**PHASE 4**

**DEVELOPMENT**

**FEATURE ENGINEERING**

Feature engineering is a crucial step in building effective predictive models, especially for time series data like electricity prices

import pandas as pd

# Load your dataset (assuming it has columns like ‘Date’ and ‘Price’)

Data = pd.read\_csv(‘electricity\_prices.csv’)

# Step 1: Extract Date Features

Data[‘Date’] = pd.to\_datetime(data[‘Date’])

Data[‘Year’] = data[‘Date’].dt.year

Data[‘Month’] = data[‘Date’].dt.month

Data[‘Day’] = data[‘Date’].dt.day

Data[‘Weekday’] = data[‘Date’].dt.weekday

# Step 2: Lag Features

Data[‘Price\_Lag1’] = data[‘Price’].shift(1) # Lag of 1 day

Data[‘Price\_Lag7’] = data[‘Price’].shift(7) # Lag of 1 week

# Step 3: Rolling Statistics

Data[‘Rolling\_Mean’] = data[‘Price’].rolling(window=7).mean() # 7-day rolling mean

Data[‘Rolling\_Std’] = data[‘Price’].rolling(window=7).std() # 7-day rolling standard deviation

# Step 4: Exponential Moving Average (EMA)

Data[‘EMA’] = data[‘Price’].ewm(span=7, adjust=False).mean() # 7-day EMA

# Step 5: Remove NaN values after feature engineering

Data = data.dropna()

# Save the engineered dataset

Data.to\_csv(‘engineered\_electricity\_prices.csv’, index=False)

**MODEL TRAINING**

To train your electricity prices prediction model using the engineered features, we can follow these steps:

import pandas as pd

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

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

# Load your engineered dataset

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

# Define your features and target variable

X = data[['Year', 'Month', 'Day', 'Weekday', 'Price\_Lag1', 'Price\_Lag7', 'Rolling\_Mean', 'Rolling\_Std', 'EMA']]

y = data['Price']

# Normalize the features

scaler = MinMaxScaler()

X\_scaled = scaler.fit\_transform(X)

# Split the data into training and testing sets

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

# Reshape the data for LSTM

X\_train = X\_train.reshape((X\_train.shape[0], 1, X\_train.shape[1]))

X\_test = X\_test.reshape((X\_test.shape[0], 1, X\_test.shape[1]))

# Build the LSTM model

model = Sequential([

LSTM(50, activation='relu', input\_shape=(X\_train.shape[1], X\_train.shape[2])),

Dense(1)

])

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Train the model

model.fit(X\_train, y\_train, epochs=50, batch\_size=32, validation\_data=(X\_test, y\_test))

**EVALUATION**:

To evaluate the performance of the electricity prices prediction model, you can use various metrics commonly used for regression tasks

# Evaluate the model

Loss = model.evaluate(X\_test, y\_test, verbose=0)

Print(f’Mean Squared Error (MSE): {loss}’)

# Calculate additional metrics if needed

From sklearn.metrics import mean\_absolute\_error, r2\_score

Predictions = model.predict(X\_test)

Mae = mean\_absolute\_error(y\_test, predictions)

R2 = r2\_score(y\_test, predictions)

Print(f’Mean Absolute Error (MAE): {mae}’)

Print(f’R-squared (R2) Score: {r2}’)

These metrics provide different perspectives on the model’s performance. Lower MSE and MAE values are better, while a higher R2 score is desirable

**SUMMARY**:

By running these codes, you’ll get a summary of the model’s performance. This information will help us understand how well the model is doing at predicting electricity prices.