PREDICTING ELECTRICITY PRICE PREDICTION USING DEEP LEARNING

TEAM LEADER

723721243038: PRAJEESH P

Phase 5 Submission Document

Project: Electricity Price Prediction

Topic: *In this section we will document the complete project and prepare it for submission*

Introduction:

- Electricity price prediction is a fascinating area of study that focuses on forecasting the future prices of electricity.
- It involves using historical data, along with various statistical and machine learning techniques, to analyze and predict the fluctuations in electricity prices.
- This prediction can be valuable for both consumers and energy providers, as it helps them make informed decisions regarding energy usage, pricing strategies, and resource allocation.
- The goal is to develop accurate models that consider factors such as market conditions, weather patterns, supply and demand dynamics.

- By leveraging advanced analytics and data-driven insights, electricity price prediction can contribute to optimizing energy consumption, reducing costs, and promoting sustainability

Data Source:

Dataset Link: (**https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction**)

Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspeed	CO2Intensity	ActualWindProduction	SystemLoadEP2	SMPEP2
1	11	2011	0	315.31	3388.77	49.26	6	9.3	600.71	356	3159.6	54.32
1	11	2011	1	321.8	3196.66	49.26	6	11.1	605.42	317	2973.01	54.23
1	11	2011	2	328.57	3060.71	49.1	5	11.1	589.97	311	2834	54.23
1	11	2011	3	335.6	2945.56	48.04	6	9.3	585.94	313	2725.99	53.47
1	11	2011	4	342.9	2849.34	33.75	6	11.1	571.52	346	2655.64	39.87
1	11	2011	5	342.97	2810.01	33.75	5	11.1	562.61	342	2585.99	39.87
1	11	2011	6	343.18	2780.52	33.75	5	7.4	545.81	336	2561.7	39.87
1	11	2011	7	343.46	2762.67	33.75	5	9.3	539.38	338	2544.33	39.87
1	11	2011	8	343.88	2766.63	33.75	4	11.1	538.7	347	2549.02	39.87
1	11	2011	9	344.39	2786.8	33.75	4	7.4	540.39	338	2547.15	39.87
1	11	2011	10	345.02	2817.59	33.75	4	7.4	532.3	372	2584.58	39.87
1	11	2011	11	342.23	2895.62	47.42	5	5.6	547.57	361	2641.37	39.87
1	11	2011	12	339.22	3039.67	44.31	5	3.7	556.14	383	2842.19	51.45
1	11	2011	13	335.39	3325.1	45.14	5	3.7	590.34	358	3082.97	51.45
1	11	2011	14	330.95	3661.02	46.25	4	9.3	596.22	402	3372.55	52.82
1	11	2011	15	325.93	4030	52.84	5	3.7	581.52	368	3572.64	53.65
1	11	2011	16	320.91	4306.54	59.44	5	5.6	577.27	361	3852.42	54.21
1	11	2011	17	365.15	4438.05	62.15	6	5.6	568.76	340	4116.03	58.33
1	11	2011	18	410.55	4585.84	61.81	8	7.4	560.79	358	4345.42	58.33
1	11	2011	19	458.56	4723.93	61.88	9	7.4	542.8	339	4427.29	58.33
1	11	2011	20	513.17	4793.6	61.46	?	?	535.37	324	4460.41	58.33
1	11	2011	21	573.36	4829.44	61.28	11	13	532.52	335	4493.22	58.27
1	11	2011	22	636.75	4888.29	61.63	11	22.2	534.34	372	4513.02	58.26
1	11	2011	23	683.59	4936.25	62.12	11	18.5	530.08	415	4490.71	58.26
1	11	2011	24	731.07	4995.51	62.83	11	22.2	517.55	513	4493.73	58.26
1	11	2011	25	780.23	5044.68	60.2	11	20.4	506.83	623	4481.31	58.15
1	11	2011	26	828.09	5018.8	56.25	12	20.4	513.98	683	4408.46	54.74
1	11	2011	27	873.81	4916.93	56.25	11	24.1	518.96	711	4341.14	54.74
1	11	2011	28	920.69	4933.87	56.25	12	22.2	525.69	761	4338.35	54.14
1	11	2011	29	985.09	4978.87	56.25	11	25.9	528.47	750	4294.17	53.63
1	11	2011	30	1044.37	5013.1	56.25	11	22.2	528.17	758	4318.87	53.63
1	11	2011	31	1098.97	5061.1	56.25	11	24.1	513.22	805	4375.62	53.63

Here's a list of tools and software commonly used in theprocess:

1. Programming Language:

- Python is the most popular language for machine learning due to its extensive libraries and frameworks. You can use libraries like NumPy, pandas, scikit-learn, and more.

Integrated Development Environment (IDE):

- Choose an IDE for coding and running machine learning experiments. Some popular options include Jupyter Notebook, GoogleColab, or traditional IDEs like PyCharm.

2. Machine Learning Libraries:

- You'll need various machine learning libraries, including:

- scikit-learn for building and evaluating machine learning models.

3. Data Visualization Tools:

- Tools like Matplotlib, Seaborn are essential for data exploration and visualization.

4. Data Preprocessing Tools:

- Libraries like pandas help with data cleaning, manipulation, and preprocessing.

5.Data Collection and Storage:

- Depending on your data source, you might need web scraping tools *(e.g., BeautifulSoup or Scrapy)* or databases *(e.g., SQLite, PostgreSQL)* for data storage.

5. Version Control:

- Version control systems like Git are valuable for tracking changes in your code and collaborating with others.

6.Notebooks and Documentation:

- Tools for documenting your work, such as Jupyter Notebooks or Markdown for creating *README* files and documentation.

7.Hyperparameter Tuning:

- Tools like GridSearchCV or RandomizedSearchCV from scikit-learn can help with hyperparameter tuning.

1..DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

1. **Empathize**: Gain a deep understanding of the users and stakeholders involved in electricity price prediction. This may include consumers, energy providers, regulators, and policymakers. Conduct interviews, surveys, and observations to understand their pain points, challenges, and goals related to electricity pricing.

2. **Define:** Define the specific problem or challenge related to electricity price prediction. This could involve identifying the key factors that influence electricity prices, such as supply and demand dynamics, renewable energy integration, regulatory policies, and market trends.

3. **Ideate:** Generate a wide range of ideas and potential solutions for electricity price prediction. Encourage brainstorming sessions and collaboration among diverse stakeholders. Explore different approaches, such as statistical modeling, machine learning algorithms, time series analysis, or a combination of techniques.

4. **Prototype:** Create prototypes or mockups of the potential solutions. This could involve developing a proof-of-concept model or a visualization tool that demonstrates how electricity price prediction could work. The prototype should be simple, low-cost, and easy to iterate upon.

5. **Test:** Gather feedback and test the prototypes with users and stakeholders. This could involve conducting user testing sessions, collecting data, and evaluating the performance of the prediction models. Iterate and refine the prototypes based on the feedback received.

6. **Implement:** Once a viable solution has been identified and refined, implement the electricity price prediction system. This may involve developing a software application, integrating the prediction model into existing energy management systems, or creating a user-friendly interface for consumers to access price information.

7. **Evaluate:** Continuously monitor and evaluate the performance of the electricity price prediction system. Collect feedback from users and stakeholders to identify areas for improvement and make necessary adjustments to enhance accuracy, usability, and overall effectiveness.

2.DESIGN INTO INNOVATION

1. **Identify emerging trends and technologies:** Stay updated with the latest advancements in the energy sector, such as renewable energy integration, smart grid technologies, energy storage, and demand response. Identify how these trends can impact electricity price prediction and create opportunities for innovation.

2. **Conduct market research:** Analyze the existing electricity price prediction solutions in the market. Identify their strengths, weaknesses, and gaps. Explore potential areas for improvement and innovation, such as enhancing accuracy, scalability, real-time prediction capabilities, or incorporating new data sources.

3. **Define the problem and goals:** Clearly define the problem statement and set specific goals for the innovation project. For example, the goal could be to develop a predictive model that accurately forecasts electricity prices at a granular level, considering various factors like weather patterns, renewable energy generation, and consumer behavior.

4. **Ideation and brainstorming:** Engage a diverse team of experts, including data scientists, domain specialists, and designers, to generate innovative ideas. Encourage out-of-the-box thinking and explore unconventional approaches, such as leveraging artificial intelligence, machine learning, big data analytics, or blockchain technology.

5. **Prototype and experimentation:** Develop prototypes or proof-of-concept models to test and validate the innovative ideas. Experiment with different algorithms, data sources, and modeling techniques. Leverage historical data and simulation tools to evaluate the performance and feasibility of the prototypes.

6. **Collaborate and iterate:** Foster collaboration and feedback loops with stakeholders, including energy providers, regulators, consumers, and researchers. Incorporate their insights and suggestions to refine the prototypes. Iterate on the design based on user feedback and real-world testing.

7. **Scalability and implementation:** Ensure that the innovative solution is scalable and can handle large volumes of data in real-time. Consider the infrastructure requirements, computational resources, and integration with existing systems. Develop a roadmap for implementation, considering factors like data acquisition, model deployment, and ongoing maintenance.

3.BUILD LOADING AND PREPROCESSING THE DATASET

Data Collection and Preprocessing:

- Importing the dataset: Obtain a comprehensive dataset containing relevant features such as ForeCastWind production, SystemLoadEA , SMPEA , ORKTemprature , ORKWindspeed etc.
- Data preprocessing: Clean the data by handling missing values, outliers, and categorical variables. Standardize or normalize numerical features

PROGRAM:

ELECTRICTY PRICE PREDICTION

 # IMPORTING REQUIRED PACKAGES

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import os

df = pd.read_csv("C:/Users/Lenovo/Desktop/Electricity updated.csv", low memory = False)

df
```

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWindspe
0	0	1	44	1	11	2011	0	315.31	3388.77	49.26	6	
1	0	1	44	1	11	2011	1	321.8	3196.66	49.26	6	1
2	0	1	44	1	11	2011	2	328.57	3060.71	49.1	5	1
3	0	1	44	1	11	2011	3	335.6	2945.56	48.04	6	
4	0	1	44	1	11	2011	4	342.9	2849.34	33.75	6	1
...	...	...	...	...	...	...	...	...	...	...	...	
38009	1	1	1	31	12	2013	43	1179.14	3932.22	34.51	6	2
38010	1	1	1	31	12	2013	44	1152.01	3821.44	33.83	5	2
38011	1	1	1	31	12	2013	45	1123.67	3724.21	31.75	4	2
38012	1	1	1	31	12	2013	46	1094.24	3638.16	33.83	5	1
38013	1	1	1	31	12	2013	47	1064	3624.25	33.83	5	1

 # CHECKING FOR NULL VALUES

df1 = df.isnull()

df1

	HolidayFlag	DayOfWeek	WeekOfYear	Day	Month	Year	PeriodOfDay	ForecastWindProduction	SystemLoadEA	SMPEA	ORKTemperature	ORKWinds
0	False	False	False	False	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	...	...	...	...	
38009	False	False	False	False	False	False	False	False	False	False	False	
38010	False	False	False	False	False	False	False	False	False	False	False	
38011	False	False	False	False	False	False	False	False	False	False	False	
38012	False	False	False	False	False	False	False	False	False	False	False	
38013	False	False	False	False	False	False	False	False	False	False	False	

 # ADDING NULL VALUES

df1 = df.isnull().sum()

df1

```
Out[5]: 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
        SystemLoadEP2.1          0
        SMPEP2.1                 0
        dtype: int64
```

CHECKING THE DATA TYPES

df2 = df.dtypes

df2

```
Out[6]: HolidayFlag               int64
        DayOfWeek                 int64
        WeekOfYear                int64
        Day                       int64
        Month                     int64
        Year                      int64
        PeriodOfDay               int64
        ForecastWindProduction   object
        SystemLoadEA             object
        SMPEA                    object
        ORKTemperature           object
        ORKWindspeed             object
        CO2Intensity             object
        ActualWindProduction     object
        SystemLoadEP2            object
        SMPEP2                   object
        SystemLoadEP2.1          object
        SMPEP2.1                 object
        dtype: object
```

 # REPLACING SPECIAL CHARACTER WITH NULL VALUE

df.replace(to_replace='?',value='0',inplace=True)

df

 # CHECKING FOR ANY SPECIAL CHARACTERS IN FEATURES

df4 = df[df['ORKWindspeed']=='?']

df4

 # COVERTING THE FILE AS CSV

```
df.to_csv(r'E:\file3.csv')
df
```

 # IMPORTING THE FILE

```
dtf1=pd.read_csv("E:/file3.csv")
dtf1
```

 # CONVERTING THE DATATYPES OF FEATURES

```
dtf1['ForecastWindProduction'] = dtf1['ForecastWindProduction'].astype(float)
dtf1['SystemLoadEA'] = dtf1['SystemLoadEA'].astype(float)
dtf1['SMPEA'] = dtf1['SMPEA'].astype(float)
dtf1
```

 # CHECKING THE DATA TYPES

```
dtf1.dtypes
```

```
Out[17]: Unnamed: 0                int64
         HolidayFlag               int64
         DayOfWeek                 int64
         WeekOfYear                int64
         Day                       int64
         Month                     int64
         Year                      int64
         PeriodOfDay               int64
         ForecastWindProduction  float64
         SystemLoadEA            float64
         SMPEA                   float64
         ORKTemperature            int64
         ORKWindspeed            float64
         CO2Intensity            float64
         ActualWindProduction      int64
         SystemLoadEP2           float64
         SMPEP2                  float64
         SystemLoadEP2.1         float64
         SMPEP2.1                float64
         dtype: object
```

4. PERFORMING DIFFERENT ACTIVITIES LIKE FEATURE ENGINEERING, MODEL TRAINING, EVALUATION **etc.,**

1. Prepare the data: This includes cleaning the data, removing outliers, and handling missing values.

2. Perform feature selection: This can be done using a variety of methods, such as correlation analysis, information gain, and recursive feature elimination.

3. Train the model: There are many different machine learning algorithms that can be used for electricity price prediction. Some popular choices include linear regression, random forests, and gradient boosting machines.

4. Evaluate the model: This can be done by calculating the mean squared error (MSE) or the root mean squared error (RMSE) of the model's predictions on the held-out test set.

5. Deploy the model: Once the model has been evaluated and found to be performing well, it can be deployed to production so that it can be used to predict the electricity prices.

Feature selection:

1. Identify the target variable. This is the variable that you want to predict, such as electricity price.
2. Explore the data. This will help you to understand the relationships between the different features and the target variable. You can use data visualization and correlation analysis to identify features that are highly correlated with the target variable.
3. Remove redundant features. If two features are highly correlated with each other, then you can remove one of the features, as they are likely to contain redundant information.
4. Remove irrelevant features. If a feature is not correlated with the target variable, then you can remove it, as it is unlikely to be useful for prediction.

PROGRAM:

```
import pandas as pd

import numpy as np

# importing dataset

dft1 = pd.read_csv("C:/Users/Lenovo/Downloads/Electricity.csv" , low_memory = False)

dft1

correlation_matrix = dtf1.corr()


# Get the absolute correlation values for the target variable

correlation_with_target = abs(correlation_matrix['SystemLoadEP2.1'])


# Sort the features based on their correlation with the target variable

sorted_features = correlation_with_target.sort_values(ascending=False)


# Select the top k features

k = 5  # Select top 5 features

selected_features = sorted_features[1:k+1]

# Exclude the target variable


# Print the selected feature names and their correlation values

print("Selected features and their correlation with the target variable:")

for feature, correlation in selected_features.items():

    print(f"{feature}: {correlation}")
```

```
Selected features and their correlation with the target variable:
SystemLoadEP2: 1.0
SystemLoadEA: 0.9725930860315541
PeriodOfDay: 0.5939813356813026
SMPEA: 0.5354531998087732
SMPEP2: 0.516934044199139
```

Model training:

1. Choose a machine learning algorithm. There are a number of different machine learning algorithms that can be used for electricity price prediction, such as linear regression, ridge regression.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Separate the features (X) and the target variable (y)
X = dataset.drop('SMPEP2', axis=1)
y = dataset['SMPEP2.1']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a linear regression model
model = LinearRegression()

# Train the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)
```

 # Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

```
Mean Squared Error: 1.4856130887245058e-26
```

RIDGE REGRESSION :

PROGRAM:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Create a ridge regression model

model = Ridge(alpha=0.5) # You can adjust the alpha value

 # Train the model on the training data

model.fit(X_train, y_train)

 # Make predictions on the testing data

y_pred = model.predict(X_test)

 # Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

```
Mean Squared Error: 3.802783296886913e-13
```

LASSO REGRESSION:

model = Lasso(alpha=0.5) # You can adjust the alpha value

 # Train the model on the training data

model.fit(X_train, y_train)

 # Make predictions on the testing data

y_pred = model.predict(X_test)

 # Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

```
Mean Squared Error: 0.00024767657356100355
```

SUPPORT VECTOR MACHINE:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Create a Support Vector Machine (SVM) model

model = SVR(kernel='linear') # You can choose different kernels like 'linear', 'rbf', 'poly', etc.

 # Train the model on the training data

model.fit(X_train, y_train)

 # Make predictions on the testing data

y_pred = model.predict(X_test)

 # Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

```
Mean Squared Error: 0.006036091532677277
```

RANDOM FOREST REGRESSION:

 # Split the dataset into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

 # Create a Random Forest Regression model

model = RandomForestRegressor(n_estimators=100, random_state=42) # You can adjust the number of estimators

 # Train the model on the training data

model.fit(X_train, y_train)

 # Make predictions on the testing data

y_pred = model.predict(X_test)

 # Calculate the mean squared error

mse = mean_squared_error(y_test, y_pred)

print("Mean Squared Error:", mse)

EVALUATION

PROGRAM:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Load your dataset and split it into features and target variable

X = dtf1[['SMPEA']]  # Features
y = dtf1['SMPEP2'] # Target variable

# Scale the features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Create a Support Vector Machine (SVM) model
svm = SVR(kernel='linear')  # You can choose different kernels like 'rbf', 'poly', etc.

# Train the SVM model
svm.fit(X_scaled, y)

# Create a Random Forest regression model
rf = RandomForestRegressor(n_estimators=100, random_state=42)
```

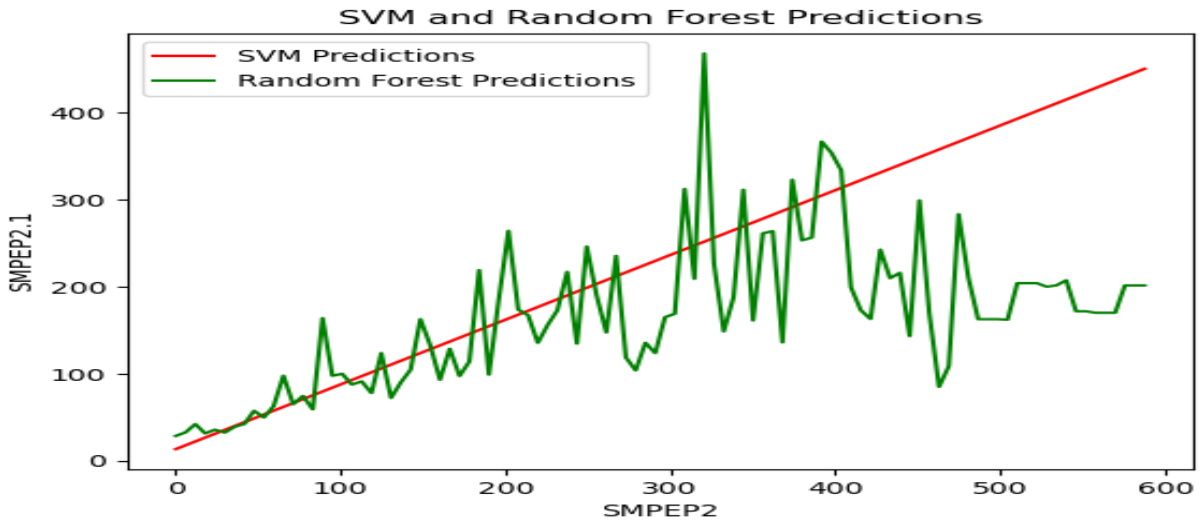
```
# You can adjust the number of estimators

# Train the Random Forest model
rf.fit(X, y)

# Generate predictions for the entire range of X values
X_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
svm_predictions = svm.predict(scaler.transform(X_range))
rf_predictions = rf.predict(X_range)

# Plot the predictions of SVM and Random Forest
plt.plot(X_range, svm_predictions, color='red', label='SVM Predictions')
plt.plot(X_range, rf_predictions, color='green', label='Random Forest Predictions')

plt.xlabel('SMPEP2')
plt.ylabel('SMPEP2.1')
plt.title('SVM and Random Forest Predictions')
plt.legend()
plt.show()
```



PROGRAM:

```
import matplotlib.pyplot as plt

import numpy as np

from sklearn.linear_model import LinearRegression, Ridge

from sklearn.preprocessing import StandardScaler

import pandas as pd


X = dtf1[['SMPEA']]  # Features

y = dtf1['SMPEP2']


# Scale the features using StandardScaler

scaler = StandardScaler()


X_scaled = scaler.fit_transform(X)

# Create a Linear Regression model

linear_reg = LinearRegression()


# Train the Linear Regression model

linear_reg.fit(X_scaled, y)


# Create a Ridge Regression model

ridge_reg = Ridge(alpha=0.5)  # You can adjust the regularization parameter
alpha
```

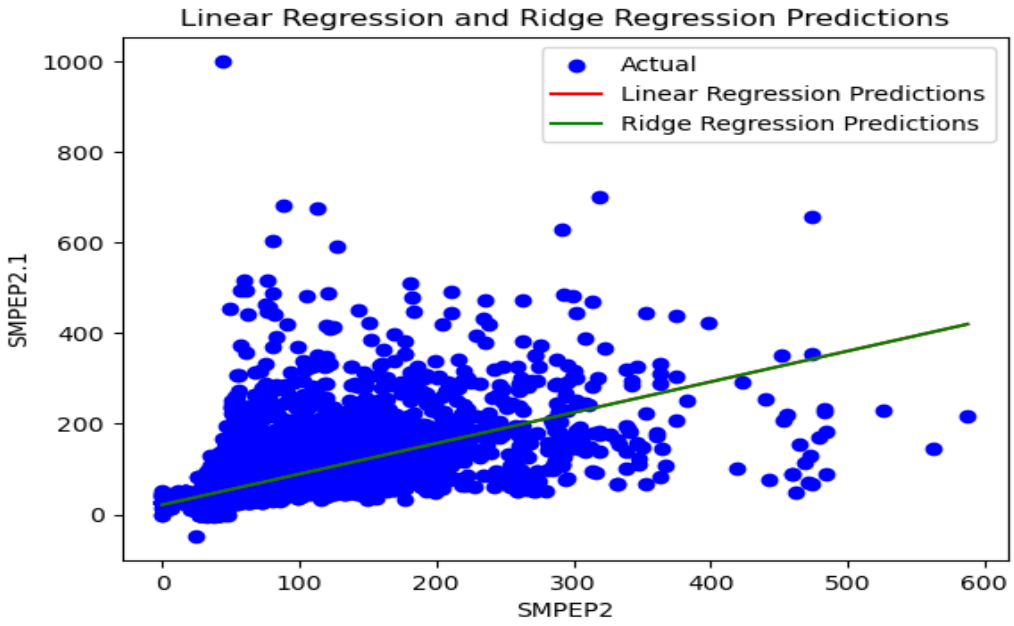
```
# Train the Ridge Regression model
ridge_reg.fit(X_scaled, y)

# Generate predictions for the entire range of X values
X_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
linear_reg_predictions = linear_reg.predict(scaler.transform(X_range))
ridge_reg_predictions = ridge_reg.predict(scaler.transform(X_range))

# Plot the actual data points
plt.scatter(X, y, color='blue', label='Actual')

# Plot the predictions of Linear Regression and Ridge Regression
plt.plot(X_range, linear_reg_predictions, color='red', label='Linear Regression Predictions')
plt.plot(X_range, ridge_reg_predictions, color='green', label='Ridge Regression Predictions')

plt.xlabel('SMPEP2')
plt.ylabel('SMPEP2.1')
plt.title('Linear Regression and Ridge Regression Predictions')
plt.legend()
plt.show()
```



Conclusion:

- Accurate electricity price prediction is a critical component of energy management, enabling utilities, businesses, and consumers to make informed decisions regarding consumption, investment in renewable energy sources, and overall cost management.

- With the integration of advanced data analytics, machine learning, and a deep understanding of market dynamics, we can strive to improve the precision of these predictions, ultimately contributing to a more sustainable and efficient energy future.