**EX:No.2 221501026**

**31/01/25**

**IMPLEMENTING DIFFERENT VISUALIZATION TECHNIQUE USING TIME SERIES DATA**

**AIM:**

To implementing different visualization technique using time series dataset.

**PROCESS:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**# Load the stock data**

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 (assuming this is the relevant data)**

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)

**# 1. Handling Missing Values:**

**# Check for missing values in each column**

print("Missing values in each column:")

print(data.isnull().sum())

**# Fill missing values in 'IPG2211A2N' with the mean if there are any NaNs**

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

**# 2. Handling Outliers:**

**# Using IQR (Interquartile Range) to handle outliers 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)]

**# Normalize the 'IPG2211A2N' data for use in further analysis**

data\_values = production\_data\_reverse.values.reshape(-1, 1) # Reshape for normalization

data\_normalized = data\_values / np.max(data\_values) # Normalize the data

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

train\_size = int(len(data\_normalized) \* 0.8)

train\_data = data\_normalized[:train\_size]

test\_data = data\_normalized[train\_size:]

**# Visualization:**

**# 1. Box plot to check for outliers and distribution**

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

sns.boxplot(data['IPG2211A2N'])

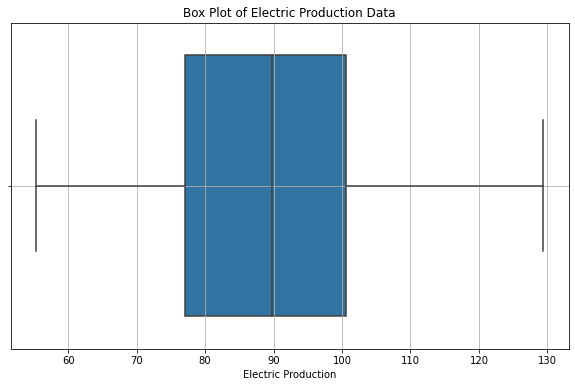
plt.title('Box Plot of Electric Production Data')

plt.xlabel('Electric Production')

plt.grid(True)

plt.show()

**OUTPUT:**



**# 2. Scatter plot to visualize the relationship between time and production**

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

plt.scatter(range(len(production\_data\_reverse)), production\_data\_reverse, color='blue', alpha=0.5)

plt.title('Scatter Plot of Electric Production Data')

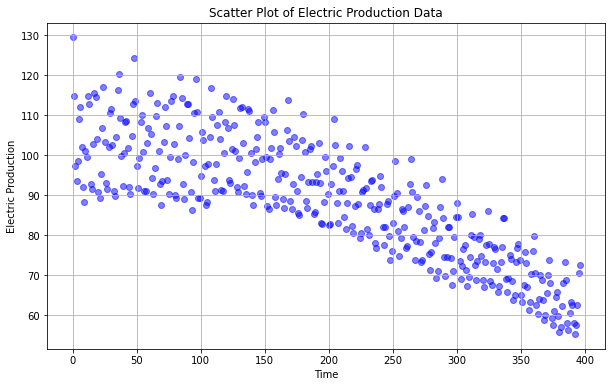
plt.xlabel('Time')

plt.ylabel('Electric Production')

plt.grid(True)

plt.show()

**OUTPUT:**



**# 3. Line plot (Already present) to show the time-series trend**

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

plt.plot(production\_data\_reverse)

plt.title('Electric Production Data (Line Plot)')

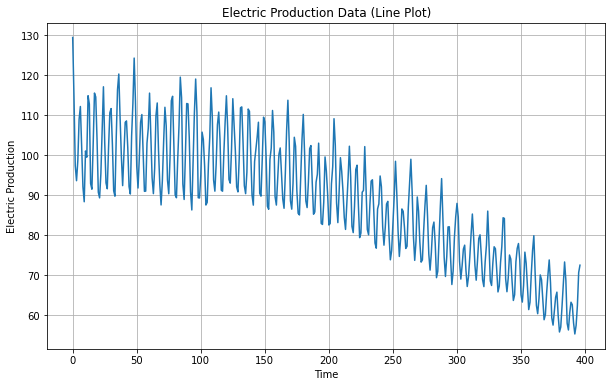
plt.xlabel('Time')

plt.ylabel('Electric Production')

plt.grid(True)

plt.show()

**OUTPUT:**



#**# 4. Bar chart to show the time-series trend**

import matplotlib.pyplot as plt

# Sample data (replace with your actual bank data)

bank\_names = ['Bank A', 'Bank B', 'Bank C']

account\_balances = [10000, 15000, 20000] # Example: Account balances

# Create the bar chart

plt.bar(bank\_names, account\_balances)

# Add labels and title

plt.xlabel('Bank Name')

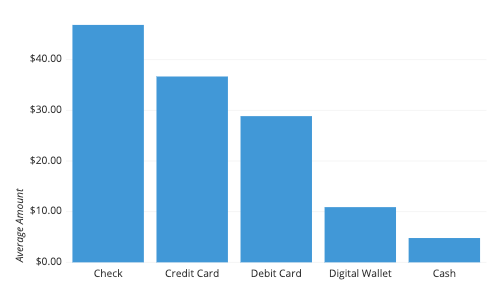
plt.ylabel('Account Balance')

plt.title('Bank Account Balances')

# Show the chart

plt.show()

**OUTPUT:**



**RESULT:**

The implementing different visualization technique using time series dataset is successfully implemented.