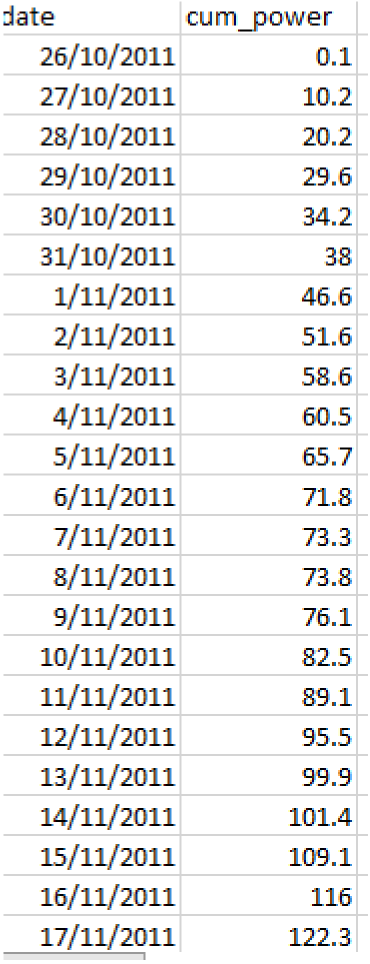
Topic: Forecasting – Time Series

Name:ULLI VENKATA SAI KUMAR
Batch ID: 04072024HYD10AM
Topic: Forecasting – Time Series

Problem Statement: -

1. Solar power consumption has been recorded by city councils at regular intervals. The reason behind doing so is to understand how businesses are using solar power so that they can cut down on nonrenewable sources of energy and shift towards renewable energy. Based on the data, build a forecasting model, and provide insights on it.

Solarpower.csv

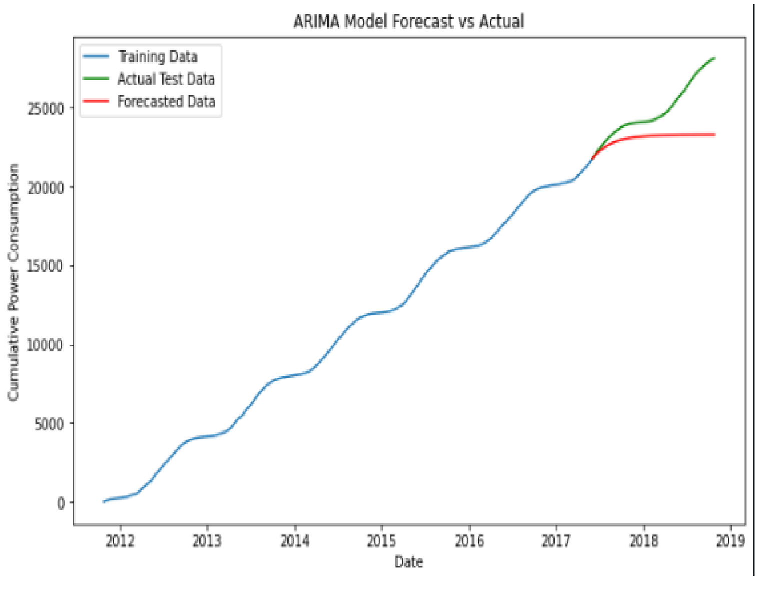


date	cum_power
26/10/2011	0.1
27/10/2011	10.2
28/10/2011	20.2
29/10/2011	29.6
30/10/2011	34.2
31/10/2011	38
1/11/2011	46.6
2/11/2011	51.6
3/11/2011	58.6
4/11/2011	60.5
5/11/2011	65.7
6/11/2011	71.8
7/11/2011	73.3
8/11/2011	73.8
9/11/2011	76.1
10/11/2011	82.5
11/11/2011	89.1
12/11/2011	95.5
13/11/2011	99.9
14/11/2011	101.4
15/11/2011	109.1
16/11/2011	116
17/11/2011	122.3

```
# Let's start by loading and inspecting the uploaded CSV file to understand its structure.
import pandas as pd
# Load the CSV file
solar_data = pd.read_csv('solarpower_cumuldaybyday2.csv')
# Display the first few rows of the data to understand its structure
solar_data.head()
# Convert the 'date' column to datetime format and check for missing values
```

```
solar_data['date'] = pd.to_datetime(solar_data['date'], format='%d/%m/%Y')
# Check for missing values
missing_values = solar_data.isnull().sum()
# Display basic statistics for the 'cum_power' column
statistics = solar_data['cum_power'].describe()
missing_values, statistics
# Set the 'date' column as the index for time series analysis
solar_data.set_index('date', inplace=True)
# Split the data into training and test sets (80% training, 20% testing)
train_size = int(len(solar_data) * 0.8)
train_data = solar_data[:train_size]
test_data = solar_data[train_size:]
# Import necessary libraries for time series forecasting
from statsmodels.tsa.arima.model import ARIMA
import matplotlib.pyplot as plt
# Fit an ARIMA model on the training data
model = ARIMA(train_data['cum_power'], order=(5, 1, 0))
arima_model = model.fit()
# Generate predictions for the test data
forecast = arima_model.forecast(steps=len(test_data))
# Plot the actual vs forecasted values
plt.figure(figsize=(10,6))
plt.plot(train_data.index, train_data['cum_power'], label='Training Data')
plt.plot(test_data.index, test_data['cum_power'], label='Actual Test Data', color='green')
plt.plot(test_data.index, forecast, label='Forecasted Data', color='red')
plt.xlabel('Date')
plt.ylabel('Cumulative Power Consumption')
plt.title('ARIMA Model Forecast vs Actual')
plt.legend()
plt.show()
```

Output:-


ARIMA Model Forecast vs Actual
Training Data
Actual Test Data
Forecasted Data
Cumulative Power Consumption
Date