Aim

The aim of this code is to develop a linear regression model to predict electricity prices using lag features, difference features, and interaction features.

Feature details

* Lag features: Lag features are the values of the target variable at previous time steps. In this case, the lag features are the electricity prices at the previous 23 hours.
* Difference features: Difference features are the changes in the target variable between successive time steps. In this case, the difference features are the changes in electricity prices between hourly intervals.
* Interaction features: Interaction features are the products of two or more features. In this case, the interaction features are the products of the lag features, difference features, and other features.

Program:

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

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

from sklearn.preprocessing import StandardScaler

# Load the dataset

df = pd.read\_csv('https://www.kaggle.com/datasets/chakradharmattapalli/electricity-price-prediction/download')

# Preprocess the data

# Convert the date column to datetime format

df['Date'] = pd.to\_datetime(df['Date'])

# Create a new column for the time of day

df['Time'] = df['Date'].dt.hour

# Create a new column for the day of the week

df['Day of Week'] = df['Date'].dt.dayofweek

# Create a new column for the month

df['Month'] = df['Date'].dt.month

# Drop the Date column

df.drop('Date', axis=1, inplace=True)

# Fill in missing values

df.fillna(method='ffill', inplace=True)

# Create lag features

lag\_features = []

for lag in range(1, 24):

lag\_feature = df['Electricity Price'].shift(lag)

lag\_features.append(lag\_feature)

# Add the lag features to the features array

features = np.concatenate((features, np.array(lag\_features)), axis=1)

# Create difference features

difference\_features = []

for i in range(1, len(features)):

difference\_feature = features[i] - features[i - 1]

difference\_features.append(difference\_feature)

# Add the difference features to the features array

features = np.concatenate((features, np.array(difference\_features)), axis=1)

# Create interaction features

interaction\_features = []

for i in range(len(features)):

for j in range(len(features)):

interaction\_feature = features[i] \* features[j]

interaction\_features.append(interaction\_feature)

# Add the interaction features to the features array

features = np.concatenate((features, np.array(interaction\_features)), axis=1)

# Scale the features

scaler = StandardScaler()

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

# Set the features to the scaled features

features = features\_scaled

# Convert the target variable to a NumPy array

target = df['Electricity Price'].to\_numpy()

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target, test\_size=0.25, random\_state=42)

# Train a linear regression model

model = LinearRegression()

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

# Evaluate the model on the test data

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

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

rmse = np.sqrt(mse)

r2 = r2\_score(y\_test, y\_pred)

# Print the evaluation metrics

print('MSE:', mse)

print('RMSE:', rmse)

print('R2:', r2)

# Make predictions on the test data

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

# Visualize the predictions

plt.scatter(X\_test[:, 1], y\_test, color='blue', label='Actual')

plt.scatter(X\_test[:, 1], y\_test\_pred, color='red', label='Predicted')

plt.legend()

plt.title('Electricity Price Prediction')

plt.xlabel('Time of Day (Hour)')

plt.ylabel('Electricity Price')

plt.show()