that's predictable from the features. A higher R-squared indicates a better fit.*

5. Mean Absolute Percentage Error (MAPE): *Calculate the percentage difference between predicted and actual prices, and then take the mean. It's useful when you want to understand the error as a percentage of the actual values.*

6. Time Series-Specific Metrics: *If your electricity price data is time-series data, you may want to use metrics like Mean Absolute Scaled Error (MASE), Seasonal decomposition of time series (STL), or Autocorrelation to assess model performance.*

7. Cross-Validation: *Split your dataset into training and testing subsets, using techniques like k-fold cross-validation, time series cross-validation, or walk-forward validation. This helps you assess how well your model generalizes to new data.*

8. Visual Inspection: *Plot the predicted prices against the actual prices to visually assess how well the model captures trends and patterns.*

9. Residual Analysis: *Examine the residuals (the differences between actual and predicted prices) for any patterns or autocorrelation. This can help identify model deficiencies.*

10. Domain Expertise: *Consulting with domain experts in the energy industry can provide valuable insights into whether your model's predictions make practical sense.*

PROGRAM

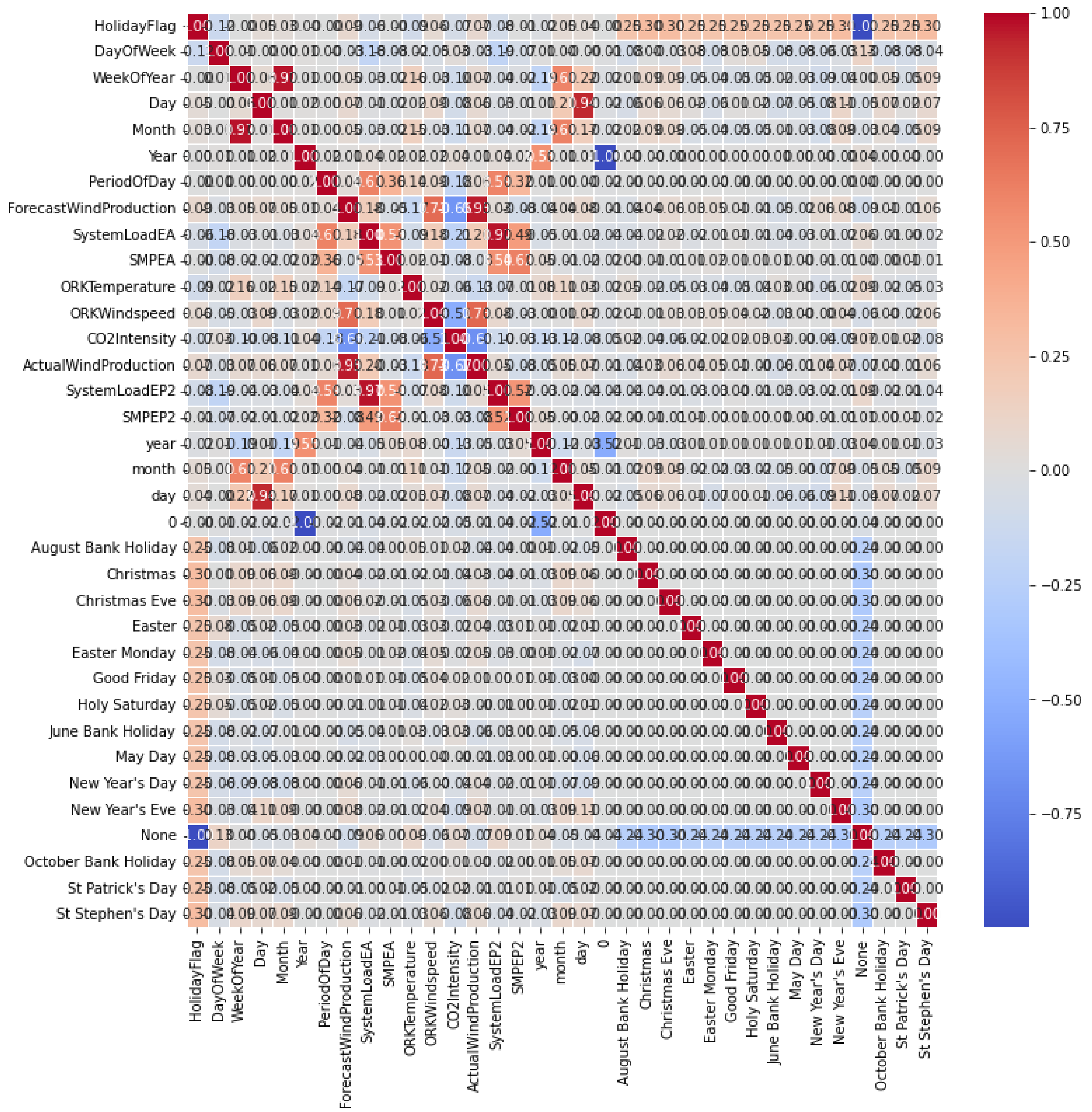
ELECTRICITY PRICES PREDICTION

Correlation Analysis:

In[1]:

```
plt.figure(figsize=(12,12))
sns.heatmap(df.corr(),annot=True,linewidths=0.7,fmt=".2f",cmap="coolwarm")
plt.show()
```

Out[1]:



In[2]:

```
cor=df.corr()["SMPEP2"].sort_values(ascending=False)
pd.DataFrame({"column":cor.index,"Correlation with
a":cor.values})
```

Out[2]:

	column	Correlation with a
0	SMPEP2	1.000000
1	SMPEA	0.617234
2	SystemLoadEP2	0.516938
3	SystemLoadEA	0.490945
4	PeriodOfDay	0.323052
5	year	0.047701
6	Year	0.017688
7	Easter	0.014242
8	St Patrick's Day	0.012972
9	Good Friday	0.011269
10	None	0.006365
11	May Day	0.004863
12	June Bank Holiday	0.004235
13	Holy Saturday	0.003777
14	October Bank Holiday	0.003084
15	Easter Monday	-0.001341
16	month	-0.001578
17	August Bank Holiday	-0.003159
18	HolidayFlag	-0.005645
19	ORKTemperature	-0.008571
20	New Year's Day	-0.009114
21	Christmas Eve	-0.009609
22	Christmas	-0.011435
23	New Year's Eve	-0.011679
24	Day	-0.013459
25	Month	-0.015255
26	0	-0.016091
27	WeekOfYear	-0.016170
28	St Stephen's Day	-0.018729
29	day	-0.019355
30	CO2Intensity	-0.033225
31	ORKWindspeed	-0.034614
32	DayOfWeek	-0.069597
33	ForecastWindProduction	-0.079611
34	ActualWindProduction	-0.082813

Modeling:

In[3]:

```
X=df.drop("SMPEP2",axis=1)
y=df["SMPEP2"]
```

In[4]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=0)
```

In[5]:

```
!pip install catboost
```

Out[5]:

```
Requirement already satisfied: catboost in /opt/conda/lib/python3.7/site-packages (1.1)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from catboost) (1.7.3)
Requirement already satisfied: graphviz in /opt/conda/lib/python3.7/site-packages (from catboost) (0.8.4)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.7/site-packages (from catboost) (3.5.3)
Requirement already satisfied: numpy>=1.16.0 in /opt/conda/lib/python3.7/site-packages (from catboost) (1.21.6)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (from catboost) (1.15.0)
Requirement already satisfied: pandas>=0.24.0 in /opt/conda/lib/python3.7/site-packages (from catboost) (1.3.5)
Requirement already satisfied: plotly in /opt/conda/lib/python3.7/site-packages (from catboost) (5.10.0)
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.24.0->catboost) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.7/site-packages (from pandas>=0.24.0->catboost) (2022.1)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib->catboost) (9.1.1)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib->catboost) (21.3)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (from matplotlib->catboost) (0.11.0)
```

```
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/site-pa
ckages (from matplotlib->catboost) (4.33.3)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/site-pac
kages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-pa
ckages (from matplotlib->catboost) (1.4.3)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.7/site-pack
ages (from plotly->catboost) (8.0.1)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.7/site-pa
ckages (from kiwisolver>=1.0.1->matplotlib->catboost) (4.4.0)
```

In[6]:

```
!pip install lightgbm
```

Out[6]:

```
Requirement already satisfied: lightgbm in /opt/conda/lib/python3.7/site-packages (3
.3.2)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from
 lightgbm) (1.21.6)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from
 lightgbm) (1.7.3)
Requirement already satisfied: scikit-learn!=0.22.0 in /opt/conda/lib/python3.7/site
-packages (from lightgbm) (1.0.2)
Requirement already satisfied: wheel in /opt/conda/lib/python3.7/site-packages (from
 lightgbm) (0.37.1)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-package
s (from scikit-learn!=0.22.0->lightgbm) (1.0.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.7/site
-packages (from scikit-learn!=0.22.0->lightgbm) (3.1.0)
```

In[7]:

```
!pip install xgboost
```

Out[7]:

```
Requirement already satisfied: xgboost in /opt/conda/lib/python3.7/site-packages (1.
6.2)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from
 xgboost) (1.7.3)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from
 xgboost) (1.21.6)
```

In[8]:

```
from xgboost import XGBRegressor
from catboost import CatBoostRegressor
from lightgbm import LGBMRegressor
```

In[9]:

```
ridge=Ridge().fit(X_train,y_train)
lasso=Lasso().fit(X_train,y_train)
enet=ElasticNet().fit(X_train,y_train)
knn=KNeighborsRegressor().fit(X_train,y_train)
ada=AdaBoostRegressor().fit(X_train,y_train)
```

In[10]:

```
svm=SVR().fit(X_train,y_train)
mlpc=MLPRegressor().fit(X_train,y_train)
dtc=DecisionTreeRegressor().fit(X_train,y_train)
rf=RandomForestRegressor().fit(X_train,y_train)
xgb=XGBRegressor().fit(X_train,y_train)
gbm=GradientBoostingRegressor().fit(X_train,y_train)
lgb=LGBMRegressor().fit(X_train,y_train)
catbost=CatBoostRegressor().fit(X_train,y_train)
```

Out[10]:

```
Learning rate set to 0.06876
0:	learn: 34.5202564	total: 64.1ms	remaining: 1m 4s
1:	learn: 33.6345770	total: 70ms	remaining: 34.9s
2:	learn: 32.8676895	total: 74.8ms	remaining: 24.8s
3:	learn: 32.1602282	total: 79.9ms	remaining: 19.9s
4:	learn: 31.4994441	total: 85.2ms	remaining: 17s
5:	learn: 30.9246676	total: 91ms	remaining: 15.1s
6:	learn: 30.4012671	total: 96.3ms	remaining: 13.7s
7:	learn: 29.9286504	total: 101ms	remaining: 12.5s
8:	learn: 29.5172364	total: 106ms	remaining: 11.7s
9:	learn: 29.1386738	total: 111ms	remaining: 11s
10:	learn: 28.7562568	total: 116ms	remaining: 10.4s
11:	learn: 28.4486528	total: 122ms	remaining: 10s
12:	learn: 28.1657127	total: 128ms	remaining: 9.7s
13:	learn: 27.8831288	total: 133ms	remaining: 9.38s
14:	learn: 27.6408000	total: 138ms	remaining: 9.07s
15:	learn: 27.4359313	total: 143ms	remaining: 8.8s
16:	learn: 27.2173157	total: 148ms	remaining: 8.55s
17:	learn: 27.0422970	total: 153ms	remaining: 8.36s
18:	learn: 26.8771233	total: 158ms	remaining: 8.16s
19:	learn: 26.6893652	total: 164ms	remaining: 8.03s
20:	learn: 26.5538529	total: 169ms	remaining: 7.86s
21:	learn: 26.4492316	total: 174ms	remaining: 7.71s
22:	learn: 26.3094102	total: 178ms	remaining: 7.58s
23:	learn: 26.1955562	total: 184ms	remaining: 7.47s
24:	learn: 26.0928013	total: 189ms	remaining: 7.39s
25:	learn: 25.9962431	total: 195ms	remaining: 7.3s
26:	learn: 25.9124311	total: 200ms	remaining: 7.19s
27:	learn: 25.8302408	total: 204ms	remaining: 7.09s
28:	learn: 25.7387686	total: 209ms	remaining: 7s
29:	learn: 25.6728034	total: 214ms	remaining: 6.92s
30:	learn: 25.5926581	total: 219ms	remaining: 6.84s
```

```
31:	learn: 25.4940606	total: 224ms	remaining: 6.77s
32:	learn: 25.4182581	total: 229ms	remaining: 6.7s
33:	learn: 25.3652752	total: 233ms	remaining: 6.63s
34:	learn: 25.3223579	total: 238ms	remaining: 6.56s
35:	learn: 25.2489125	total: 243ms	remaining: 6.5s
36:	learn: 25.2076996	total: 247ms	remaining: 6.43s
37:	learn: 25.1667858	total: 252ms	remaining: 6.38s
38:	learn: 25.1180288	total: 257ms	remaining: 6.34s
39:	learn: 25.0752794	total: 262ms	remaining: 6.3s
40:	learn: 25.0339522	total: 267ms	remaining: 6.25s
41:	learn: 24.9977064	total: 272ms	remaining: 6.2s
42:	learn: 24.9697517	total: 276ms	remaining: 6.15s
43:	learn: 24.9190191	total: 281ms	remaining: 6.11s
44:	learn: 24.8957832	total: 285ms	remaining: 6.06s
45:	learn: 24.8597053	total: 290ms	remaining: 6.01s
46:	learn: 24.8392094	total: 294ms	remaining: 5.97s
47:	learn: 24.8102239	total: 299ms	remaining: 5.92s
48:	learn: 24.7789045	total: 303ms	remaining: 5.89s
49:	learn: 24.7500880	total: 308ms	remaining: 5.85s
50:	learn: 24.7178926	total: 313ms	remaining: 5.82s
51:	learn: 24.6968006	total: 317ms	remaining: 5.78s
52:	learn: 24.6800087	total: 321ms	remaining: 5.74s
53:	learn: 24.6507233	total: 326ms	remaining: 5.71s
54:	learn: 24.6186034	total: 331ms	remaining: 5.68s
55:	learn: 24.5907057	total: 335ms	remaining: 5.65s
56:	learn: 24.5572812	total: 340ms	remaining: 5.63s
57:	learn: 24.5264215	total: 345ms	remaining: 5.6s
58:	learn: 24.4987088	total: 350ms	remaining: 5.58s
59:	learn: 24.4794572	total: 355ms	remaining: 5.56s
60:	learn: 24.4427210	total: 359ms	remaining: 5.53s
61:	learn: 24.4156283	total: 364ms	remaining: 5.51s
62:	learn: 24.3917659	total: 369ms	remaining: 5.49s
63:	learn: 24.3642679	total: 374ms	remaining: 5.47s
64:	learn: 24.3450265	total: 379ms	remaining: 5.45s
```

```
65:	learn: 24.3204788	total: 383ms	remaining: 5.42s
66:	learn: 24.2830890	total: 389ms	remaining: 5.41s
67:	learn: 24.2650713	total: 393ms	remaining: 5.39s
68:	learn: 24.2399899	total: 397ms	remaining: 5.36s
69:	learn: 24.2164965	total: 402ms	remaining: 5.34s
70:	learn: 24.1964369	total: 407ms	remaining: 5.32s
71:	learn: 24.1807660	total: 412ms	remaining: 5.31s
72:	learn: 24.1689245	total: 417ms	remaining: 5.29s
73:	learn: 24.1436655	total: 421ms	remaining: 5.27s
74:	learn: 24.1253300	total: 426ms	remaining: 5.25s
75:	learn: 24.0935058	total: 430ms	remaining: 5.23s
76:	learn: 24.0763530	total: 435ms	remaining: 5.21s
77:	learn: 24.0532679	total: 439ms	remaining: 5.19s
78:	learn: 24.0366265	total: 443ms	remaining: 5.17s
79:	learn: 24.0167814	total: 448ms	remaining: 5.15s
80:	learn: 23.9978824	total: 453ms	remaining: 5.14s
81:	learn: 23.9867891	total: 457ms	remaining: 5.12s
82:	learn: 23.9763622	total: 462ms	remaining: 5.1s
83:	learn: 23.9460132	total: 467ms	remaining: 5.09s
84:	learn: 23.9186528	total: 471ms	remaining: 5.07s
85:	learn: 23.9048177	total: 476ms	remaining: 5.06s
86:	learn: 23.8918401	total: 481ms	remaining: 5.05s
87:	learn: 23.8653562	total: 486ms	remaining: 5.04s
88:	learn: 23.8508496	total: 491ms	remaining: 5.02s
89:	learn: 23.8371390	total: 495ms	remaining: 5.01s
90:	learn: 23.8302205	total: 500ms	remaining: 4.99s
91:	learn: 23.8033151	total: 504ms	remaining: 4.98s
92:	learn: 23.7824900	total: 509ms	remaining: 4.96s
93:	learn: 23.7735478	total: 514ms	remaining: 4.96s
94:	learn: 23.7586709	total: 519ms	remaining: 4.94s
95:	learn: 23.7389694	total: 523ms	remaining: 4.93s
96:	learn: 23.7269390	total: 528ms	remaining: 4.91s
97:	learn: 23.7126293	total: 532ms	remaining: 4.9s
```

```
98:     learn: 23.6890770       total: 537ms    remaining: 4.88s
99:     learn: 23.6685269       total: 541ms    remaining: 4.87s
100:    learn: 23.6485375       total: 546ms    remaining: 4.86s
101:    learn: 23.6257557       total: 551ms    remaining: 4.85s
102:    learn: 23.6065188       total: 556ms    remaining: 4.84s
103:    learn: 23.5901493       total: 561ms    remaining: 4.83s
104:    learn: 23.5784682       total: 565ms    remaining: 4.82s
105:    learn: 23.5614537       total: 570ms    remaining: 4.81s
106:    learn: 23.5429277       total: 574ms    remaining: 4.79s
107:    learn: 23.5283062       total: 579ms    remaining: 4.78s
108:    learn: 23.5054101       total: 584ms    remaining: 4.77s
109:    learn: 23.4962552       total: 590ms    remaining: 4.78s
110:    learn: 23.4881442       total: 595ms    remaining: 4.77s
111:    learn: 23.4706061       total: 600ms    remaining: 4.76s
112:    learn: 23.4534164       total: 605ms    remaining: 4.75s
113:    learn: 23.4365195       total: 609ms    remaining: 4.73s
114:    learn: 23.4195741       total: 614ms    remaining: 4.72s
```

In[11]:

```
models=[ridge,lasso,dtc,rf,xgb,gbm,lgb,catbost,enet,knn,ada,mlpc,svm]
```

In[12]:

```
def ML(y,models):
  accuary=models.score(X_train,y_train)
  return accuary
```

In[13]:

```
for i in models:
   print(i,"Algorithm succed rate :",ML("SMPEP2",i))
```

Out[13]:

```
Ridge() Algorithm succed rate : 0.43121105926644243
Lasso() Algorithm succed rate : 0.42883198265818245
DecisionTreeRegressor() Algorithm succed rate : 1.0
RandomForestRegressor() Algorithm succed rate : 0.9424727172628374
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, ...) Algorithm succed rate : 0.8732530340524252
GradientBoostingRegressor() Algorithm succed rate : 0.5739399134995518
LGBMRegressor() Algorithm succed rate : 0.6953551703738294
<catboost.core.CatBoostRegressor object at 0x7f29e8bcc0d0> Algorithm succed rate : 0
.7878389350009978
ElasticNet() Algorithm succed rate : 0.4290970871174
KNeighborsRegressor() Algorithm succed rate : 0.5964451293083145
AdaBoostRegressor() Algorithm succed rate : 0.26365965295085403
MLPRegressor() Algorithm succed rate : 0.15355550237922466
SVR() Algorithm succed rate : 0.23458514968207922
```

In[14]:

```
cor=df.corr()["SMPEP2"].sort_values(ascending=False)
pd.DataFrame({"column":cor.index,"Correlation with
a":cor.values})
```

Out[14]:

	column	Correlation with a
0	SMPEP2	1.000000
1	SMPEA	0.617234
2	SystemLoadEP2	0.516938
3	SystemLoadEA	0.490945
4	PeriodOfDay	0.323052
5	year	0.047701
6	Year	0.017688
7	Easter	0.014242
8	St Patrick's Day	0.012972
9	Good Friday	0.011269
10	None	0.006365
11	May Day	0.004863
12	June Bank Holiday	0.004235
13	Holy Saturday	0.003777
14	October Bank Holiday	0.003084
15	Easter Monday	-0.001341
16	month	-0.001578
17	August Bank Holiday	-0.003159
18	HolidayFlag	-0.005645
19	ORKTemperature	-0.008571
20	New Year's Day	-0.009114
21	Christmas Eve	-0.009609
22	Christmas	-0.011435
23	New Year's Eve	-0.011679
24	Day	-0.013459
25	Month	-0.015255
26	0	-0.016091
27	WeekOfYear	-0.016170
28	St Stephen's Day	-0.018729
29	day	-0.019355
30	CO2Intensity	-0.033225
31	ORKWindspeed	-0.034614

In[15]:

```
X2=df[["SMPEA","SystemLoadEP2","SystemLoadEA","PeriodOfDay",
```

```
   "year","ActualWindProduction"]]
y2=df["SMPEP2"]
```

In[16]:

```
X_train2,X_test2,y_train2,y_test2=train_test_split(X2,y2,test_size=0.3,random_state=0)
```

In[17]:

```
rf2=RandomForestRegressor().fit(X_train2,y_train2)
```

In[18]:

```
rf2.score(X_train2,y_train2)
```

Out[18]:

```
0.9345853165856398
```

In[19]:

```
X3=df_remove_out.drop("SMPEP2",axis=1)
y3=df_remove_out["SMPEP2"]
```

In[20]:

```
X_train3,X_test3,y_train3,y_test3=train_test_split(X3,y3,test_size=0.3,random_state=0)
```

In[21]:

```
rf3=RandomForestRegressor().fit(X_train3,y_train3)
```

In[22]:

```
rf3.score(X_train,y_train)
```

Out[22]:

```
0.8965242074080007
```

In[23]:

```
dtc3=DecisionTreeRegressor().fit(X_train3,y_train3)
```

In[24]:

```
rf3.score(X_train3,y_train3)
```

Out[24]:

```
0.9525199962605772
```

Random Forest:

In[1]:

```
!pip install hyperopt
from hyperopt import tpe,STATUS_OK,Trials,fmin,hp
from hyperopt.pyll.base import scope
```

Out[1]:

```
Requirement already satisfied: hyperopt in /opt/conda/lib/python3.7/site-packages (0
.2.7)
Requirement already satisfied: scipy in /opt/conda/lib/python3.7/site-packages (from
 hyperopt) (1.7.3)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packages (from
hyperopt) (4.64.0)
Requirement already satisfied: py4j in /opt/conda/lib/python3.7/site-packages (from
hyperopt) (0.10.9.7)
```

```
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from
 hyperopt) (1.21.6)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-packages (from h
yperopt) (1.15.0)
Requirement already satisfied: cloudpickle in /opt/conda/lib/python3.7/site-packages
 (from hyperopt) (2.1.0)
Requirement already satisfied: networkx>=2.2 in /opt/conda/lib/python3.7/site-packag
es (from hyperopt) (2.5)
Requirement already satisfied: future in /opt/conda/lib/python3.7/site-packages (fro
m hyperopt) (0.18.2)
Requirement already satisfied: decorator>=4.3.0 in /opt/conda/lib/python3.7/site-pac
kages (from networkx>=2.2->hyperopt) (5.1.1)
```

In[2]:

```
space={
  "max_depth":hp.randint("max_depth",2,15),
  "min_samples_split":hp.randint("min_samples_split",2,20),
  "min_samples_leaf":hp.randint("min_samples_leaf",1,20),
  "n_estimators":hp.randint("n_estimators",50,1000)


}
```

In[3]:

```
def hyperparameter_tuning(params):
  clf=RandomForestRegressor(**params).fit(X_train,y_train)
  acc=rf.score(X_train,y_train)
  return acc
```

In[4]:

```
trials=Trials()


best=fmin(fn=hyperparameter_tuning,
     space=space,
```

```python
    algo=tpe.suggest,max_evals=100,trials=trials
    )
print("best:{}".format(best))
```

In[5]:

```python
best
```

Out[5]:

```
{'max_depth': 12,
 'min_samples_leaf': 2,
 'min_samples_split': 8,
 'n_estimators': 303}
```

Conclusion

In conclusion, predicting electricity prices is a multifaceted task that requires a well-structured process involving feature engineering, model training, and evaluation. Here are the key takeaways:

- ✓ *Feature engineering is essential for capturing the underlying factors affecting electricity prices.*
- ✓ *Time-based features, historical data, weather information, market conditions, and domain-specific features are crucial for creating informative input variables.*
- ✓ *Effective feature engineering improves the model's ability to capture patterns, trends, and relationships in the data.*

- ○ *The choice of the predictive model depends on the specific characteristics of the dataset and the goals of the prediction.*
- ○ *Linear regression, time series models, machine learning algorithms (e.g., decision trees, random forests, gradient boosting), and deep learning models can all be employed for electricity price prediction.*
- ○ *The training process involves optimizing model parameters and hyperparameters to achieve the best predictive performance.*

- ➢ *Proper model evaluation is crucial to ensure the model's reliability and predictive accuracy.*

- *Common evaluation metrics for regression tasks include Mean Squared Error (MSE), Mean Absolute Error (MAE), and R-squared (R^2).*
- *Time series-specific evaluation metrics and techniques, such as forecasting accuracy metrics and residual analysis, are important for time series data.*
- *The chosen model should be tested on a separate test dataset to assess its generalization performance.*

In summary, predicting electricity prices involves a data-driven approach that combines domain expertise, thoughtful feature engineering, model selection, and thorough evaluation. The accuracy of predictions can have a significant impact on energy market operations, planning, and decision-making. Ongoing monitoring and model maintenance are necessary to ensure the model continues to provide reliable forecasts in dynamic and changing environments.