Phase 1 : Problem Definition and Design Thinking

Name : Dilli Babu D

Register Number : 312621243011

College Name : Thangavelu engineering college

Project 4: Electricity Prices Prediction

Objective :

- The electricity price prediction task is based on a case study where you need to predict daily price of electricity based on the daily consumption of heavy machinery used by businessess.

Problem Statement :

- Create a predictive model that utilizes historical electricity prices and relevant factors to forecast future electricity prices, assisting energy providers and consumers in making informed decisions regarding consumption and investment.

Problem Definition :

- The problem is to develop a predictive model that uses historical electricity prices and relevant factors to forecast future electricity prices. The objective is to create a tool that assists both energy providers and consumers in making informed decisions regarding consumption and investment by predicting future electricity prices. This project involves data preprocessing, feature engineering, model selection, training, and evaluation.

Design Thinking :

Data Source： Utilize a dataset containing historical electricity prices and relevant factors like date, demand, supply, weather conditions, and economic indicators.

```
Load your electricity price dataset

import pandas as pd
data = pd.read_csv('Electricity.csv')
```

Data Preprocessing :

```
print(data.describe())  # Summary statistics
print(data.isnull().sum())  # Check for missing values


# Handle missing values (if any)
data.fillna(data.mean(), inplace=True)


# Remove duplicate values (if any)
data = data.drop_duplicates()
```

Feature Engineering :

```
# Time-based features
data['Date'] = pd.to_datetime(data['Date'])
data['Year'] = data['Date'].dt.year
```

```
data['Month'] = data['Date'].dt.month
data['DayOfWeek'] = data['Date'].dt.dayofweek

# Lagged variables
data['ElectricityPrice_Lag1'] = data['ElectricityPrice'].shift(1)
data['ElectricityPrice_Lag7'] = data['ElectricityPrice'].shift(7)
```

Model Selection :

```
from statsmodels.tsa.arima_model import ARIMA

# Define the ARIMA order (p, d, q)
p = 1  # Example value
d = 1  # Example value
q = 1  # Example value

# Create the ARIMA model
model = ARIMA(data['ElectricityPrice'], order=(p, d, q))

# Fit the model to the data
model_fit = model.fit()

# Print the summary of the model
```

```
print(model_fit.summary())
```

Model Training :

```
# Split the data into training and testing sets
train_size = int(len(data) * 0.8)
train, test = data['ElectricityPrice'][:train_size], data['ElectricityPrice'][train_size:]

# Initialize and fit the ARIMA model on the training data
model = ARIMA(train, order=(p, d, q))
model_fit = model.fit()

# Print the summary of the model
print(model_fit.summary())
```

Evaluation :

```
# Make predictions on the test set
predictions = model_fit.forecast(steps=len(test))

# Calculate MAE, MSE, RMSE (import necessary libraries)
from sklearn.metrics import mean_absolute_error, mean_squared_error
import math
```

```
mae = mean_absolute_error(test, predictions)
mse = mean_squared_error(test, predictions)
rmse = math.sqrt(mse)

# Print the evaluation results
print(f'Mean Absolute Error (MAE): {mae}')
 print(f'Mean Squared Error (MSE): {mse}')
 print(f'Root Mean Squared Error (RMSE): {rmse}')
```

codings:

```
Load your electricity price dataset
import pandas as pd
data = pd.read_csv('Electricity.csv')
print(data.describe())  # Summary statistics
print(data.isnull().sum())  # Check for missing values

# Handle missing values (if any)
data.fillna(data.mean(), inplace=True)

# Remove duplicate values (if any)
data = data.drop_duplicates()
```

```
# Time-based features
data['Date'] = pd.to_datetime(data['Date'])
data['Year'] = data['Date'].dt.year
data['Month'] = data['Date'].dt.month
data['DayOfWeek'] = data['Date'].dt.dayofweek

# Lagged variables
data['ElectricityPrice_Lag1'] = data['ElectricityPrice'].shift(1)
data['ElectricityPrice_Lag7'] = data['ElectricityPrice'].shift(7)
from statsmodels.tsa.arima_model import ARIMA

# Define the ARIMA order (p, d, q)
p = 1  # Example value
d = 1  # Example value
q = 1  # Example value

# Create the ARIMA model
model = ARIMA(data['ElectricityPrice'], order=(p, d, q))

# Fit the model to the data
model_fit = model.fit()
```

```
# Print the summary of the model
print(model_fit.summary())
# Split the data into training and testing sets
train_size = int(len(data) * 0.8)
train, test = data['ElectricityPrice'][:train_size],
data['ElectricityPrice'][train_size:]
```

```
# Initialize and fit the ARIMA model on the training data
model = ARIMA(train, order=(p, d, q))
model_fit = model.fit()

# Print the summary of the model
print(model_fit.summary())
# Make predictions on the test set
predictions = model_fit.forecast(steps=len(test))
```

```
# Calculate MAE, MSE, RMSE (import necessary libraries)
from sklearn.metrics import mean_absolute_error,
mean_squared_error
import math

mae = mean_absolute_error(test, predictions)
mse = mean_squared_error(test, predictions)
```

```
rmse = math.sqrt(mse)


# Print the evaluation results
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

NOTE : Run the program with compiler with csv.file