**Develop a linear regression model for forecasting time series data for the given dataset**

**EX.No:4**

**DATE:**

**AIM:**

To develop a Linear Regression model for forecasting future values in a time series dataset by analyzing historical trends. The goal is to provide a simple, interpretable approach to predict upcoming data points based on time-dependent patterns.

**ALGORITHM:**

1. Load data, explore for missing values and target variable distribution.
2. Clean data, handle missing entries, remove duplicates, address outliers, convert 'DATE\_TIME'.
3. Create lagged, time-based, and rolling window features.
4. Split data into training, validation, and test sets, keeping temporal order.
5. Train a linear regression model, evaluate using MAE, RMSE, and R2 on validation data.
6. Evaluate model on test data, deploy for predictions on new data.

**CODE:**

import pandas as pd

from IPython.display import display

df = pd.read\_csv('Plant\_1\_Generation\_Data.csv')

display(df.head())

print(df.dtypes)

print(df.isnull().sum())

print(df['AC\_POWER'].describe())

import matplotlib.pyplot as plt

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

plt.figure(figsize=(12, 6))

plt.plot(df['DATE\_TIME'], df['AC\_POWER'])

plt.xlabel('Date and Time')

plt.ylabel('AC Power')

plt.title('AC Power Over Time')

plt.show()

numerical\_df = df.select\_dtypes(include=['number'])

correlation\_matrix = numerical\_df.corr()

print(correlation\_matrix['AC\_POWER'].sort\_values(ascending=False))

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

print("Missing values:\n", df.isnull().sum())

print("\nDuplicate timestamps:", df.duplicated(subset=['DATE\_TIME']).sum())

df = df.sort\_values('DATE\_TIME')

df = df.drop\_duplicates(subset=['DATE\_TIME'], keep='first')

Q1 = df['AC\_POWER'].quantile(0.25)

Q3 = df['AC\_POWER'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

df = df[(df['AC\_POWER'] >= lower\_bound) & (df['AC\_POWER'] <= upper\_bound)]

df = df.set\_index('DATE\_TIME')

df['AC\_POWER\_lag\_1d'] = df['AC\_POWER'].shift(24)

df['AC\_POWER\_lag\_2d'] = df['AC\_POWER'].shift(48)

df['AC\_POWER\_lag\_7d'] = df['AC\_POWER'].shift(168)

df['day\_of\_week'] = df.index.dayofweek

df['month'] = df.index.month

df['year'] = df.index.year

target = 'AC\_POWER'

df['rolling\_mean\_7d'] = df['AC\_POWER'].rolling(window=168).mean()

df = df.dropna()

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

X = df.drop('AC\_POWER', axis=1)

y = df['AC\_POWER']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, shuffle=False)

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, shuffle=False)

from sklearn.linear\_model import LinearRegression

X\_train = X\_train.select\_dtypes(include=['number'])

X\_val = X\_val.select\_dtypes(include=['number'])

model = LinearRegression()

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

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

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import numpy as np

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

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

rmse = np.sqrt(mse)

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

print("Mean Absolute Error (MAE):", mae)

print("Root Mean Squared Error (RMSE):", rmse)

print("R-squared (R2) Score:", r2)

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))

plt.plot(y\_val.index, y\_val, label='Actual')

plt.plot(y\_val.index, y\_pred\_val, label='Predicted')

plt.xlabel('Date and Time')

plt.ylabel('AC Power')

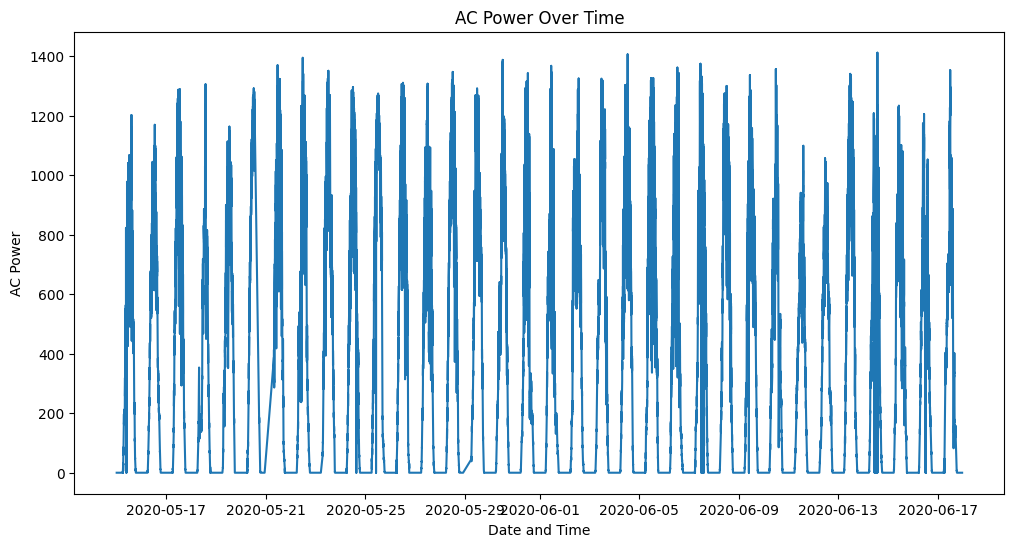
plt.title('Actual vs. Predicted AC Power')

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT:**



**RESULT:**

Thus the program has been completed and verified successfully.