EXPERIMENT 3

AIM:

To develop a linear regression model for time series forecasting using historical data and visualize actual vs. predicted values.

PROCEDURE:

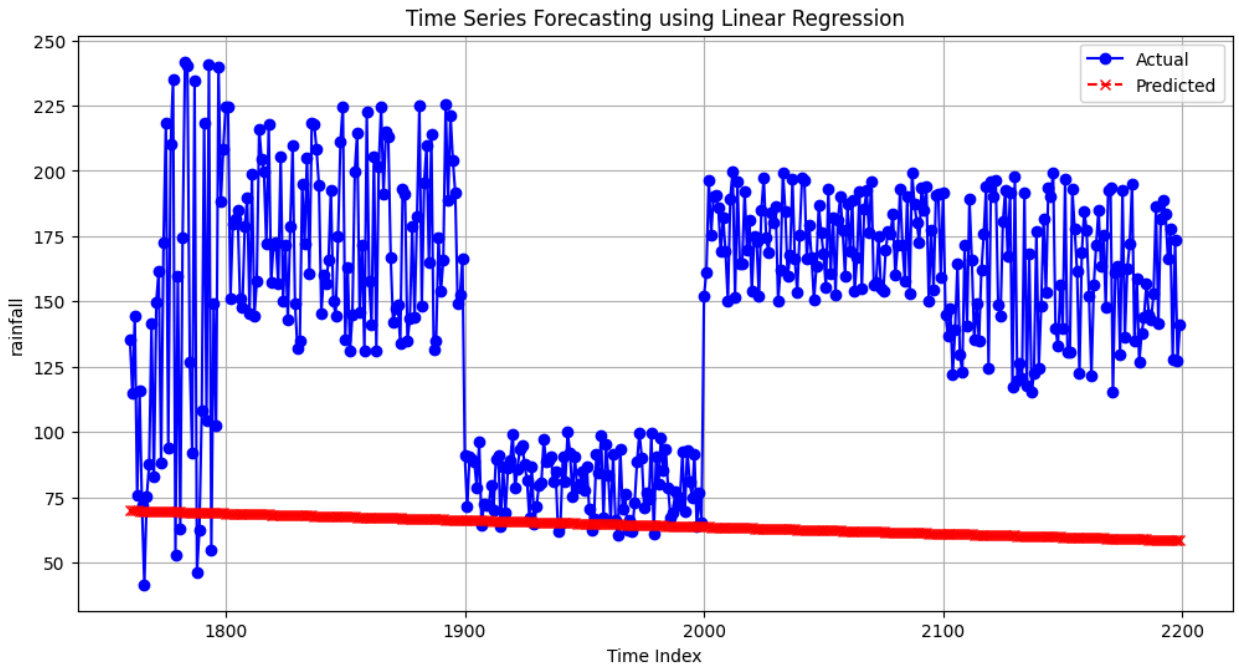
1. Upload the dataset and load it into a pandas DataFrame.
2. Check for a 'Date' column; if missing, create a time index.
3. Select a numerical target variable for prediction.
4. Split the dataset into training and testing sets.
5. Train a linear regression model and generate predictions.
6. Visualize actual vs. predicted values and evaluate model performance using MSE.

CODE:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from google.colab import files

# Upload file
uploaded = files.upload()

# Load the dataset
file_path = list(uploaded.keys())[0]
df = pd.read_csv(file_path)

# Display basic info about dataset
print(df.info())
print(df.head())

# Automatically create a time index
if 'Date' in df.columns:
    df['Date'] = pd.to_datetime(df['Date'])
    df = df.sort_values(by='Date')
    df.set_index('Date', inplace=True)
else:
    df['Time_Index'] = np.arange(len(df))
```

```
    df.set_index('Time_Index', inplace=True)

# Select target column (choosing a numerical column for regression)
numeric_cols = df.select_dtypes(include=[np.number]).columns
if len(numeric_cols) == 0:
    raise ValueError("No numerical column found for regression.")

target_column = numeric_cols[-1]  # Choosing last numerical column as target

# Splitting dataset into features (X) and target (y)
X = np.arange(len(df)).reshape(-1, 1)  # Using index as a proxy for time
y = df[target_column].values

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=False)

# Train Linear Regression model
model = LinearRegression()
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Visualization
plt.figure(figsize=(12, 6))
plt.plot(df.index[len(X_train):], y_test, label='Actual', marker='o', color='blue')
plt.plot(df.index[len(X_train):], y_pred, label='Predicted', linestyle='dashed', marker='x',
color='red')
plt.xlabel('Time Index' if 'Date' not in df.columns else 'Date')
plt.ylabel(target_column)
plt.title('Time Series Forecasting using Linear Regression')
plt.legend()
plt.grid()
plt.show()

# Model performance
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
```

OUTPUT:



RESULT:

The model predicts future values using linear regression, with a visualization showing actual vs. predicted trends. The Mean Squared Error (MSE) indicates prediction accuracy.