**Electricity Price Prediction using Python**

**Project description:**

This project aims to Predicting electricity prices using Python with sensor typically involves time series forecasting.Obtain historical electricity price data from reliable sources. You may also want to collect related data, like weather information, demand, or supply data.

**Data Collection:**

Gather historical electricity price data.

Collect sensor data that might influence electricity prices

(e.g., weather data, energy consumption, market trends).

**Import Required Libraries:**

import the necessary libraries, including Pandas.

**Load the Dataset:**

You'll need a dataset containing historical electricity prices. Make sure you have the dataset in a compatible format (e.g., CSV). Use Pandas to load the dataset into a DataFrame.

import pandas as pd

# Load the dataset

data = pd.read\_csv('electricity\_prices.csv')

**Explore the Data:**

It's essential to understand the structure of your dataset. Use functions like head(), info(), and describe() to get a sense of the data.

# Display the first few rows of the dataset

print(data.head())

# Get information about the dataset

print(data.info())

# Summary statistics

print(data.describe())

**Data Preprocessing:**

1. Load and merge the historical electricity price data with sensor data.

2.Handle missing values and outliers.

3.Convert timestamps to datetime objects if necessary.

Clean and preprocess the data. This involves handling missing values, outliers, and converting data into a suitable format.

**Feature Engineering:**

Create relevant features that might influence electricity prices. This can include lag features, seasonality indicators, and external factors like holidays.

Create relevant features from sensor data (e.g., average temperature, energy demand) that could impact electricity prices.

**Lag features:**

Include past electricity prices or sensor readings as input features for time series forecasting.

**Feature Selection:**

Choose the relevant features (independent variables) for your prediction model. In this case, features might include time, weather data, and any other relevant factors that affect electricity prices.

**Split Data:**

Split the dataset into training and testing sets, preserving the temporal order.

Divide the data into training and testing sets. A common split is 80% for training and 20% for testing.

**Model Training:**

Train the selected model using the training dataset.

**Model Evaluation:**

Evaluate the model's performance using appropriate metrics (e.g., Mean Absolute Error, Root Mean Squared Error) on the testing dataset.

**Hyperparameter Tuning:**

If using machine learning models, consider hyperparameter tuning to improve model performance.

**Prediction:**

Use the trained model to make future electricity price predictions based on sensor data.

**Visualization:**

Visualize the predicted prices alongside actual prices to assess model accuracy.

**Deployment:**

If needed, deploy the model for real-time or batch predictions.

Here's a simplified example of code to get you started using Python's scikit-learn library for regression with sensor data.

**Python program:**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error

# Load and preprocess your dataset

# 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)

# Choose and train a model (e.g., Linear Regression)

model = LinearRegression()

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluate the model

mae = mean\_absolute\_error(y\_test, y\_pred)

print(f"Mean Absolute Error: {mae}")