**Develop a linear regression model for forecasting time series data**

**EX:No.4**

**DATE: 15/02/25**

# AIM:

To implement a program that develops a linear regression model for forecasting time series sales data.

## OBJECTIVE:

To forecast office supply sales using a linear regression model trained on cleaned and prepared time series data.

## BACKGROUND:

* Linear regression is a widely used method for modeling relationships between variables.
* In time series forecasting, it uses past values to predict future ones.
* Forecasting helps businesses plan ahead by estimating future trends based on historical data.
* Simple models like linear regression are easy to implement and interpret.

## SCOPE OF THE PROGRAM:

* Load and preprocess the office supply sales dataset
* Prepare time series data for forecasting
* Train a linear regression model on the prepared data
* Predict future sales using the model
* Visualize actual vs predicted sales

## ALGORITHM:

* Import required libraries (pandas, matplotlib, sklearn)
* Load and clean the sales dataset
* Set the date column as the index and resample by day
* Prepare lagged features from the data
* Train a linear regression model
* Make predictions on test data
* Plot actual and predicted sales for visual comparison

**CODE:**

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

import numpy as np

# Load the dataset

df = pd.read\_csv("cleaned\_sales\_data.csv")

# Convert 'Order Date' to datetime

df['Order Date'] = pd.to\_datetime(df['Order Date'], errors='coerce')

df.set\_index('Order Date', inplace=True)

# Daily aggregated sales

df\_daily = df['Sales'].resample('D').sum()

# Estimate trend using 30-day moving average

window\_size = 30

df\_trend = df\_daily.to\_frame(name='Sales')

df\_trend['Moving\_Avg'] = df\_trend['Sales'].rolling(window=window\_size).mean()

# Detrending: Remove the estimated trend

df\_trend['Detrended'] = df\_trend['Sales'] - df\_trend['Moving\_Avg']

# Drop NaNs caused by moving average

df\_trend.dropna(inplace=True)

# Create lag feature on detrended sales

df\_trend['Lagged\_Detrended'] = df\_trend['Detrended'].shift(1)

# Drop resulting NaNs

df\_trend.dropna(inplace=True)

# Train/test split

split\_idx = int(len(df\_trend) \* 0.8)

train = df\_trend.iloc[:split\_idx]

test = df\_trend.iloc[split\_idx:]

# Features and targets

x\_train = train[['Lagged\_Detrended']]

y\_train = train['Detrended']

x\_test = test[['Lagged\_Detrended']]

y\_test = test['Detrended']

# Train linear regression on detrended data

model = LinearRegression()

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

# Predict detrended sales

detrended\_preds = model.predict(x\_test)

# Add back the trend to get final predicted sales

test = test.copy()

test['Predicted\_Detrended'] = detrended\_preds

test['Final\_Predicted\_Sales'] = test['Predicted\_Detrended'] + test['Moving\_Avg']

# Plot: Actual Sales vs Predicted Sales (after trend added back)

plt.figure(figsize=(14, 6))

plt.plot(test.index, test['Sales'], label='Actual Sales', alpha=0.6)

plt.plot(test.index, test['Final\_Predicted\_Sales'], label='Predicted Sales Using Linear Regression', color='orange', linestyle='--')

plt.title("Actual vs Predicted Sales")

plt.xlabel("Date")

plt.ylabel("Sales")

plt.legend()

plt.xticks(rotation=45)

plt.grid(True)

plt.tight\_layout()

plt.show()

# OUTPUT:

# exp4

**RESULT:**

Thus, the program to apply moving average smoothing for data preparation and time series forecasting has been done successfully.