Smt. Indira Gandhi College of Engineering
Computer Engineering Department
Ghansoli – Navi Mumbai
Academic Year 2023-24 (Even Sem)

Student Name: Khyati Garude **Roll No.:** 13 **Class:** BE **Sem:** VIII

Course Name: Applied Data Science Lab

Course Code: CSL8023

Experiment No. 09

Experiment Title: Time Series Forecasting Using ARIMA Model

Date of Performance	Date of Submission	Marks (10)					Sign / Remark
		A	B	C	D	E	
		2	3	2	2	1	
21/3/24	4/4/24	2	3	2	2	1	12/4/24.
		Total Marks					
		10					

A: Prerequisite Knowledge
B: Implementation
C: Oral
D: Content
E: Punctuality & Discipline

Khyati Garude
BE COMPS
ADS

Date:

DATE	EXPERIMENT-9	SIGN
4/4/24	Time Series forecasting using ARIMA model	24/4/24

AIM: Time series forecasting using ARIMA model.

THEORY:

→ Auto Regressive Integrated Moving Average (ARIMA) is a widely used statistical method for modeling and forecasting time series data.

→ It combines three components : Auto Regressive (AR), Integrated (I), and Moving Average (MA).

(i) Auto Regressive (AR) Component:

→ The AR component models the relationship between an observation and a number of lagged observations (i.e. past values of the series).

→ The term 'autoregressive' indicates that the model uses the dependent relationship between an observation and some number of lagged observations.

Date:_____

(ii) Integrated (I) component:

→ The Integrated component accounts for the differencing operation applied to the time series data to make it stationary.

→ Differencing involves computing the difference between consecutive observations to remove trends or seasonality.

→ The 'd' parameter in ARIMA represents the order of differencing.

(iii) Moving Average (MA) component:

→ The MA component models the relationship between an observation and a residual error from a moving average model applied to lagged observations.

→ The moving average part represents the error of the model as a combination of error terms from previous time points.

→ Notation:

ARIMA models are denoted by the notation ARIMA(p, d, q), where:

- 'p' represents the order of the autoregressive component (AR).
- 'd' represents the order of differencing (I).
- 'q' represents the order of the moving average component (MA).

Date:_____

→ SARIMA:

Seasonal AutoRegressive Integrated Moving Average (SARIMA) extends the ARIMA model to handle seasonal time series data, which exhibit periodic patterns or seasonal fluctuations.

SARIMA includes additional parameters to model seasonal patterns in the data. In addition to the AR, I, and MA terms, SARIMA introduces seasonal AR, seasonal difference, and seasonal MA terms.

Notation:

SARIMA models are denoted by the notation SARIMA $(p,d,q)(P,D,Q,s)$, where:

- → 'p', 'd', 'q' are the same as in ARIMA.
- → 'P', 'D', 'Q' represent the seasonal order of the AR, difference, and MA terms, respectively.
- → 's' is the seasonal period, indicating the length of the seasonal pattern.

SARIMA can be used to forecast future values of the seasonal time series data.

It takes into account both the non-seasonal and seasonal components to generates accurate forecasts.

Similar to ARIMA, SARIMA model performance can be evaluated using metrics like MAE, MSE and RMSE.

Date:

ALGORITHM:

(i) **Data Preparation:**
Load the time series data from the CSV file and processes it, ensuring it is in the appropriate format for time series analysis.

(ii) **Stationarity Check:**
Plot the time series data and conduct a stationarity check using the Augmented Dickey - Fuller (ADF) test to determine if differencing is necessary

(iii) **Auto ARIMA model selection:**
Use the 'auto-arima' function from the 'pmdarima' library to automatically select the best SARIMA model parameters based on AIC criterion and trace the model selection process.

(iv) **Fit SARIMA model:**
Fit the SARIMA model to the training data using the selected model parameters obtained from the previous step.

(v) **Model summary:**
print the summary of the fitted SARIMA model, displaying the estimated parameters and diagnostic statistics.

Date:_____

(vi) Forecasting:

Generate predictions using the fitted SARIMA model for the test data period.

(vii) Plotting:

plot the SARIMA predictions along with the actual test data to visualize the forecasted values.

The main change while implementing ARIMA lies in:

(i) Model Identification:

- ARIMA identifies parameters (p, d, q) for the non-seasonal component only.
- SARIMA additionally identifies parameters (P, D, Q, s) for the seasonal component.

(ii) Model Fitting:

- ARIMA fits the non-seasonal ARIMA model.
- SARIMA fits both the SARIMA model's non-seasonal and seasonal components.

(iii) Forecasting:

- ARIMA predicts future values based soley on the non-seasonal component.
- SARIMA considers both non-seasonal and seasonal components for forecasting.

Date:_____

CONCLUSION:

By understanding the underlying components and principles of the ARIMA model, analysts and data scientists can effectively leverage it for time series forecasting tasks in various domains such as finance, economics and sales forecasting. SARIMA is particularly useful for time series data with clear seasonal patterns, such as quarterly sales data, monthly temperature data, or weekly website traffic data.

```
pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux201
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 7.3 MB/s eta 0:00:
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/di
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/loc
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/d
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/di
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/di
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/li
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pytho
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/di
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packag
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
```

```
import pandas as pd
df=pd.read_csv('/content/Sleep_Efficiency.csv',index_col='Bedtime' ,parse_date
df=df.dropna()
print('Shape of data',df.shape)
df.head()
df
```

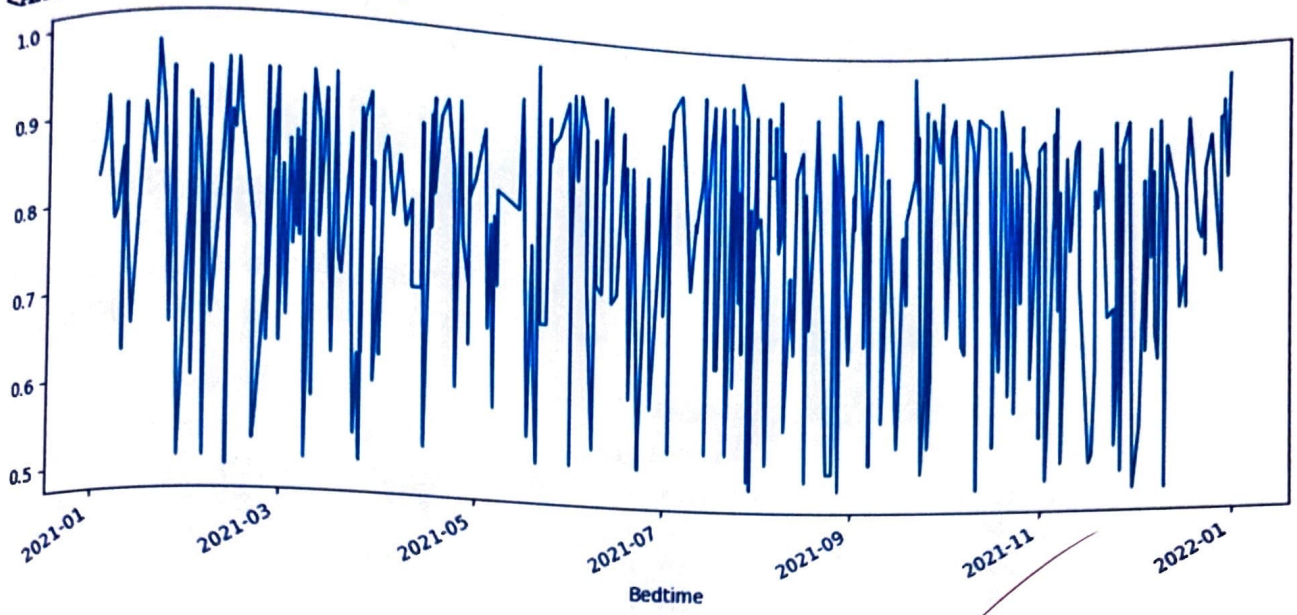
```
Shape of data (388, 14)
```

Bedtime	ID	Age	Gender	Wakeup time	Sleep duration	Sleep efficiency	REM sleep percentage	Deep sle percenta
2021-03-06 01:00:00	1	65	Female	2021-03-06 07:00:00	6.0	0.88	18	
2021-12-05 02:00:00	2	69	Male	2021-12-05 09:00:00	7.0	0.66	19	
2021-05-25 21:30:00	3	40	Female	2021-05-25 05:30:00	8.0	0.89	20	
2021-11-03 02:30:00	4	40	Female	2021-11-03 08:30:00	6.0	0.51	23	
2021-03-13 01:00:00	5	57	Male	2021-03-13 09:00:00	8.0	0.76	27	
...	...	...	...	...	...	...	...	
2021-11-16 23:00:00	446	30	Female	2021-11-16 06:30:00	7.5	0.53	28	
2021-11-13 22:00:00	448	27	Female	2021-11-13 05:30:00	7.5	0.91	22	
2021-03-31 21:00:00	449	52	Male	2021-03-31 03:00:00	6.0	0.74	28	
2021-07-29 21:00:00	451	45	Male	2021-07-29 04:00:00	7.0	0.76	18	
2021-03-17 02:30:00	452	18	Male	2021-03-17 10:00:00	7.5	0.63	22	

388 rows × 14 columns

```
df['Sleep efficiency'].plot(figsize=(12,5))
```

```
<Axes: xlabel='Bedtime'>
```




```
from statsmodels.tsa.stattools import adfuller
def ad_test(dataset):
    dftest = adfuller(dataset, autolag = 'AIC')
    print("1. ADF : ",dftest[0])
    print("2. P-Value : ", dftest[1])
    print("3. Num Of Lags : ", dftest[2])
    print("4. Num Of Observations Used For ADF Regression:",      dftest[3])
    print("5. Critical Values :")
    for key, val in dftest[4].items():
        print("\t",key, ": ", val)
ad_test(df['Sleep efficiency'])
```

```
1. ADF :  -12.816564788490636
2. P-Value :  6.282000661013602e-24
3. Num Of Lags :  1
4. Num Of Observations Used For ADF Regression: 386
5. Critical Values :
        1% :  -3.447405233596701
        5% :  -2.8690569369014605
        10% :  -2.5707743450830893
```

```
from pmdarima import auto_arima

# Fit SARIMA model using auto_arima
stepwise_fit = auto_arima(df['Sleep efficiency'], seasonal=True, trace=True, St

# Get the best SARIMA model
sarima_model = stepwise_fit.to_dict()['order']
sarima_seasonal_order = stepwise_fit.to_dict()['seasonal_order']

print("Best SARIMA model order:", sarima_model)
print("Best SARIMA seasonal order:", sarima_seasonal_order)
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(0,0,0)[0] intercept   : AIC=-448.491, Time=2.92 sec
 ARIMA(0,0,0)(0,0,0)[0] intercept   : AIC=-445.780, Time=0.15 sec
 ARIMA(1,0,0)(0,0,0)[0] intercept   : AIC=-447.435, Time=0.23 sec
 ARIMA(0,0,1)(0,0,0)[0] intercept   : AIC=-446.696, Time=0.25 sec
 ARIMA(0,0,0)(0,0,0)[0]             : AIC=930.717, Time=0.08 sec
 ARIMA(1,0,2)(0,0,0)[0] intercept   : AIC=-450.312, Time=1.93 sec
 ARIMA(0,0,2)(0,0,0)[0] intercept   : AIC=-451.270, Time=0.70 sec
 ARIMA(0,0,3)(0,0,0)[0] intercept   : AIC=-450.151, Time=0.61 sec
 ARIMA(1,0,1)(0,0,0)[0] intercept   : AIC=-449.274, Time=1.90 sec
 ARIMA(1,0,3)(0,0,0)[0] intercept   : AIC=-448.350, Time=3.83 sec
 ARIMA(0,0,2)(0,0,0)[0]             : AIC=352.412, Time=0.35 sec

Best model:  ARIMA(0,0,2)(0,0,0)[0] intercept
Total fit time: 12.995 seconds
Best SARIMA model order: (0, 0, 2)
Best SARIMA seasonal order: (0, 0, 0, 0)
```

```
from pmdarima import auto_arima

# Fit SARIMA model using auto_arima
stepwise_fit = auto_arima(train['Sleep efficiency'], seasonal=True, m=20, trace

# Get the best SARIMA model order parameters
best_order = stepwise_fit.order
best_seasonal_order = stepwise_fit.seasonal_order

print("Best SARIMA model order parameters (p, d, q):", best_order)
print("Best SARIMA seasonal order parameters (P, D, Q, s):", best_seasonal_orde
```

```
Performing stepwise search to minimize aic
 ARIMA(2,0,2)(1,0,1)[20] intercept   : AIC=inf, Time=7.40 sec
 ARIMA(0,0,0)(0,0,0)[20] intercept   : AIC=-393.959, Time=0.17 sec
 ARIMA(1,0,0)(1,0,0)[20] intercept   : AIC=-394.698, Time=5.17 sec
 ARIMA(0,0,1)(0,0,1)[20] intercept   : AIC=-393.833, Time=0.72 sec
 ARIMA(0,0,0)(0,0,0)[20]             : AIC=836.728, Time=0.04 sec
 ARIMA(1,0,0)(0,0,0)[20] intercept   : AIC=-396.516, Time=0.10 sec
 ARIMA(1,0,0)(0,0,1)[20] intercept   : AIC=-394.744, Time=1.07 sec
 ARIMA(1,0,0)(1,0,1)[20] intercept   : AIC=inf, Time=2.26 sec
 ARIMA(2,0,0)(0,0,0)[20] intercept   : AIC=-398.965, Time=0.14 sec
 ARIMA(2,0,0)(1,0,0)[20] intercept   : AIC=-393.773, Time=3.51 sec
 ARIMA(2,0,0)(0,0,1)[20] intercept   : AIC=-397.109, Time=2.04 sec
 ARIMA(2,0,0)(1,0,1)[20] intercept   : AIC=inf, Time=3.64 sec
 ARIMA(3,0,0)(0,0,0)[20] intercept   : AIC=-397.312, Time=0.39 sec
 ARIMA(2,0,1)(0,0,0)[20] intercept   : AIC=-397.309, Time=0.47 sec
 ARIMA(1,0,1)(0,0,0)[20] intercept   : AIC=-397.553, Time=0.54 sec
 ARIMA(3,0,1)(0,0,0)[20] intercept   : AIC=-395.344, Time=1.18 sec
 ARIMA(2,0,0)(0,0,0)[20]             : AIC=inf, Time=0.09 sec

Best model:  ARIMA(2,0,0)(0,0,0)[20] intercept
Total fit time: 29.022 seconds
Best SARIMA model order parameters (p, d, q): (2, 0, 0)
Best SARIMA seasonal order parameters (P, D, Q, s): (0, 0, 0, 20)
```

```
print(df.shape)
train=df.iloc[:-40]
test=df.iloc[-40:]
print(train.shape,test.shape)
```

```
(388, 14)
(348, 14) (40, 14)
```

```
from statsmodels.tsa.statespace.sarimax import SARIMAX

# Define SARIMA model parameters
order = best_order  # Non-seasonal order (p, d, q)
seasonal_order = best_seasonal_order # Seasonal order (P, D, Q, s), assuming a

# Fit SARIMA model
sarima_model = SARIMAX(train['Sleep efficiency'], order=order, seasonal_order=s
sarima_model_fit = sarima_model.fit()

# Print SARIMA model summary
print(sarima_model_fit.summary())
```

SARIMAX Results

Dep. Variable:	Sleep efficiency	No. Observations:	
Model:	SARIMAX(2, 0, 0)	Log Likelihood	138.
Date:	Tue, 09 Apr 2024	AIC	-270.
Time:	12:38:54	BIC	-259.
Sample:	0	HQIC	-266.
	- 348		
Covariance Type:	opg		

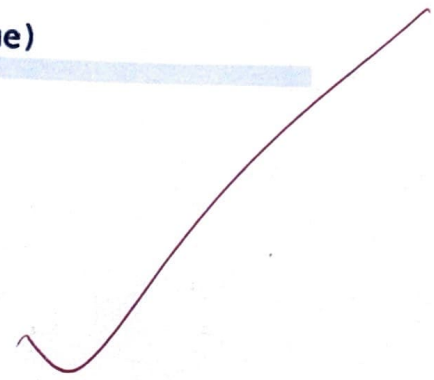
	coef	std err	z	P>|z|	[0.025	0.9
ar.L1	0.3869	0.047	8.197	0.000	0.294	0.
ar.L2	0.6007	0.047	12.673	0.000	0.508	0.
sigma2	0.0262	0.002	12.080	0.000	0.022	0.

Ljung-Box (L1) (Q):	14.41	Jarque-Bera (JB):	
Prob(Q):	0.00	Prob(JB):	
Heteroskedasticity (H):	1.03	Skew:	
Prob(H) (two-sided):	0.87	Kurtosis:	

```
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comp
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
```

```
# Specify start and end points for predictions
start = len(train)  # Start index for predictions
```

```
end = len(train) + len(test) - 1  # End index for predictions

sarima_model = SARIMAX(train['Sleep efficiency'] , order = best_order, seasonal
sarima_model_fit = sarima_model.fit()

# Generate SARIMA predictions
sarima_pred = sarima_model_fit.predict(start=start, end=end, typ='levels').rena

# Set index of predictions to match the dates in the test dataset
sarima_pred.index = test.index

#print(sarima_pred)

# Plot SARIMA predictions and test data
sarima_pred.plot(legend=True)
test['Sleep efficiency'].plot(legend=True)
```



12/4/24.

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:4
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:8
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:8
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/statespace/represen
  warnings.warn(msg, FutureWarning)
<Axes: xlabel='Bedtime'>
```