Project 4: Electricity Prices Prediction

Phase 1: Innovation

Team Members:
Ajita Fairen J - 715521106003 - B.E. ECE - 3rd year
M. Mekanya - 715521106028 - B.E. ECE - 3rd year
Shanmugapriya M - 715521106043 - B.E. ECE - 3rd year
K Shree Harini - 715521106044 - B.E. ECE - 3rd year
Vasanth L - 715521106055 - B.E. ECE - 3rd year
G. R. Tharunika - 715521106310 - B.E. ECE - 3rd year

Short explanation about the Problem Statement:

Develop a predictive model that leverages historical electricity prices and pertinent influencing factors to project forthcoming electricity prices. This model aims to empower energy providers and consumers with the capability to make well-informed decisions regarding their electricity consumption and investment strategies. By analyzing past pricing trends and considering factors like demand, weather conditions, and environmental impact, the model will offer valuable insights into future pricing, allowing stakeholders to optimize their operations and investments in the electricity market.

Dataset:

Source:
https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction/

This dataset contains information related to electricity markets and factors that can influence electricity prices. Here's a brief explanation of the columns:

1. DateTime:
   Records the timestamp, which can be used for time-based analysis.
2. Holiday:
   Indicates if a day is a holiday (categorical variable).
3. HolidayFlag:
   Possibly a binary flag related to holidays.
4. DayOfWeek:
   Indicates the day of the week (categorical variable).

5. WeekOfYear:
    Represents the week number within the year.
6. Day:
    Captures the day of the month.
7. Month:
    Records the month of the year.
8. Year:
    Specifies the year.
9. PeriodOfDay:
    Indicates a time period within a day (morning, afternoon, evening, etc.).
10. ForecastWindProduction:
    Predicted wind power production.
11. SystemLoadEA:
    Represents the electricity system load.
12. SMPEA:
    Likely the spot market price for energy in Australia (target variable).
13. ORKTemperature:
    Records temperature data.
14. ORKWindspeed:
    Captures wind speed data.
15. CO2 Intensity:
    Represents carbon dioxide intensity, possibly related to environmental factors.
16. ActualWindProduction:
    Actual wind power production.
17. SystemLoadEP2:
    Another measurement of the electricity system load.
18. SMPEP2:
    Possibly another instance of spot market price for energy.

This dataset seems to include a combination of temporal, weather, market, and environmental data. It can be valuable for analyzing and predicting electricity prices and related factors in the context of the energy market.

Columns we are going to use:

1. DateTime:
This column can be useful for capturing temporal patterns and trends in electricity prices. You may need to extract features such as the time of day or day of the week from this column to improve the model's performance.

2. SystemLoadEA:
This column represents the system load in the electricity grid, which is a critical factor affecting electricity prices. Including this as a feature is essential.

3. SMPEA:
SMPEA (Spot Market Price for Energy Australia) likely represents the target variable, i.e., the electricity prices you want to predict. You should include this column as the target variable in your model.

4. ORKTemperature:
Weather-related features, such as temperature, can have a significant impact on electricity prices. Including temperature data as a feature can help capture weather-related price fluctuations.

5. ORKWindspeed:
Similar to temperature, wind speed can also impact electricity prices, particularly in regions where wind power generation is significant. Include this column as a feature.

6. CO2Intensity:
Carbon dioxide (CO2) intensity can be another relevant factor in electricity prices, especially in regions where carbon pricing mechanisms are in place. Including this column can help account for environmental factors affecting prices.

7. ActualWindProduction:
Wind power generation can directly influence electricity prices. Including this column as a feature can capture the impact of wind power on prices.

Details of Libraries to be used:

1. Scikit-Learn (sklearn):
Scikit-Learn is a powerful machine learning library in Python that provides tools for data preprocessing, model training, and evaluation. You can use it to build and train your Random Forest Regressor model.

2. NumPy:
NumPy is essential for numerical computations and working with arrays, which are often required for data manipulation and feature engineering.

3. Pandas:
Pandas is useful for data manipulation and analysis. It allows you to load, clean, and preprocess your dataset effectively.

4. Matplotlib and Seaborn:
These libraries are handy for data visualization and exploring the results of your Random Forest Regressor model.

You can download and install Python libraries using a package manager called `pip`, which comes pre-installed with Python. To download and install a library, you can open your command prompt or terminal and run the following command:

pip install library_name

Replace `library_name` with the name of the library you want to download. H

pip install scikit-learn

pip install numpy

pip install pandas

pip install matplotlib

pip install seaborn

Running these commands will download and install the specified libraries and their dependencies. Make sure your Python environment is properly set up and that `pip` is in your system's PATH for the installation to work.

Training and Testing the Model: (Random Forest Regressor Model)

Implementation of a Random Forest Regressor for predicting electricity prices using Python and Scikit-Learn.

1. Import Necessary Libraries:
   - Import essential libraries such as NumPy, Pandas, Scikit-Learn, and Matplotlib.

2. Load and Prepare the Data:
   - Load your dataset from a CSV file using Pandas.
   - Split the dataset into features (X) and the target variable (y). In this case, 'SMPEA' is the target variable, and the other columns are used as features.

3. Split the Data:
   - Use the `train_test_split` function from Scikit-Learn to split the dataset into training and testing sets. The `test_size` parameter controls the proportion of the data used for testing.

4. Create and Train the Random Forest Regressor Model:
   - Create an instance of the Random Forest Regressor with specified hyperparameters, such as the number of trees (n_estimators) and a random seed for reproducibility (random_state).
   - Train the model using the training data (X_train and y_train) with the `.fit()` method.

5. Make Predictions:
   - Use the trained model to make predictions on the test data (X_test).

6. Evaluate the Model:
   - Calculate the Mean Squared Error (MSE) using the `mean_squared_error` function from Scikit-Learn. The code then calculates the Root Mean Squared Error (RMSE) by taking the square root of the MSE.
   - Print the RMSE to assess the model's performance.

7. Visualize the Predictions:
   - Create a scatter plot to compare the actual prices (y_test) and the predicted prices (y_pred). This visualization helps you understand how well the model's predictions align with the actual values.

Code:

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

```
# Load and prepare the data
data = pd.read_csv('your_dataset.csv')
X = data.drop('SMPEA', axis=1)  # Features (excluding the target variable)
y = data['SMPEA']  # Target variable

# Split the data 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 and train the Random Forest Regressor model
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
y_pred = rf_regressor.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(f'Root Mean Squared Error: {rmse}')

# Visualize the predictions
plt.scatter(y_test, y_pred)
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual Prices vs. Predicted Prices")
plt.show()
```

Accuracy test Metrics:

For electricity price prediction, the choice of evaluation metric can depend on the specific goals and characteristics of your project. However, some of the most commonly used metrics for electricity price prediction include:

1. Mean Absolute Error (MAE): MAE measures the average absolute difference between the actual and predicted electricity prices. It is useful for understanding the average magnitude of errors in price predictions.

2. Root Mean Squared Error (RMSE): RMSE calculates the square root of the average of the squared differences between actual and predicted electricity prices. RMSE provides a measure of prediction accuracy in the same unit as the target variable and is widely used in regression tasks.

3. R-squared (R^2) or Coefficient of Determination: R^2 measures the proportion of the variance in electricity prices that is predictable from the independent variables. It indicates how well the model fits the data and can provide insights into its predictive power.

4. Mean Absolute Percentage Error (MAPE): MAPE expresses the error as a percentage of the actual electricity prices, making it useful for understanding the relative accuracy of predictions, particularly when dealing with percentage changes.
