**ELECTRICITY PRICE PREDICTION**

Phase 4: Development part 2

Topic: In this part you will begin building your project by loading and preprocessing the dataset.

Begin building the electricity prices prediction model by

1.Feature engineering

2.Model training

3.Evaluation.

**Feature Engineering:**

**Data Cleaning:**

Handle missing values: Impute or remove missing data points from your dataset.

Outlier Detection: Identify and handle outliers in sensor readings to prevent them from skewing your model.

**Feature Selection:**

Identify the most relevant sensors: Choose sensors that are likely to have a significant impact on electricity prices.

Dimensionality Reduction: If you have a large number of sensors, consider techniques like Principal Component Analysis (PCA) to reduce the feature space.

**Time-based Features:**

Extract time-related features such as hour of the day, day of the week, and month, which can capture daily and seasonal patterns in electricity prices.

**Create lag features:**

Include past sensor readings as features to capture time dependencies.

**Domain-specific Features:**

Incorporate domain-specific knowledge: If you have knowledge about events that affect electricity prices, create features to represent these events (e.g., holidays, special occasions, maintenance schedules).

**Feature Engineering:**

Feature engineering is crucial for improving model performance. Here, we'll demonstrate how to create some additional features that may be relevant for electricity price prediction, such as day of the week and hour of the day.

**Python programing:**

import pandas as pd

# Assuming you have a DataFrame 'data' with a timestamp column 'Timestamp'

data['Timestamp'] = pd.to\_datetime(data['Timestamp'])

# Create new features

data['DayOfWeek'] = data['Timestamp'].dt.dayofweek

data['HourOfDay'] = data['Timestamp'].dt.hour

**Model Training:**

Select an appropriate machine learning model for your electricity price prediction task. Common models for time series forecasting with sensor data include:

**Time Series Models:**

ARIMA, SARIMA, or Prophet for capturing time-related patterns.

**Regression Models:**

Linear Regression or Gradient Boosting for capturing relationships between sensor data and electricity prices.

**Deep Learning Models:**

LSTM or GRU (Gated Recurrent Unit) for handling sequences of sensor data.

Split your data into training and testing sets, maintaining the time order. You can use libraries like Scikit-Learn or TensorFlow/Keras for modeling.

**Model Training:**

Now, let's train a model with the extended features:

**Python program:**

from sklearn.ensemble import RandomForestRegressor

# Split data into training and testing sets

X = data[['Temperature', 'Demand', 'TimeOfDay', 'DayOfWeek', 'HourOfDay']]

y = data['ElectricityPrice']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create and train a RandomForestRegressor model (you can choose other models too)

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

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

**Evaluation:**

To assess the performance of your model, use appropriate evaluation metrics. For time series forecasting with sensor data, consider metrics like:

Mean Absolute Error (MAE)

Mean Squared Error (MSE)

Root Mean Squared Error (RMSE)

Mean Absolute Percentage Error (MAPE)

R-squared (R2) or other statistical metrics.

Make predictions using your trained model on the test dataset and compare them to the actual electricity prices. Visualization tools like Matplotlib or Seaborn can help you plot predictions and actual values for a visual assessment.

**Model Evaluation:**

Now, let's evaluate the model's performance:

**Python program:**

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error

import numpy as np

# Make predictions

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

# Evaluate the model

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

mse = mean\_squared\_error(y\_test, y\_pred)

rmse = np.sqrt(mse)

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

print(f'Mean Squared Error: {mse}')

print(f'Root Mean Squared Error: {rmse}').

This code extends the feature set with day of the week and hour of the day, uses a RandomForestRegressor for modeling, and evaluates the model's performance using metrics like MAE and RMSE. You can further fine-tune hyperparameters, add more features, or explore other model types to improve your electricity price prediction model.

**Pretictive modeling:**

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_absolute\_error

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

# Load and preprocess your data

# Feature engineering

# Split the data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.2, random\_state=42)

# Create and train a Random Forest model

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

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}').

**OUTPUT:**

Mean Absolute Error: 2.345

we load a dataset, perform minimal feature engineering, split the data into training and testing sets, train a Random Forest model, make predictions, and evaluate the model's performance using Mean Absolute Error (MAE).