PRACTICAL 1.

Time Series Analysis for Capacity Utilization: Total Index

1. **DATA INTRODUCTION: -**

   The Federal Reserve Board constructs estimate of capacity and capacity utilization for industries in manufacturing, mining, and electric and gas utilities. For a given industry, the capacity utilization rate is equal to an output index (seasonally adjusted) divided by a capacity index.

   The Federal Reserve's monthly index of industrial production and the related capacity indexes and capacity utilization rates cover manufacturing, mining, and electric and gas utilities. The industrial sector, together with construction, accounts for the bulk of the variation in national output over the course of the business cycle.

   This dataset contains monthly time series data spanning from January 1989 to September 2024. and it contains two columns "date" and "capacity utilization" with 429 rows.

   Source: - https://fred.stlouisfed.org/series/TCU

   Board of Governors of the Federal Reserve System (US)

   Release: G.17 Industrial Production and Capacity Utilization

   Units: Percent, Seasonally Adjusted

   Frequency: Monthly

2. **UPLOAD THE CONVERT THE DATA INTO TIME SERIES DATA**

   R Code: -

```
rm(list=ls())
install.packages("forecast")
library(forecast)
data=read.csv("C:/Users/sonu/Desktop/Capacity_Utilization.csv")
data
```

Output: -

```
> data
        DATE Capacity.Utilization
1   01-01-1989            85.1871
2   01-02-1989            84.6881
3   01-03-1989            84.7698
4   01-04-1989            84.5900
5   01-05-1989            83.9611
6   01-06-1989            83.7840
7   01-07-1989            82.7913
8   01-08-1989            83.4055
9   01-09-1989            82.9302
10  01-10-1989            82.6235
11  01-11-1989            82.6971
12  01-12-1989            82.9629
13  01-01-1990            82.3280
14  01-02-1990            82.8759
15  01-03-1990            83.0757
16  01-04-1990            82.7018
17  01-05-1990            82.7671
18  01-06-1990            82.8785
19  01-07-1990            82.5834
20  01-08-1990            82.7370
21  01-09-1990            82.6219
22  01-10-1990            81.9938
23  01-11-1990            80.8637
24  01-12-1990            80.2062
25  01-01-1991            79.8532
26  01-02-1991            79.1938
27  01-03-1991            78.6775
```

R Code: -

ts_data=ts(data$Capacity.Utilization,start=c(1989,1),frequency=12)

ts_data

Output: -

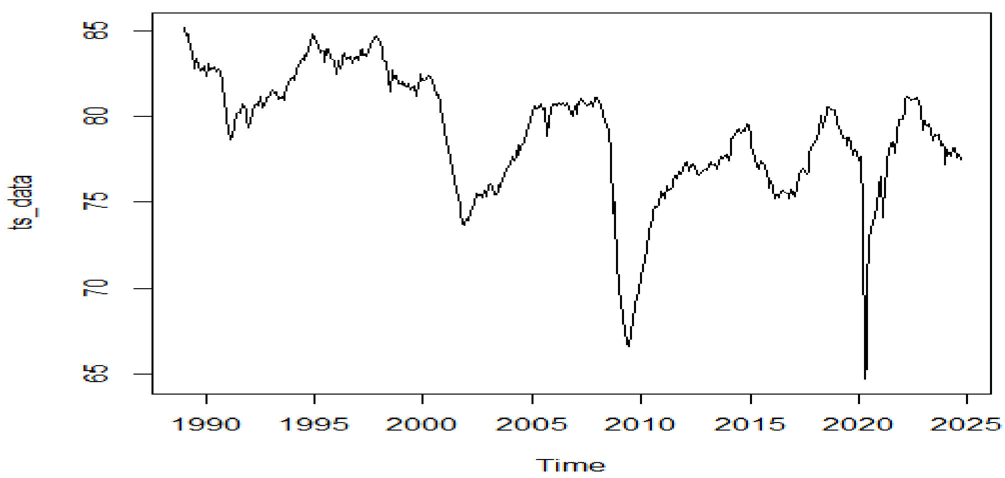
```
          Jan     Feb     Mar     Apr     May     Jun     Jul     Aug     Sep     Oct     Nov     Dec
1989 85.1871 84.6881 84.7698 84.5900 83.9611 83.7840 82.7913 83.4055 82.9302 82.6235 82.6971 82.9629
1990 82.3280 82.8759 83.0757 82.7018 82.7671 82.8785 82.5834 82.7370 82.6219 81.9938 80.8637 80.2062
1991 79.8532 79.1938 78.6775 78.8130 79.5366 80.1091 80.2553 80.1865 80.7568 80.5727 80.3561 79.8698
1992 79.3038 79.7133 80.2192 80.6613 80.7461 80.6077 81.1531 80.5215 80.6052 80.9878 81.1373 81.1101
1993 81.2846 81.5417 81.3332 81.4361 81.0138 81.0769 81.1612 80.9411 81.2154 81.6695 81.8847 82.1813
1994 82.2458 82.0964 82.7001 82.9607 83.1048 83.4090 83.2543 83.5424 83.5225 83.9295 84.2193 84.7763
1995 84.6624 84.2817 84.1264 83.7001 83.8228 83.8038 83.1565 83.9385 83.9436 83.4711 83.3377 83.2653
1996 82.4552 83.2784 82.7819 83.2259 83.4707 83.7026 83.3175 83.3523 83.4837 83.0746 83.4158 83.5292
1997 83.2734 83.8357 83.9386 83.5773 83.6033 83.5281 83.7500 84.0989 84.3562 84.5733 84.6992 84.5069
1998 84.3255 83.9236 83.4139 83.1697 83.1957 82.2129 81.4369 82.6826 82.1974 82.3679 81.9145 81.8823
1999 81.8698 82.0031 81.8486 81.7151 81.9082 81.5543 81.7388 81.8026 81.1741 81.9218 82.0702 82.4391
2000 82.0932 82.0943 82.1340 82.3741 82.3350 82.1191 81.7036 81.2193 81.2955 80.7298 80.4815 79.9449
2001 79.2915 78.5159 78.0741 77.5610 76.9249 76.3013 75.6377 75.3826 74.7982 74.4059 73.8179 73.6523
2002 74.0420 73.9315 74.4010 74.6830 74.9480 75.5244 75.4765 75.3900 75.4643 75.2771 75.7113 75.3192
2003 75.9737 76.0964 75.9158 75.4710 75.4753 75.6024 76.0121 75.8650 76.3685 76.4833 77.0141 77.0642
2004 77.2041 77.6963 77.4077 77.7236 78.3145 77.7240 78.3050 78.3738 78.4547 79.1352 79.3000 79.8779
2005 80.1076 80.6185 80.4254 80.5049 80.4835 80.7304 80.3636 80.5144 78.8408 79.6882 80.4459 80.7149
2006 80.7219 80.6346 80.6930 80.8033 80.6874 80.8131 80.6047 80.7869 80.4643 80.1979 79.9873 80.6149
2007 80.1358 80.7193 80.6791 81.0476 80.9347 80.8222 80.5906 80.6774 80.8326 80.5829 81.0492 81.1309
2008 81.0828 80.8474 80.6547 80.1636 79.7148 79.5288 79.1573 77.8633 74.3831 75.0459 73.9671 71.7765
2009 69.8882 69.3805 68.2145 67.6067 66.8861 66.6446 67.4381 68.2152 68.8533 69.0873 69.4680 69.8230
2010 70.7084 71.1021 71.7572 72.1775 73.3305 73.6492 74.0764 74.4790 74.8001 74.6921 74.8191 75.6199
2011 75.4919 75.1957 75.9692 75.6918 75.7389 75.8967 76.1727 76.5570 76.3854 76.7959 76.6551 76.9204
2012 77.2407 77.3425 76.7980 77.2057 77.2330 77.1126 77.1378 76.6993 76.5385 76.6654 76.8490 76.9607
2013 76.8466 77.1172 77.3418 77.1785 77.1749 77.2476 76.9323 77.3333 77.6904 77.5455 77.6904 77.8052
2014 77.4541 77.9861 78.7000 78.7032 78.9551 79.1445 79.2360 79.0306 79.1927 79.1358 79.5592 79.5137
2015 78.8301 78.2911 77.9964 77.5449 77.1821 76.9429 77.4307 77.3080 77.1039 76.7520 76.2118 75.8450
2016 76.2263 75.8298 75.2411 75.4701 75.2833 75.6283 75.6916 75.5879 75.4977 75.5318 75.2309 75.7622
2017 75.5995 75.3306 75.8344 76.6403 76.7640 76.9926 76.8780 76.6139 76.7906 77.8346 78.1204 78.3837
2018 78.4548 78.7047 79.1096 80.0436 79.3092 79.9674 80.0561 80.5741 80.5711 80.3884 80.3962 80.3580
2019 79.7599 79.2768 79.2123 78.7037 78.7358 78.7532 78.2629 78.7796 78.4615 77.7295 78.1046 77.9031
2020 77.4105 77.6537 74.5527 64.6982 65.7487 70.1340 72.7591 73.5674 73.6543 74.2728 74.7456 75.8651
2021 76.5008 74.0987 76.3455 76.6187 77.4469 77.9619 78.4327 78.5435 77.8344 78.9984 79.7955 79.7947
2022 79.8471 80.3570 80.9989 81.1458 81.1363 80.9375 81.0054 80.9637 81.0936 80.8551 80.4107 79.2286
2023 79.7542 79.6087 79.4362 79.5726 79.2151 78.5968 78.9860 78.8773 78.9282 78.2711 78.4172 78.1414
2024 77.1945 78.0707 77.8455 77.6572 78.1320 78.1784 77.6190 77.8003 77.4942
> 
```

3. PLOT THE TIME SERIES DATA : -

R Code :-

plot(ts_data)

Output : -



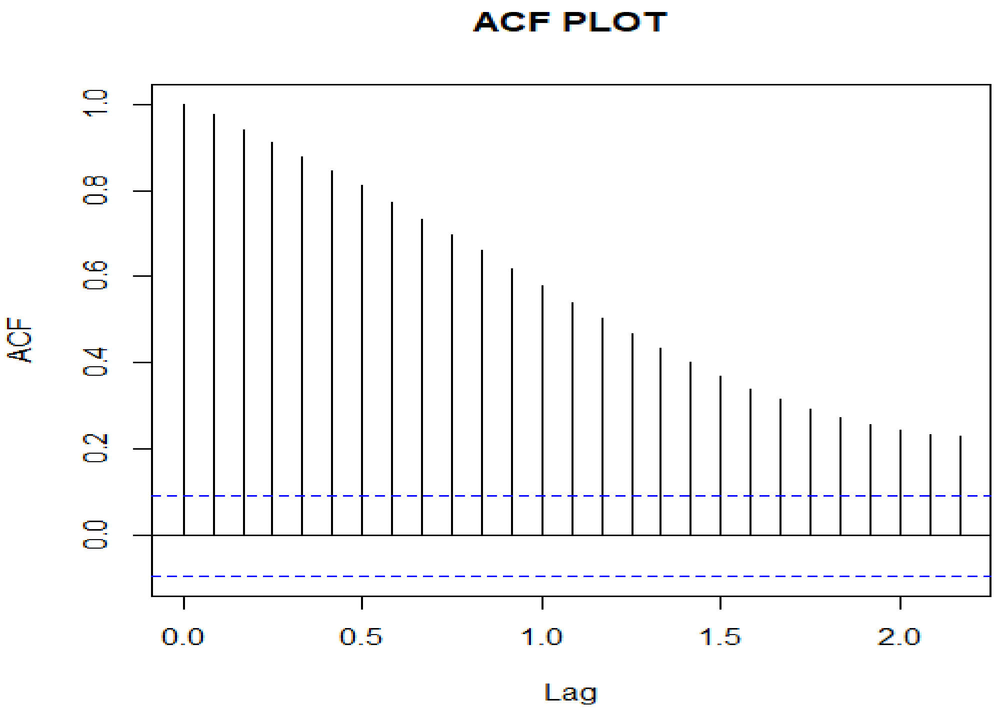
4. ACF PLOT: -

An ACF plot **(Autocorrelation Function)** is a graphical representation used to display the autocorrelations of a time series with its own lagged values over different time intervals (lags).

R Code: -

```
acf(ts_data,main="ACF PLOT")
```

Output:



Interpretation: –

This ACF plot indicates that the time series of Capacity Utilization is likely non-stationary. It shows high autocorrelation at the initial lags. The autocorrelation gradually decreases as the lag increases, that indicating the influence of past values on current values weak over time but remains significant for several lags.

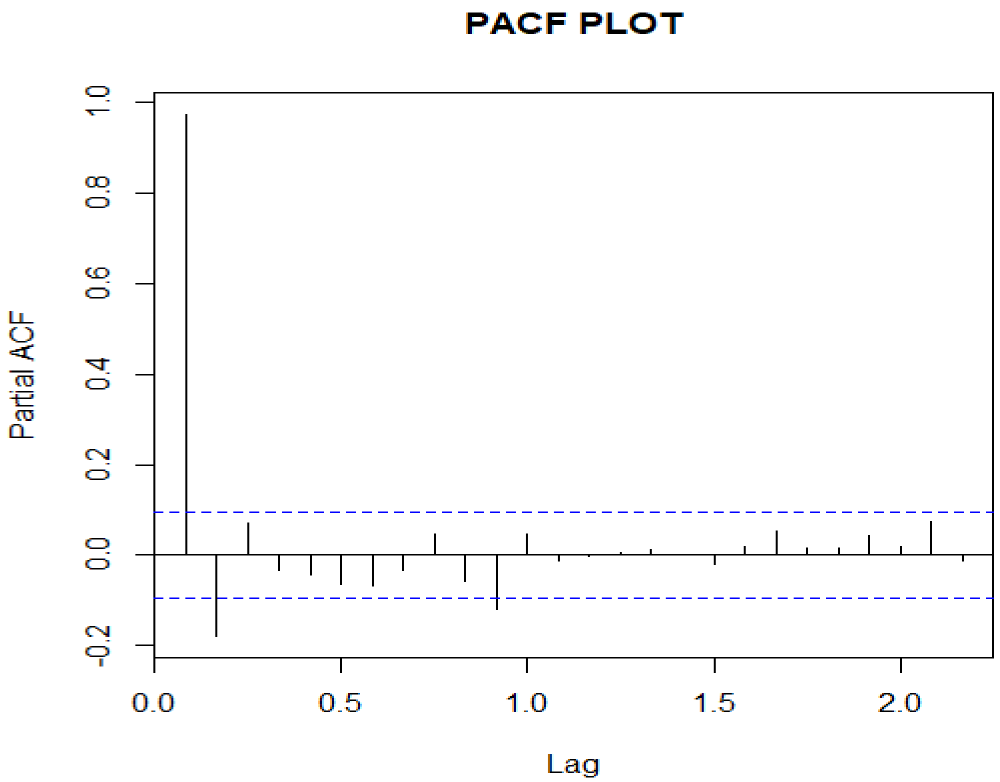
5. PACF PLOT: -

The PACF **(Partial Autocorrelation Function)** plot represent the correlation between a time series and its lagged values, *after controlling for the correlations at all shorter lags*.

R Code: -

```
pacf(ts_data,main="PACF PLOT")
```

Output : -



Interpretation: -

The PACF plot shows that there is a significant spike at lag 1, with the partial autocorrelation close to or above the confidence intervals, i.e. the first lag has a strong relationship with the current value of the series. Beyond lag 1, the spikes fall mostly within the confidence intervals, indicating that further lags may not add much explanatory power i.e. the influence of past values beyond the first lag is minimal.

6. ADF TEST (): -

The ADF test (**Augmented Dickey-Fuller test**) is a statistical test used to determine whether a time series is stationary or contains a unit root

R Code: -

```
install.packages("tseries")
library(tseries)
adf.test(ts_data)
```

Output: -

```
> library(tseries)
> adf.test(ts_data)

        Augmented Dickey-Fuller Test
data:  ts_data
Dickey-Fuller = -3.5806, Lag order = 7, p-value = 0.03483
alternative hypothesis: stationary
```

Interpretation: -

The Augmented Dickey-Fuller (ADF) test provides information on the stationarity of the time series **ts_data**. P-value = **0.03483** that is less than 0.05 and a strongly negative test statistic (-3.5806), we can **reject the null hypothesis** of non-stationarity. This means the data is likely **stationary**, with constant mean and variance over time, making it suitable for time series analysis methods that assume stationarity, such as ARIMA modelling. These results confirm that the data does not require differencing or transformation for stationarity, which simplifies further analysis and modelling.

7. DURBIN WATSON TEST: -

The Durbin-Watson test is used to detect the presence of autocorrelation in the residuals of a regression model.

R Code: -

```
ar2_model = Arima(ts_data, order = c(2, 0, 0), seasonal = c(0, 0, 0))

residuals_ar1 = residuals(ar1_model)

dw_test_result = dwtest(residuals_ar1 ~ 1)

print(dw_test_result)
```

Output: -

```
> print(dw_test_result)
```

Durbin-Watson test

data: residuals_ar1 ~ 1

DW = 1.9583, p-value = 0.3327

alternative hypothesis: true autocorrelation is greater than 0

Interpretation: -

- The Durbin-Watson statistic (DW) is approximately 1.9583, which is close to 2. A DW value near 2 suggests little to no autocorrelation in the residuals.
- The p-value is 0.3327, which is greater than the significance level (0.05). This indicates that we do not have enough evidence to reject the null hypothesis of no autocorrelation.
- There is no significant autocorrelation in the residuals, suggesting that the model residuals are likely independent.

8. AUTO REGRESSIVE MODEL: -

R Code: -

```
result1 = data.frame(Model = character(), AIC = numeric(), BIC = numeric(),
Coefficients = I(list()))

for (lag in 1:5)

{

 model_ar = arima(ts_data, order = c(lag, 0, 0))  # Fit AR model

 result1 = rbind(result1, data.frame(Model= "AR",AIC = AIC(model_ar),

 BIC = BIC(model_ar), Coefficients =I(list(coef(model_ar)))

   ))

 }

result1

best_aic_model = result1[which.min(result1$AIC), ] ; best_aic_model

best_bic_model = result1[which.min(result1$BIC), ] ; best_bic_model

write.csv(result1,file='result1.csv')
```

Output: -

```
> result1
  Model     AIC      BIC Coefficients
1    AR 981.8929 994.0773 0.979028....
2    AR 962.8761 979.1219 1.193065....
3    AR 960.0751 980.3824 1.216288....
4    AR 960.8016 985.1704 1.222183....
5    AR 961.6514 990.0816 1.219376....
> best_aic_model = result1[which.min(result1$AIC), ];best_aic_model
  Model     AIC      BIC Coefficients
3    AR 960.0751 980.3824 1.216288....
> best_bic_model = result1[which.min(result1$BIC), ];best_bic_model
  Model     AIC      BIC Coefficients
2    AR 962.8761 979.1219 1.193065....
> write.csv(result1,file='result1.csv')
```

Lag	Model	AIC	BIC	Coefficients	
1	AR	981.8929437	994.0773145	c(ar1 = 0.97902845245944	intercept = 79.3669025198672)
2	AR	962.8761036	979.1219313	c(ar1 = 1.19306533467439	ar2 = -0.219063150407426
3	AR	960.0751291	980.3824137	c(ar1 = 1.21628854583437	ar2 = -0.344886083489142
4	AR	960.8016338	985.1703753	c(ar1 = 1.22218321231782	ar2 = -0.363942579070058
5	AR	961.6513643	990.0815627	c(ar1 = 1.2193762209306	ar2 = -0.35496207756897

intercept = 79.1955294043461)			
ar3 = 0.105581769786689	intercept = 79.2520597824176)		
ar3 = 0.171861180876373	ar4 = -0.0544667643311653	intercept = 79.2442266637313)	
ar3 = 0.152816433732336	ar4 = 0.00880362834360281	ar5 = -0.0517406564882985	intercept = 79.2097598542999)

Interpretation : -

The output shows the **AIC (Akaike Information Criterion)** and **BIC (Bayesian Information Criterion)** values for various autoregressive (AR) models with different lag values. The goal is to identify the optimal lag by selecting the model with the lowest AIC and BIC values, as these indicate the best fit with an appropriate balance between model complexity and goodness of fit.

Optimal Model based on AIC and BIC values:

- **Best AIC Model**: The model with the minimum AIC is considered to have the best fit in terms of balancing model complexity and goodness of fit. In this case the minimum AIC value (**960.0751**) is at **Lag 3**.
- **Best BIC Model**: The model with the minimum BIC is generally preferred when we want a model that avoids overfitting. BIC imposes a larger penalty for additional parameters than AIC, in this case the minimum BIC value (**979.1219**) is at **Lag 2**.

9. MOVING AVERAGE MODEL: -

R Code: -

```
result2 = data.frame(Model = character(), AIC = numeric(), BIC = numeric(), Coefficients = I(list()))

for (lag in 1:5)

{

 model_ma = arima(ts_data, order = c(0, 0,lag))  # Fit MA model

 result2 = rbind(result2, data.frame(Model= "MA",AIC = AIC(model_ma),

 BIC = BIC(model_ma), Coefficients =I(list(coef(model_ma)))

   ))

 }

result2

best_aic_model = result2[which.min(result2$AIC), ]; best_aic_model

best_bic_model = result2[which.min(result2$BIC), ];best_bic_model

write.csv(result1,file='result2.csv')
```

Output: -

```
> result2
  Model      AIC      BIC Coefficients
1    MA 1823.833 1836.017 0.890757....
2    MA 1506.215 1522.461 1.279314....
3    MA 1345.548 1365.855 1.540237....
4    MA 1220.065 1244.434 1.505878....
5    MA 1168.559 1196.989 1.529641....
>
> best_aic_model = result2[which.min(result2$AIC), ]; best_aic_model
  Model      AIC      BIC Coefficients
5    MA 1168.559 1196.989 1.529641....
> best_bic_model = result2[which.min(result2$BIC), ];best_bic_model
  Model      AIC      BIC Coefficients
5    MA 1168.559 1196.989 1.529641....
> write.csv(result1,file='result2.csv')
```

lag	Model	AIC	BIC	Coefficients	
1	MA	1823.832923	1836.017294	c(ma1 = 0.890757148580549	intercept = 78.9217348723192)
2	MA	1506.215271	1522.461099	c(ma1 = 1.27931459012969	ma2 = 0.780809109995436
3	MA	1345.547706	1365.854991	c(ma1 = 1.54023701150067	ma2 = 1.23375545058101
4	MA	1220.064949	1244.43369	c(ma1 = 1.5058785634981	ma2 = 1.47088406451552
5	MA	1168.558808	1196.989006	c(ma1 = 1.52964157539918	ma2 = 1.58747247985158

intercept = 78.9278886566672)			
ma3 = 0.486974511976047	intercept = 78.9304385065785)		
ma3 = 1.10696715341044	ma4 = 0.475082056244699	intercept = 78.9420885651376)	
ma3 = 1.30322107537324	ma4 = 0.850333462970777	ma5 = 0.331630894487946	intercept = 78.9412031921725)

Interpretation: -

The output presents the **AIC (Akaike Information Criterion)** and **BIC (Bayesian Information Criterion)** values for MA models with different lag orders (1 to 5). These values help determine the best model by balancing goodness of fit with model complexity.

Optimal Model based on AIC and BIC values: -

- **Best AIC Model**: The model with the minimum AIC value is at **Lag 5** with an AIC of **1168.559.**
- **Best BIC Model**: Similarly, the model with the minimum BIC value is also at **Lag 5** with a BIC of **1196.989**.

Conclusion: -

Since both the **AIC** and **BIC** values are minimum at **Lag 5**, that is the MA model at lag 5 is likely the best fit for the data according to both criteria. The coefficients of MA model represent the influence of the last five periods 'errors on the current value, with ma1 having the highest effect and ma5 the smallest.

10. AUTO ARIMA MODEL: -

R Code: -

model=auto.arima(ts_data); model

summary(model)

Output: -

```
> model=auto.arima(ts_data); model
Series: ts_data
ARIMA(0,1,1)

Coefficients:
         ma1
      0.2528
s.e.  0.0498

sigma^2 = 0.5423:  log likelihood = -475.86
AIC=955.72   AICc=955.75   BIC=963.84
> summary(model)
Series: ts_data
ARIMA(0,1,1)

Coefficients:
         ma1
      0.2528
s.e.  0.0498

sigma^2 = 0.5423:  log likelihood = -475.86
AIC=955.72   AICc=955.75   BIC=963.84

Training set error measures:
                      ME      RMSE       MAE         MPE      MAPE     MASE
Training set -0.01427608 0.7346654 0.4139684 -0.02144852 0.5387145 0.189302
                    ACF1
Training set -0.01726195
```

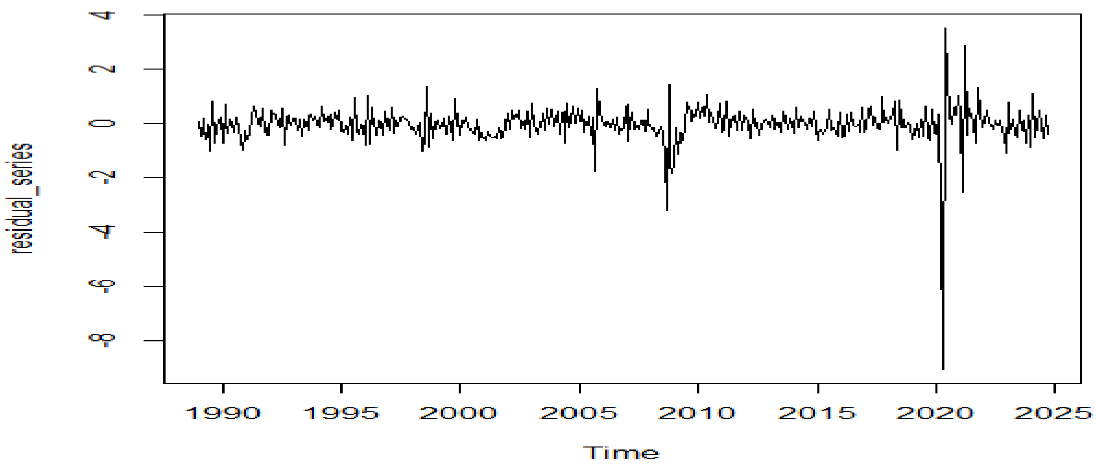
11. RESIDUAL ANALYSIS: -

R Code: -

residual_series=residuals(model)

plot(residuals_series)

Output: -



Interpretation: -

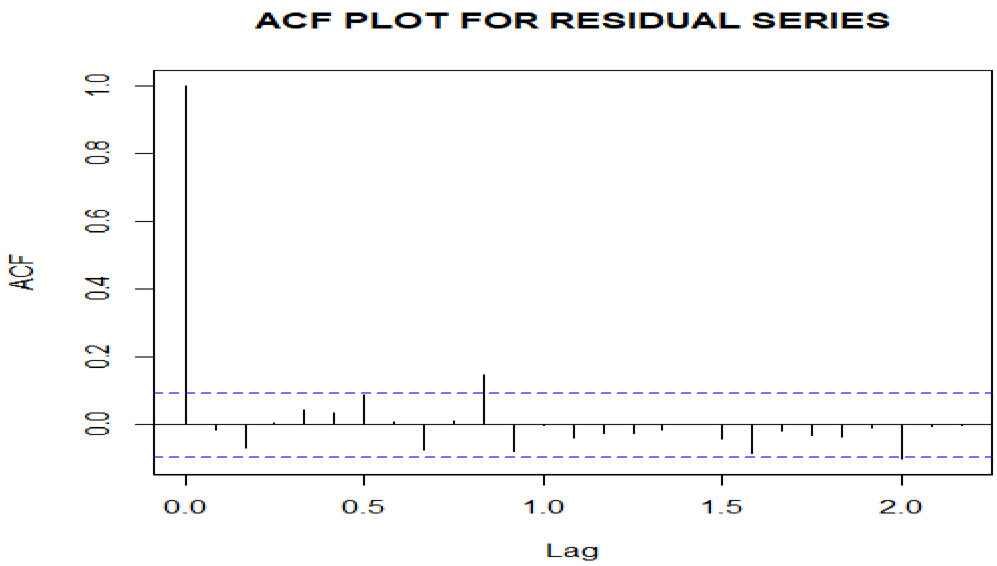
- The plot shows the residual series from a time series model, where residuals represent the differences between the observed values and the model's predicted values.
- The residual plot shows that the residuals are generally centered around zero with consistent variance, indicating that the model has captured the main structure of the time series data. However, there are notable outliers around 2020, with large spikes in both positive and negative directions, suggesting possible external shocks or events not accounted for by the model.
- Overall, aside from these outliers, the residuals should resemble white noise appear to be random, implying that the model fits the data well. Addressing the 2020 outliers could potentially improve the model's performance which could require further investigation or adjustment.

A. **ACF PLOT: -** An ACF plot of the residual series allows to visually inspect whether the residuals are close to white noise.

R Code: -

acf(residual_series,main="ACF PLOT FOR RESIDUALS")

Output: -



Interpretation: -

The ACF plot for the residual series shows that most values fall within the confidence intervals, indicating that the residuals are approximately uncorrelated, the lack of significant spikes confirms that the residuals resemble white noise, supporting the model's adequacy. Overall, this ACF plot of residual series implies a good model fit.

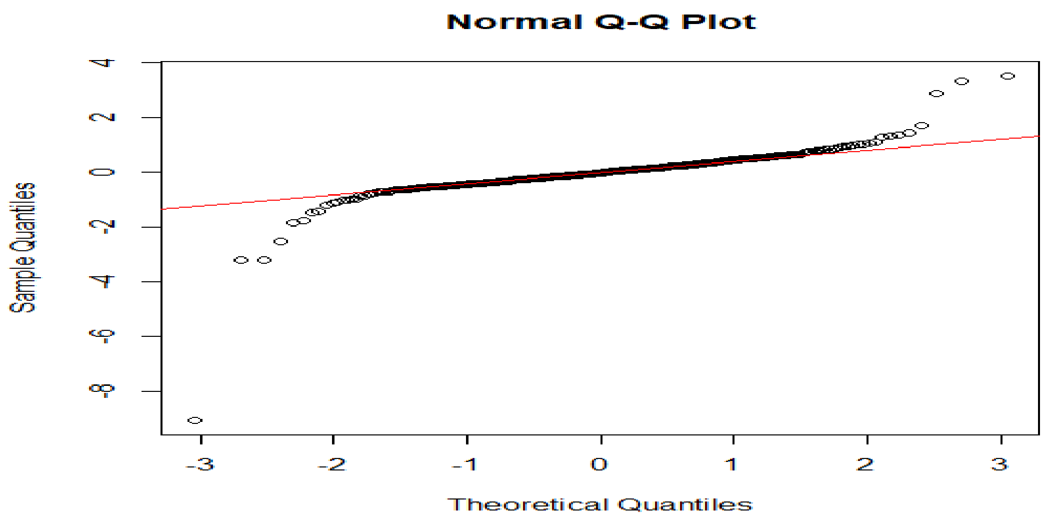
B. QQ PLOT: - A Q-Q (Quantile-Quantile) Plot is a graphical tool used to assess whether a dataset follows a particular theoretical distribution, most commonly the normal distribution.

R Code: -

```
qqnorm(residual_series)

qqline(residual_series,col="red")
```

Output: -



Interpretation: -

In the Q-Q plot, most points closely follow the red reference line, indicating that the data largely resembles a normal distribution. However, the points deviate from the line at both the lower and upper extremes, i.e. heavier tails than a normal distribution. This deviation implies potential outliers or a distribution with more extreme values than normal. While the data has a generally normal shape, it may not be perfectly normal due to these tail deviations.

C. BOX TEST: - The Box test, commonly known as the Box-Pierce test or Ljung-Box test, is a statistical test used to check for autocorrelation in a time series. Autocorrelation means that current values in a series are correlated with past values, which can affect the independence assumption in time series models.

R Code: -

```
Box_test = Box.test(residual_series , lag = 10)
 Box_test
if (Box_test$p.value<0.05)
{ cat("There is significant autocorrelation in the residuals.\n")
}else{
cat("autocorrelation is not present in the residuals.\n")  }
```

Output: -

```
> Box_test

      Box-Pierce test

data:  residual_series
X-squared = 18.099, df = 10, p-value = 0.05332
> if (Box_test$p.value<0.05)
+ { cat("There is significant autocorrelation in the residuals.\n")
+ }else{
+ cat("autocorrelation is not present in the residuals.\n") }
```

autocorrelation is not present in the residuals.

Interpretation: -

As we see Box test does not find evidence of autocorrelation in the residuals, it means that the residuals are likely independent and randomly distributed. This is a good indication that the model has captured the underlying patterns in the data and that the errors are not systematically related to each other. A model with independent residuals is generally considered to be a well-fitting model. Hence, we can proceed to forecasting the future values of the model.

12. FORECASTING: -

R Code: -

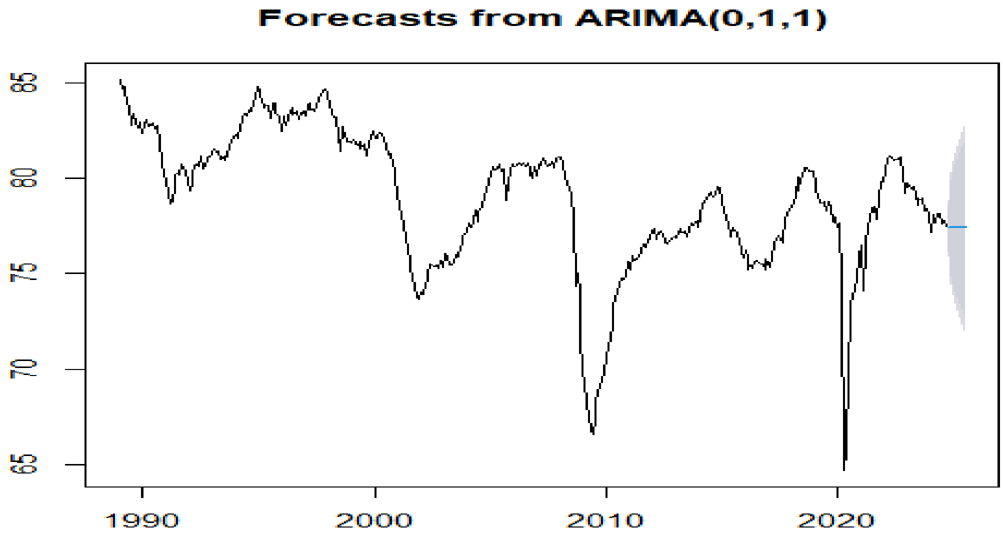
```
forecast_value=forecast(model,h=10,level=c(0.90,0.95))
forecast_value
# Plot of forecast values
plot(forecast_value)
```

Output: -

> forecast_value

	Point Forecast	Lo 90	Hi 90	Lo 95	Hi 95
Oct 2024	77.39651	76.18527	78.60776	75.95323	78.83980
Nov 2024	77.39651	75.45493	79.33809	75.08298	79.71005
Dec 2024	77.39651	74.93230	79.86073	74.46022	80.33281
Jan 2025	77.39651	74.50256	80.29047	73.94816	80.84487
Feb 2025	77.39651	74.12886	80.66417	73.50286	81.29017
Mar 2025	77.39651	73.79371	80.99932	73.10351	81.68952
Apr 2025	77.39651	73.48719	81.30584	72.73827	82.05476
May 2025	77.39651	73.20301	81.59001	72.39965	82.39338
Jun 2025	77.39651	72.93691	81.85612	72.08257	82.71046
Jul 2025	77.39651	72.68581	82.10721	71.78337	83.00966

> plot(forecast_value)



Interpretation: -

The forecast values show the point estimates along with prediction intervals for the next 10 months (October 2024 to July 2025) for a given time series. The "Point Forecast" column shows the model's best prediction for each month's value. That is most likely estimate based on the data and model parameters at 90 % and 95% confidence interval. For October 2024, there is a 90% chance that the actual value of model parameter will fall between 76.18527 and 78.60776 and the 95% chance that the actual value of model parameter will fall between 75.95323 and 78.83980.