**EX:No.3 221501013**

**07/01/25**

**IMPLEMENTING LINEAR REGRESSION MODEL USING TIME SERIES DATASET**

**AIM:**

To implement linear regression model using time series dataset.

**PROCESS:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.linear\_model import LinearRegression

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

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

**# Load the dataset**

file\_path = r'Electric\_Production.csv'

data = pd.read\_csv(file\_path)

**# Check the columns to see what data is available**

print("Columns in the dataset:", data.columns)

**# Ensure there are no leading/trailing spaces in column names**

data.columns = data.columns.str.strip()

**# Check if 'IPG2211A2N' column exists**

if 'IPG2211A2N' not in data.columns:

raise KeyError("'IPG2211A2N' column not found in the dataset. Please check the column name.")

**# Access the 'IPG2211A2N' column**

production\_data = data['IPG2211A2N']

**# Reverse the order of the data to maintain chronological order**

production\_data\_reverse = production\_data.iloc[::-1]

**# Reset index to maintain the correct time series order**

production\_data\_reverse.reset\_index(drop=True, inplace=True)

**# Handle Missing Values**

data['IPG2211A2N'].fillna(data['IPG2211A2N'].mean(), inplace=True)

**# Handle Outliers using IQR for 'IPG2211A2N' column**

Q1 = data['IPG2211A2N'].quantile(0.25)

Q3 = data['IPG2211A2N'].quantile(0.75)

IQR = Q3 - Q1

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

**# Filter data to remove outliers based on the IQR bounds**

data = data[(data['IPG2211A2N'] >= lower\_bound) & (data['IPG2211A2N'] <= upper\_bound)]

**# Prepare the data for Linear Regression**

**# We'll use time (index) as the independent variable (X) and electric production as the dependent variable (y)**

X = np.array(range(len(production\_data\_reverse))).reshape(-1, 1) # Time index as the feature

y = production\_data\_reverse.values # Electric production as the target

**# Split the data into training and testing sets (80% training, 20% testing)**

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

**# Initialize the Linear Regression model**

model = LinearRegression()

**# Train the model**

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

**# Make predictions on the test set**

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

**# Evaluate the model**

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

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

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

print(f"R-squared: {r2}")

**# Plotting the results**

plt.figure(figsize=(10, 6))

**# Plot the original data**

plt.scatter(X, y, color='blue', alpha=0.5, label="Original Data")

**# Plot the regression line**

plt.plot(X, model.predict(X), color='red', label="Linear Regression Line")

plt.title('Linear Regression - Electric Production Data')

plt.xlabel('Time')

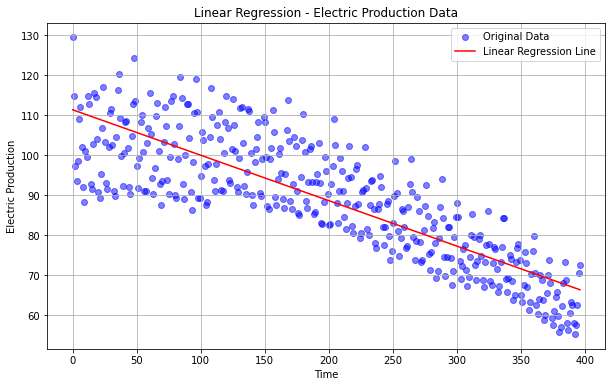
plt.ylabel('Electric Production')

plt.legend()

plt.grid(True)

plt.show()

**OUTPUT:**

****

**RESULT:**

The program to implement linear regression is created and executed successfully.