EXP No:4 Checking Stationarity of Time Series Data

Aim:

To analyze, visualize, and forecast electricity production using time series techniques and regression modeling.

Objective:

Analyze the stationarity of electricity production data using the Augmented Dickey-Fuller (ADF) test. Visualize trends, seasonality, and variability in the data.

Background:

1. **Time Series Analysis:** The analysis focuses on electricity production data and applies time series techniques like the Augmented Dickey-Fuller (ADF) test to check for stationarity.
2. **Data Visualization:** Data visualization methods, such as plotting trends and seasonality, are used to explore the dataset's characteristics.
3. **Feature Engineering:** Feature engineering creates time-based numeric features to capture trends in electricity production.
4. **Predictive Modeling:** A linear regression model is applied to forecast future electricity production values.
5. **Performance Evaluation:** Model performance is evaluated using metrics like Mean Squared Error (MSE) to assess prediction accuracy.

Code:

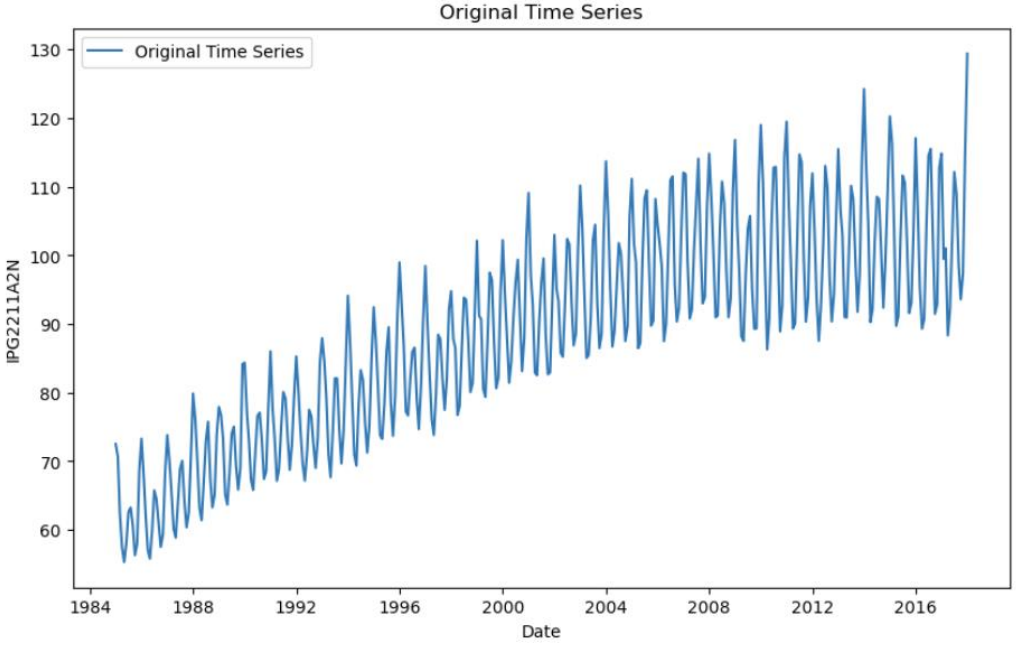
```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from statsmodels.tsa.stattools import adfuller



# Load the dataset (update this with your file path)

df = pd.read_csv(r"C:\Users\Lenovo\Downloads\exp4\Electric_Production.csv")



# Display the first few rows to understand the structure

print(df.head())
```

```
# Check the format of the 'DATE' column
print(df['DATE'].head())  # Inspect the date values

# Try converting the 'DATE' column to datetime
df['DATE'] = pd.to_datetime(df['DATE'], errors='coerce')  # Automatically handle errors

# Check for missing or invalid date values after conversion
print(df['DATE'].isnull().sum())  # Check for any missing or invalid dates

# Drop any rows with invalid 'DATE' values (if any)
df = df.dropna(subset=['DATE'])

# Set 'DATE' as the index
df.set_index('DATE', inplace=True)

# Function to perform the Augmented Dickey-Fuller test
def adf_test(series):
    result = adfuller(series)
    return result[1]  # p-value

# Plot the original time series
plt.figure(figsize=(10, 6))
plt.plot(df.index, df['IPG2211A2N'], label='Original Time Series')
plt.title('Original Time Series')
```

```
plt.xlabel('Date')
plt.ylabel('IPG2211A2N')
plt.legend()
plt.show()

# Perform ADF test on the original series
p_value_before = adf_test(df['IPG2211A2N'])
print(f"ADF Test p-value before transformation: {p_value_before}")

# Difference the series to make it stationary
df_diff = df['IPG2211A2N'].diff().dropna()

# Plot the differenced time series
plt.figure(figsize=(10, 6))
plt.plot(df_diff.index, df_diff, label='Differenced Time Series', color='orange')
plt.title('Differenced Time Series')
plt.xlabel('Date')
plt.ylabel('Differenced Value')
plt.legend()
plt.show()

# Perform ADF test on differenced series
p_value_after = adf_test(df_diff)
print(f"ADF Test p-value after transformation: {p_value_after}")
```

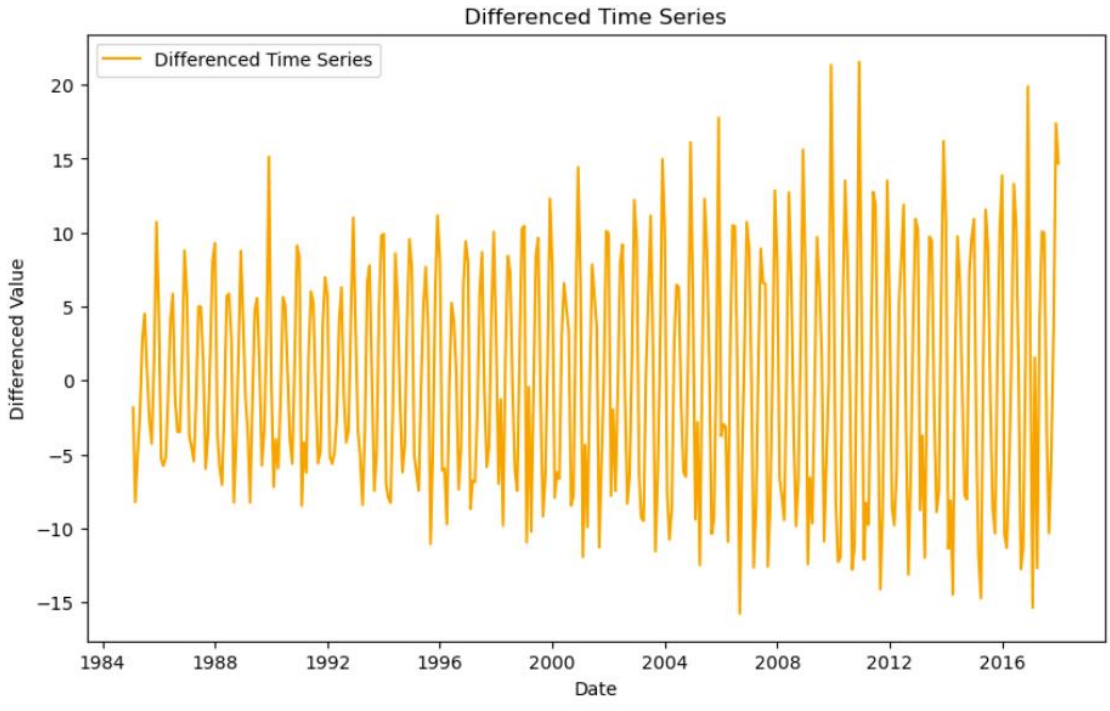
```
# Interpretation of p-values
if p_value_before < 0.05:
    print("The original time series is stationary.")
else:
    print("The original time series is non-stationary.")

if p_value_after < 0.05:
    print("The transformed time series is stationary.")
else:
    print("The transformed time series is still non-stationary.")
```

Output:



The original time series is non-stationary.

ADF Test p-value before transformation: 0.18621469116586814



ADF Test p-value after transformation: 4.0777865655383114e-10

The transformed time series is stationary.

Result:

Thus the time series program to check data stationarity using time series dataset has been implemented successfully.