University Institute of Engineering

Department of Electronics & Communication Engineering

Experiment No. :- 8

Student Name: Priyanshu Mathur **UID:** 20BEC1073
Branch: Electronics and Communication **Section/Group:** A
Semester: 7th **Date of Performance:** 30/10/23
Subject Name: AIML **Subject Code:** 20ECA-445

1. Aim of the practical: Write a program for time series analysis using AI and ML.

2. Tool Used: Google Colab

3. Theory: Time series analysis is a specific way of analyzing a sequence of data points collected over an interval of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points intermittently or randomly. However, this type of analysis is not merely the act of collecting data over time. What sets time series data apart from other data is that the analysis can show how variables change over time. In other words, time is a crucial variable because it shows how the data adjusts over the course of the data points as well as the final results. It provides an additional source of information and a set order of dependencies between the data. Time series analysis typically requires a large number of data points to ensure consistency and reliability. An extensive data set ensures you have a representative sample size and that analysis can cut through noisy data. It also ensures that any trends or patterns discovered are not outliers and can account for seasonal variance. Additionally, time series data can be used for forecasting—predicting future data based on historical data.

4. Steps for experiment/practical:

Step 1: - Open Google Colab

Step 2: - Create a new notebook

Step 3: - Write the code given below and run it.

5. Program Code and Simulation

Output:Code:-

```
import numpy as np
import pandas as pd
import statsmodels
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from math import sqrt
df = pd.read_csv('/content/drive/MyDrive/Copy of Electric_Production.csv')
df.rename(columns={'DATE':'date','IPG2211A2N':'production'},inplace=True)
df['date'] = pd.to_datetime(df['date'])
sns.lineplot(data=df,x='date',y='production')
# Decomposing the data for better view of the data
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(x=df['production'],model='additive',period=12)
decomposition.plot()
plt.show()
#Augumented Dicky Fuller(ADF) test for stationarity
from statsmodels.tsa.stattools import adfuller
dftest = adfuller(df['production'],autolag='AIC')
adf,p_value,_,_,_,_ = adfuller(df['production'])
print(f'ADF: {round(adf,2)}')
print(f'p_value: {round(p_value,2)}')
threshold = 0.05
```

```
if p_value<threshold:
    print('The data is stationary')
else:
    print('The data is not stationary')
```

#Checking the data visually with lags

```
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
plot_acf(df['production'],lags=40)
plt.show()
#Converting non stationay data into stationay data using $Differencing$
```

#Trend removal

```
diff = df['production']-df['production'].shift(1)
diff = diff.dropna(inplace = False)
```

#ACF after trend removal

```
fig = plt.figure(figsize=(20, 10))
subplot1 = fig.add_subplot(211)
subplot2 = fig.add_subplot(212)
sns.lineplot(x=df['date'],y=diff,ax=subplot1)
plot_acf(diff,lags=40,ax=subplot2)
plt.show()
```

#Seasonality Removal

```
diff = df['production']-df['production'].shift(1)
diff_season = diff - diff.shift(12)
diff_season = diff_season.dropna(inplace=False)
```

#PACF after Seasonality removal

```
fig = plt.figure(figsize=(20, 10))
subplot1 = fig.add_subplot(211)
```

```
subplot2 = fig.add_subplot(212)
sns.lineplot(x=df['date'],y=diff_season,ax=subplot1)
plot_pacf(diff_season,lags=40,ax=subplot2)
plt.show()
```

#Augumented Dicky Fuller(ADF) test for stationarity after data tranformation

```
dftest = adfuller(diff_season,autolag='AIC')
adf,p_value,_,_,_,_ = adfuller(diff_season)
print(f'ADF: {round(adf,2)}')
print(f'p_value: {round(p_value,2)}')
threshold = 0.05
if p_value<threshold:
   print('The data is stationary')
else:
   print('The data is not stationary')
df_test = df[['date', 'production']].loc[300:]
df_train = df[['date', 'production']].loc[:299] from statsmodels.tsa.arima.model import ARIMA
ARIMA_model = ARIMA(df_train['production'],order=(3,1,4)).fit(method_kwargs={"warn_convergence":
False})
```

 # generate predictions

```
df_pred = ARIMA_model.predict(start=300, end=396)
```

 # plot actual vs. predicted

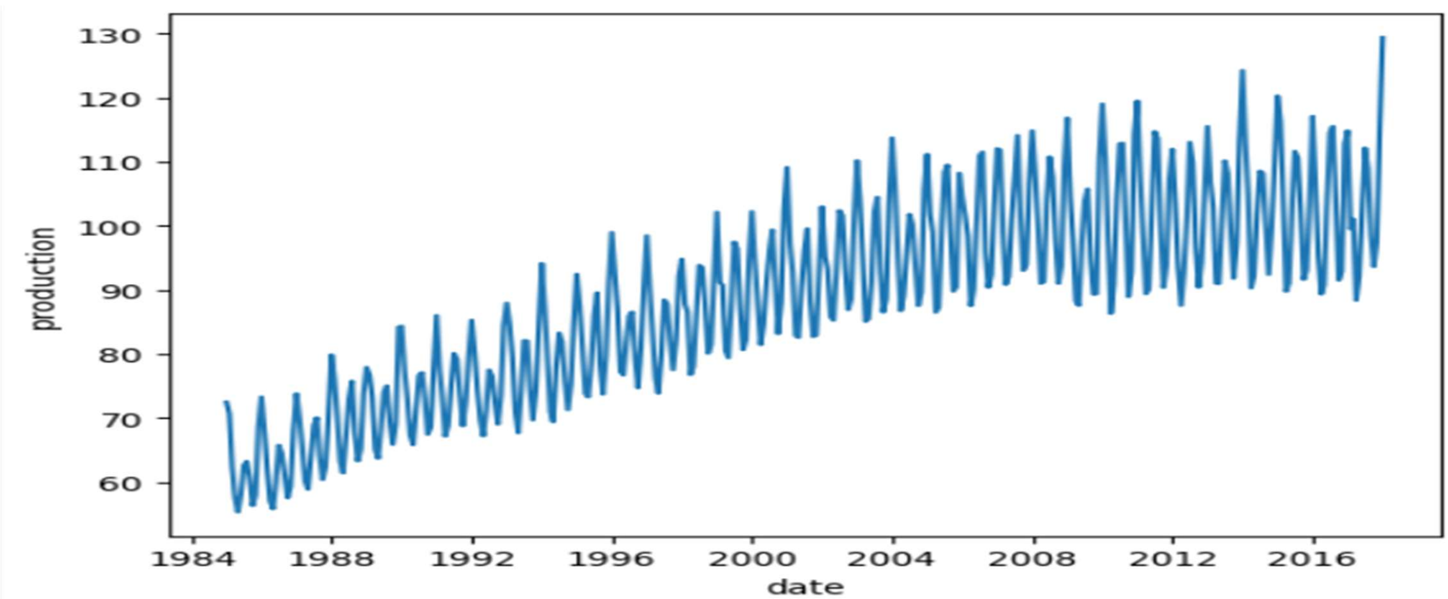
```
fig = plt.figure(figsize=(20, 10))
plt.title('ARIMA Predictions', fontsize=20)
plt.plot(df_test['production'], label='actual', color='#ABD1DC')
plt.plot(df_pred, label='predicted', color='#C6A477')
plt.legend(fontsize =20, loc='upper left')
```

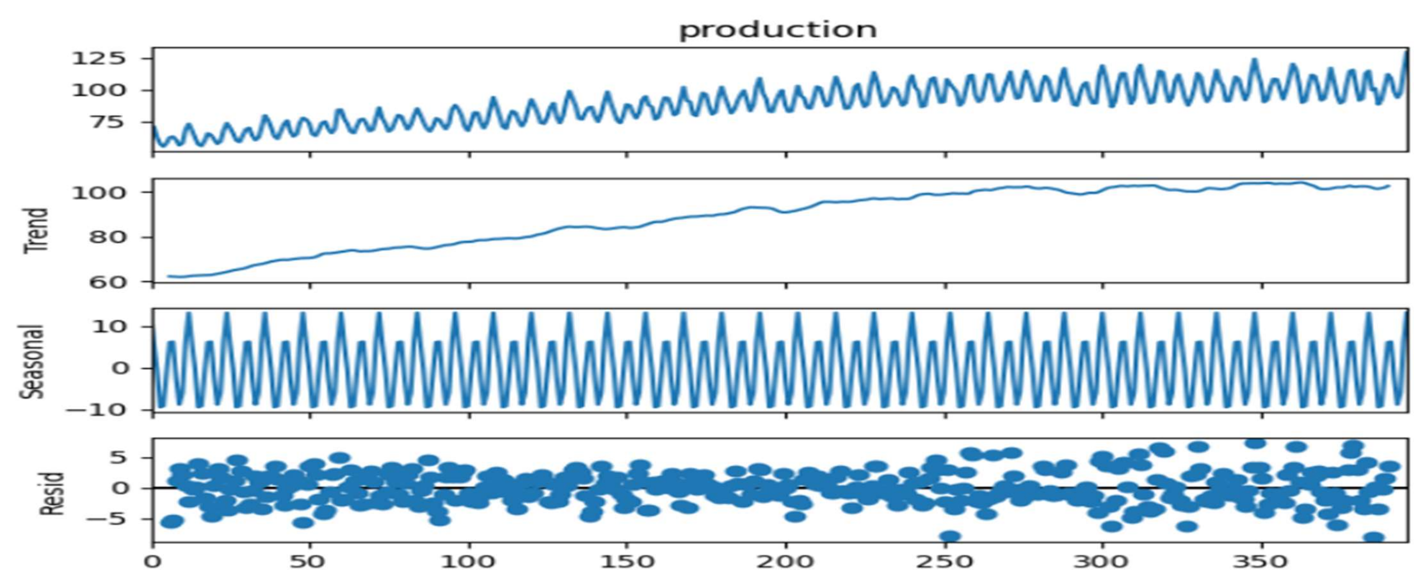
```
rmse = sqrt(mean_squared_error(df_test['production'], df_pred))

print("RMSE:", round(rmse,2))

print('Larger RMSE indicates more difference between actual and predicted values.')
```

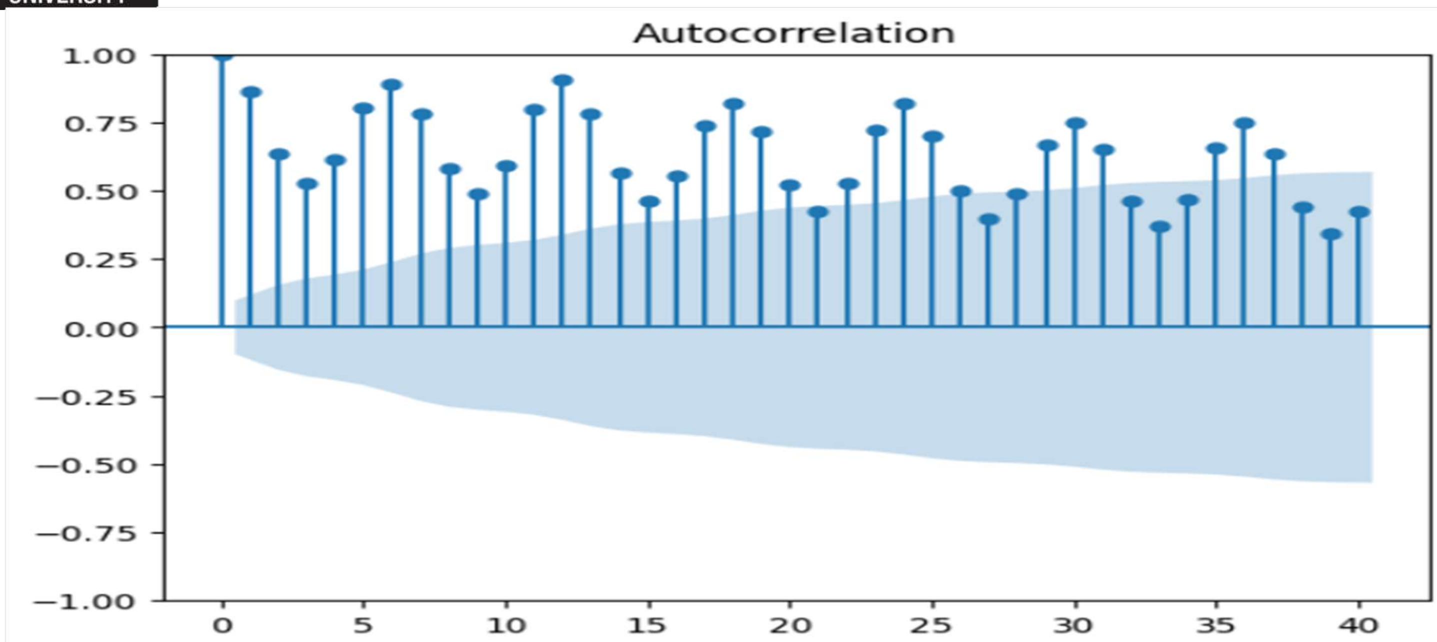
Plot

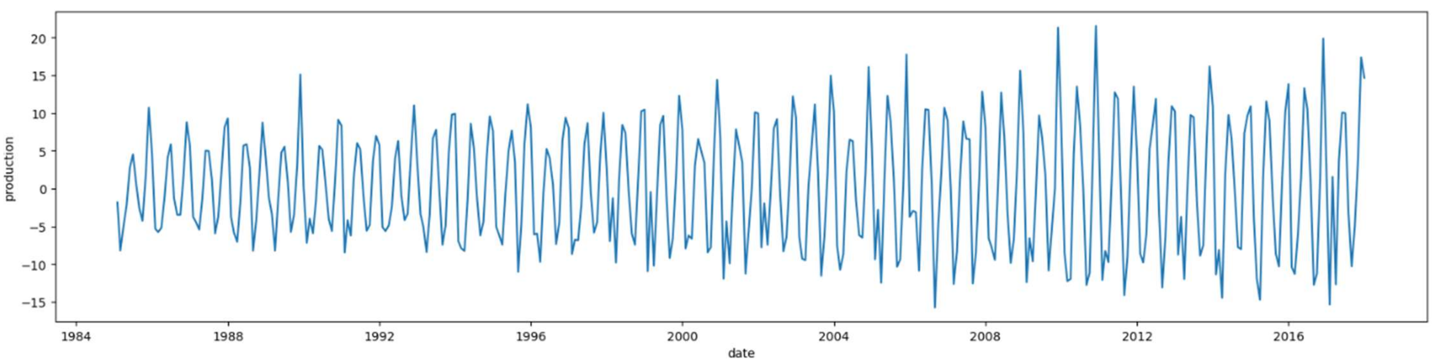


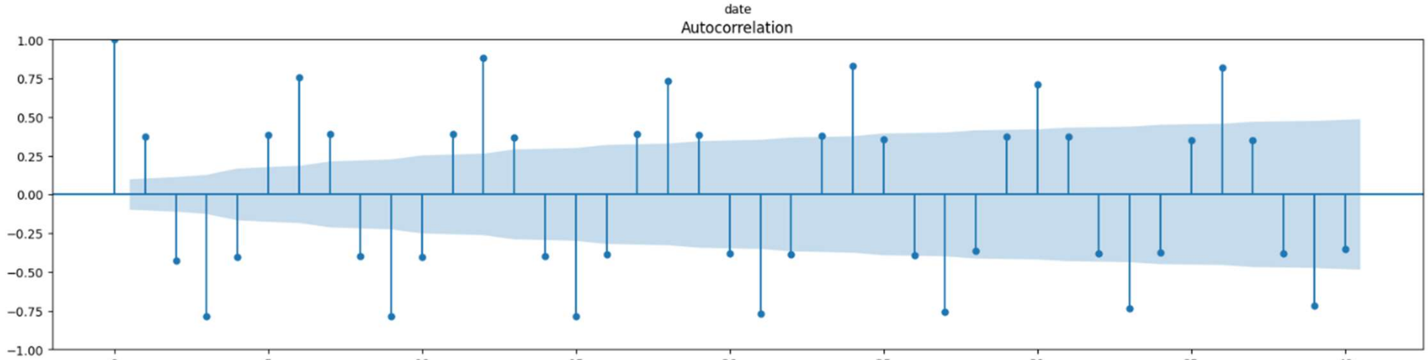


University Institute of Engineering

Department of Electronics & Communication Engineering

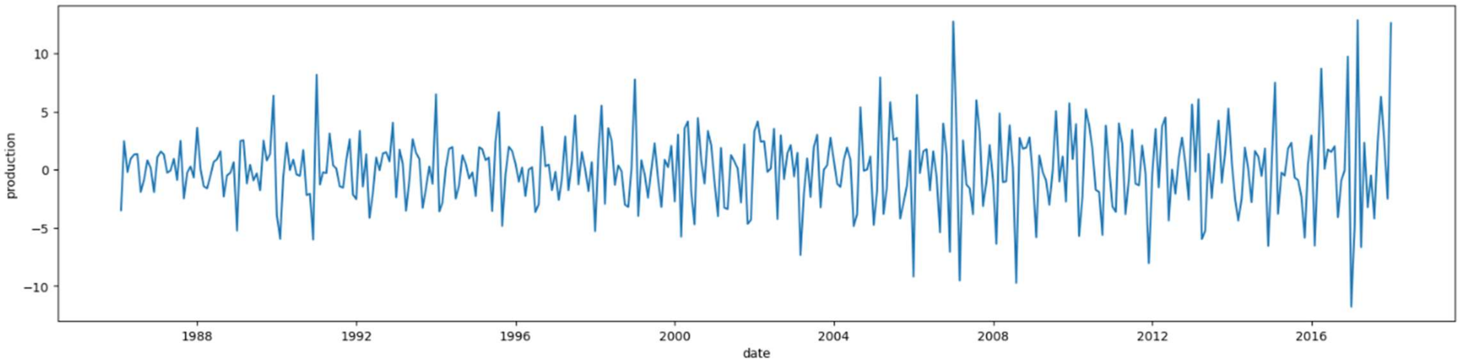


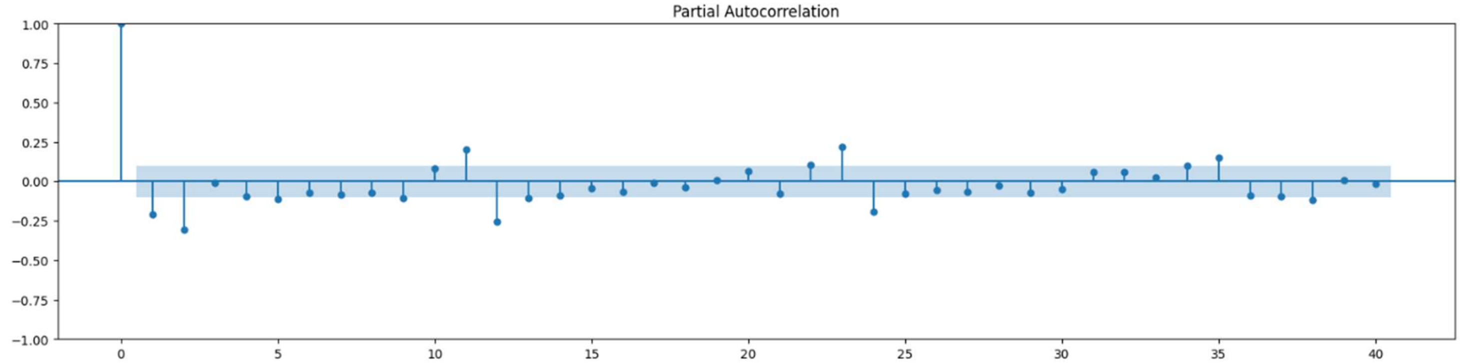


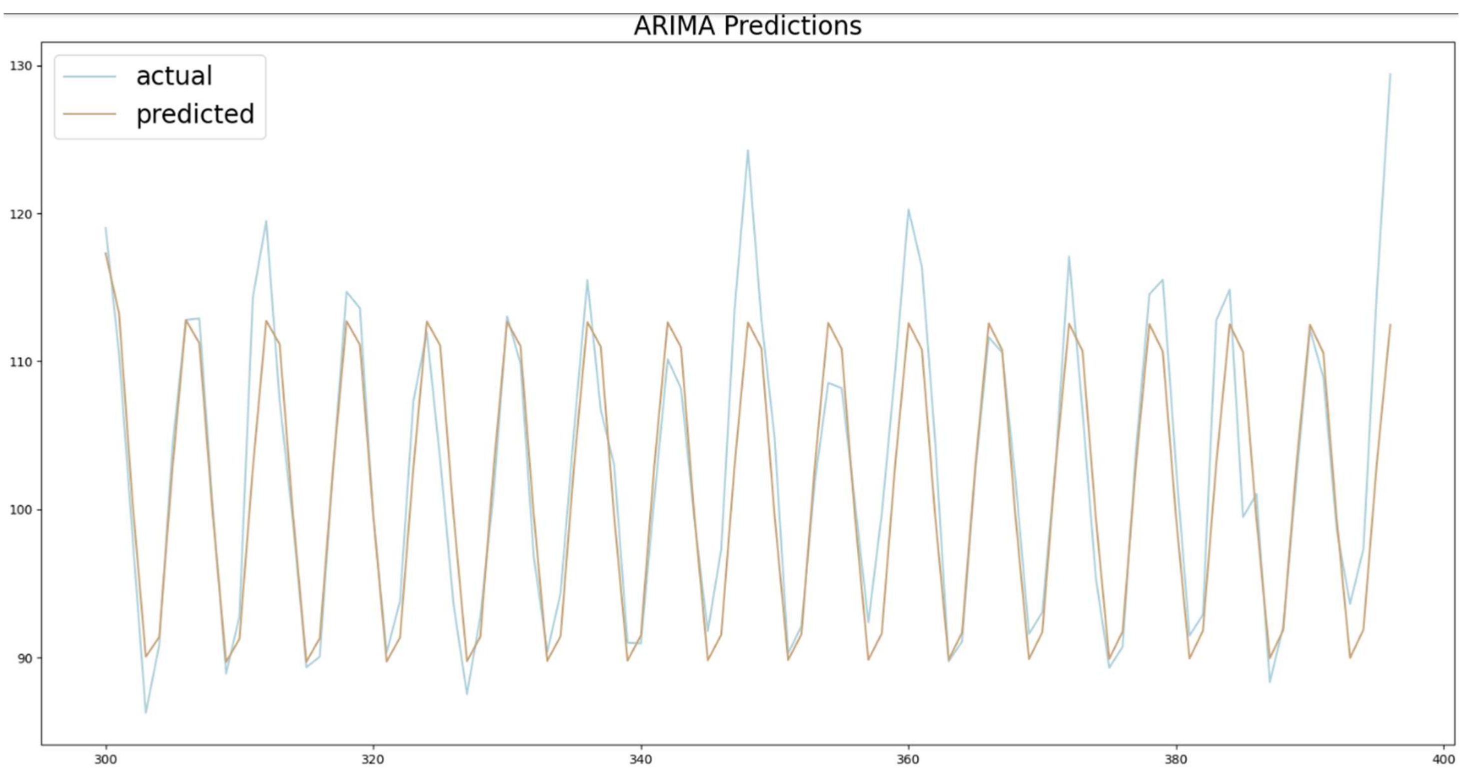


University Institute of Engineering

Department of Electronics & Communication Engineering







Result and Discussion: - In this experiment we use time series analysis to predict the future trend using past trends. We have use ARIM (Auto Regressive Moving Average) model to predict the same. We have used autocorrelation and partial autocorrelation. For stationary test we have use ADF (Augumented Dicky Fuller) test.

Learning outcomes (What I have learnt):

- Learnt about Time Series Analysis.
- Learn about trend analysis, seasonality analysis, and stationary test and outliers detections.
- Learn about MA, ARIMA, and SARIMA.