**Electricity Bill prediction**

**Problem Description: Electricity Pricing**

**Prediction:-**

The goal of this project is to develop a predictive model for electricity pricing. The provided dataset “Electricity\_Pricing.csv” contains various features related to electricity production and consumption, along with the corresponding pricing information. The dataset includes the following columns:

**Dataset Description:-**

1.Day: Day of the month (1-31)

2.Month: Month of the year (1-12).

3.ForecastWindProduction: Forecasted wind production in a specific unit.

4.SystemLoadEA: Electricity system load for a particular area.

5.SMPEA: Smart Market Price for Electricity A.

6.ORKTemperature: Temperature measured in the ORK region.

7.ORKWindspeed: Windspeed recorded in the ORK region.

8.CO2Intensity: CO2 intensity data.

9.ActualWindProduction: Actual wind production data.

10.SystemLoadEP2: Electricity system load for another specific area.

**Procedure:-**

1. Reads data from a CSV file named “Electricity\_Pricing.csv”.
2. Performs data preprocessing steps, including handling missing values and converting certain columns to numeric data types.
3. Creates visualizations such as a bar plot showing the relationship between the “Month” column and the target variable “SMPEP2”, as well as a heatmap showing the correlations between different features in the dataset.
4. Splits the data into training and testing sets.
5. Uses a Random Forest Regressor model to train on the training data.
6. Finally, the code makes a prediction for the target variable “SMPEP2” based on a specific set of feature values.

**Packages installed:-**

* Pandas: Pandas is a popular Python library used for data manipulation and analysis. It provides data structures and functions to make data analysis fast and easy.
* NumPy: NumPy is a fundamental package for scientific computing with Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
* Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It can be used to generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc.
* Seaborn: Seaborn is a data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
* Scikit-learn: Scikit-learn is a widely used machine learning library in Python. It provides simple and efficient tools for data mining and data analysis. In this project, it is used for tasks such as data splitting, model training, and model evaluation.

**Algorithm Choice:-**

* For time series forecasting, the choice of algorithm often depends on the specific characteristics of the data and the problem at hand. Some commonly used algorithms for time series forecasting include:
* ARIMA (AutoRegressive Integrated Moving Average): Suitable for univariate time series data, ARIMA models capture different aspects of time series data, including trends and seasonality.
* Prophet: Developed by Facebook, Prophet is designed for time series forecasting with added flexibility for handling various seasonal patterns and holidays.
* LSTM (Long Short-Term Memory) Networks: A type of recurrent neural network (RNN), LSTM networks are suitable for capturing long-term dependencies in time series data and are particularly effective for sequential data**.**

**Evaluation metrics:-**

* When evaluating the performance of a time series forecasting model, it is crucial to use appropriate evaluation metrics to assess the model’s accuracy and predictive capabilities. Some common evaluation metrics for time series forecasting include:
* Mean Absolute Error (MAE): Calculates the average of the absolute differences between predicted and actual values, providing a measure of the model’s forecasting accuracy.
* Mean Squared Error (MSE): Computes the average of the squared differences between predicted and actual values, giving more weight to large errors compared to MAE.
* Root Mean Squared Error (RMSE): RMSE is the square root of MSE and provides an interpretable measure of the model’s forecasting accuracy in the same unit as the target variable.
* Mean Absolute Percentage Error (MAPE): Measures the percentage difference between predicted and actual values, providing insights into the relative forecasting error.
* R-squared (R2): Determines the proportion of the variance in the dependent variable that is predictable from the independent variables, offering a measure of how well the model fits the data.

**Data Loading Process:-**

* Importing Libraries: The code begins with the import of necessary libraries, including Pandas and NumPy.
* Loading the CSV File: The line data=pd.read\_csv(“Electricity\_Pricing.csv”) loads the data from the “Electricity\_Pricing.csv” file and stores it in a variable named data.
* Displaying the Data: The line print(data.head()) is used to display the first few rows of the DataFrame, giving a quick overview of the data’s structure and contents.
* Examining Data Information: The line data.info() provides a concise summary of the DataFrame, including the number of non-null entries in each column and the data types.

**Data Preprocessing:-**

* Handling Missing Values: The code uses the pd.to\_numeric() function with the parameter errors=’coerce’ to convert specific columns, such as “ForecastWindProduction,” “SystemLoadEA,” “SMPEA,” “ORKTemperature,” “ORKWindspeed,” “CO2Intensity,” “ActualWindProduction,” and “SystemLoadEP2,” to numeric data types. The errors=’coerce’ parameter ensures that any non-numeric values are converted to NaN (Not a Number), which can be handled or removed later.
* Removing Rows with Missing Values: After converting the columns to numeric data types, the line data=data.dropna() removes any rows that contain missing values, effectively dropping rows with NaN values.
* These data preprocessing steps help ensure data consistency and integrity for subsequent analysis and modeling tasks. By handling missing values appropriately, the code prepares the data for further exploration and modeling, allowing for more accurate and reliable insights and predictions.

**Data Visualization:-**

* Bar Plot: The code uses Seaborn’s barplot function to create a bar plot, visualizing the relationship between the “Month” column and the target variable “SMPEP2.” This bar plot provides a graphical representation of the average “SMPEP2” values for each month, allowing for a quick comparison of pricing trends across different months.
* Correlation Heatmap: The code utilizes Matplotlib and Seaborn to create a correlation heatmap using the sns.heatmap function. This heatmap visualizes the Pearson correlation coefficients between different features in the dataset. The annot=True parameter adds numerical annotations to the heatmap, displaying the correlation values for each pair of features.

**Data Splitting:-**

* Importing Necessary Libraries: The code imports the train\_test\_split function from the sklearn.model\_selection module to facilitate the data splitting process.
* Data Splitting: The line xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.2, random\_state=42) splits the data into training and testing sets. It takes the feature variables x and the target variable y as input and splits them into training and testing sets, with 80% of the data allocated for training and 20% for testing. The random\_state=42 parameter ensures reproducibility by fixing the random seed for the splitting process.

**Model Training:-**

* Importing Necessary Libraries: The code imports the Random Forest Regressor algorithm from the sklearn.ensemble module to facilitate the model training process.
* Model Initialization and Training: The line model = RandomForestRegressor() initializes an instance of the Random Forest Regressor model. Subsequently, the model.fit(xtrain, ytrain) line fits the model to the training data, where xtrain represents the feature variables and ytrain represents the corresponding target variable. During the training process, the Random Forest Regressor algorithm learns the patterns and relationships present in the training data to make accurate predictions.
* Model Evaluation: The code does not explicitly include a model evaluation step. However, it is common practice to evaluate the trained model’s performance using various metrics, such as mean squared error, mean absolute error, or R-squared, to assess how well the model fits the training data and its ability to generalize to new, unseen data points.

**Prediction:-**

* Feature Selection: The line features=np.array([[10,12,54.10,4241.05,49.56,9.0,14.8,491.32,54.0,4426.84]]) selects a specific set of feature values, representing the following features: “Day,” “Month,” “ForecastWindProduction,” “SystemLoadEA,” “SMPEA,” “ORKTemperature,” “ORKWindspeed,” “CO2Intensity,” “ActualWindProduction,” and “SystemLoadEP2.” These features are used as input for the prediction.
* Model Prediction: The line model.predict(features) utilizes the trained Random Forest Regressor model to make predictions based on the selected features. The model takes the feature values as input and returns the predicted electricity pricing (SMPEP2) as the output.

**Conclusion:-**

The implemented machine learning model, based on the Random Forest Regressor algorithm, demonstrates promise in accurately predicting electricity bills. Leveraging key features such as day, month, forecasted and actual wind production, system load, temperature, wind speed, and CO2 intensity, the model offers insights into the factors influencing electricity pricing. Data preprocessing ensures data integrity, while visualizations aid in understanding feature correlations. The trained model effectively captures complex relationships within the data. However, a comprehensive evaluation of the model’s performance using standard metrics is necessary to ascertain its predictive capabilities on unseen data. Further model enhancements, including feature engineering and parameter optimization, could improve the model’s precision and robustness. With continued refinement and potential integration of additional relevant features, the model holds the potential to serve as a valuable tool for both consumers and electricity providers. It can facilitate informed decision-making and resource optimization, contributing to a more efficient and effective management of electricity pricing in the energy sector.