# Germany Unemployment predictions

## Importing dataset:

```
rm(list=ls())
data_germany=read.csv("Germany_Unemployment.csv")
head(data_germany)
```

```
##       Date Unemployment_Rates Unemployed_all  CPI Employed
## 1 1/1/1991              0.057        1606000 60.5 39317000
## 2 2/1/1991              0.057        1608000 60.6 39190000
## 3 3/1/1991              0.056        1583000 60.6 39068000
## 4 4/1/1991              0.056        1572000 60.9 39056000
## 5 5/1/1991              0.056        1583000 60.9 39021000
## 6 6/1/1991              0.056        1581000 61.4 38923000
##   Orders_received_constant_price_index Gross_wages_salaries_main_industry
## 1                                 55.8                            4084615
## 2                                 53.4                            3541725
## 3                                 54.3                            4136554
## 4                                 53.2                            4220291
## 5                                 52.9                            4206204
## 6                                 53.7                            4246912
##   Hours_worked_main_industry
## 1                     149506
## 2                     120704
## 3                     136216
## 4                     134023
## 5                     132533
## 6                     131368
```

```
tail(data_germany)
```

```
##          Date Unemployment_Rates Unemployed_all   CPI Employed
## 382 10/1/2022              0.055        2512000 113.5 45673000
## 383 11/1/2022              0.055        2531000 113.9 45713000
## 384 12/1/2022              0.055        2522000 113.4 45765000
## 385  1/1/2023              0.055        2517000 114.8 45809000
## 386  2/1/2023              0.055        2524000 115.5 45868000
## 387  3/1/2023              0.056        2543000 116.2 45924000
##     Orders_received_constant_price_index Gross_wages_salaries_main_industry
## 382                                125.2                            1934031
## 383                                121.7                            1966927
## 384                                123.9                            1941358
## 385                                125.3                            1956140
## 386                                130.6                            2035036
## 387                                117.2                            1947763
##     Hours_worked_main_industry
## 382                      52349
## 383                      52475
## 384                      47229
## 385                      52077
## 386                      54060
## 387                      52586
```

# Checking Multicollinearity using VIFs:

```
suppressWarnings(library(regclass))
```

```
## Loading required package: bestglm
```

```
## Loading required package: leaps
```

```
## Loading required package: VGAM
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
## Loading required package: rpart
```

```
## Loading required package: randomForest
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
## Important regclass change from 1.3:
## All functions that had a . in the name now have an _
## all.correlations -> all_correlations, cor.demo -> cor_demo, etc.
```

```
VIF(lm(formula = Unemployment_Rates ~ Unemployed_all+CPI+Employed+Orders_received_constant_price_i
ndex+Gross_wages_salaries_main_industry+Hours_worked_main_industry, data = data_germany))
```

```
##                       Unemployed_all                                CPI
##                             4.676527                          32.486908
##                             Employed Orders_received_constant_price_index
##                            29.481919                          10.529845
##   Gross_wages_salaries_main_industry          Hours_worked_main_industry
##                            19.565388                          27.043576
```

## Removing CPI:

```
VIF(lm(formula = Unemployment_Rates ~ Unemployed_all+Employed+Orders_received_constant_price_index
+Gross_wages_salaries_main_industry+Hours_worked_main_industry, data = data_germany))
```

```
##                       Unemployed_all                           Employed
##                             3.941744                          11.010665
## Orders_received_constant_price_index   Gross_wages_salaries_main_industry
##                             8.399168                          14.561638
##           Hours_worked_main_industry
##                            19.704522
```

## Removing Hours_worked_main_industry:

```
VIF(lm(formula = Unemployment_Rates ~ Unemployed_all+Employed+Orders_received_constant_price_index
+Gross_wages_salaries_main_industry, data = data_germany))
```

```
##                       Unemployed_all                           Employed
##                             3.091495                          10.336456
## Orders_received_constant_price_index   Gross_wages_salaries_main_industry
##                             8.100304                           2.459236
```

## Removing Employed:

```
VIF(lm(formula = Unemployment_Rates ~ Unemployed_all+Orders_received_constant_price_index+Gross_wa
ges_salaries_main_industry, data = data_germany))
```

```
##                       Unemployed_all Orders_received_constant_price_index
##                             1.890923                           3.245670
##   Gross_wages_salaries_main_industry
##                             2.391027
```

This is the final set of variables free from multicollinearity.

# Train-Test split of the dataset - last 6 months of the data

would be taken into testing part:

```
df_germany_train1=data_germany[1:(nrow(data_germany)-6),]
df_germany_test1=data_germany[(nrow(data_germany)-5):nrow(data_germany),]
head(df_germany_train1)
```

```
##       Date Unemployment_Rates Unemployed_all  CPI Employed
## 1 1/1/1991              0.057        1606000 60.5 39317000
## 2 2/1/1991              0.057        1608000 60.6 39190000
## 3 3/1/1991              0.056        1583000 60.6 39068000
## 4 4/1/1991              0.056        1572000 60.9 39056000
## 5 5/1/1991              0.056        1583000 60.9 39021000
## 6 6/1/1991              0.056        1581000 61.4 38923000
##   Orders_received_constant_price_index Gross_wages_salaries_main_industry
## 1                                 55.8                            4084615
## 2                                 53.4                            3541725
## 3                                 54.3                            4136554
## 4                                 53.2                            4220291
## 5                                 52.9                            4206204
## 6                                 53.7                            4246912
##   Hours_worked_main_industry
## 1                     149506
## 2                     120704
## 3                     136216
## 4                     134023
## 5                     132533
## 6                     131368
```

- The model would be trained on the train dataset.
- And the performance of the fitted model would be checked on the test dataset.
- If this performs fairly well, this model would be considered to get the future forecasts.

# Time series plot:

```
suppressWarnings(library(fpp2))
```

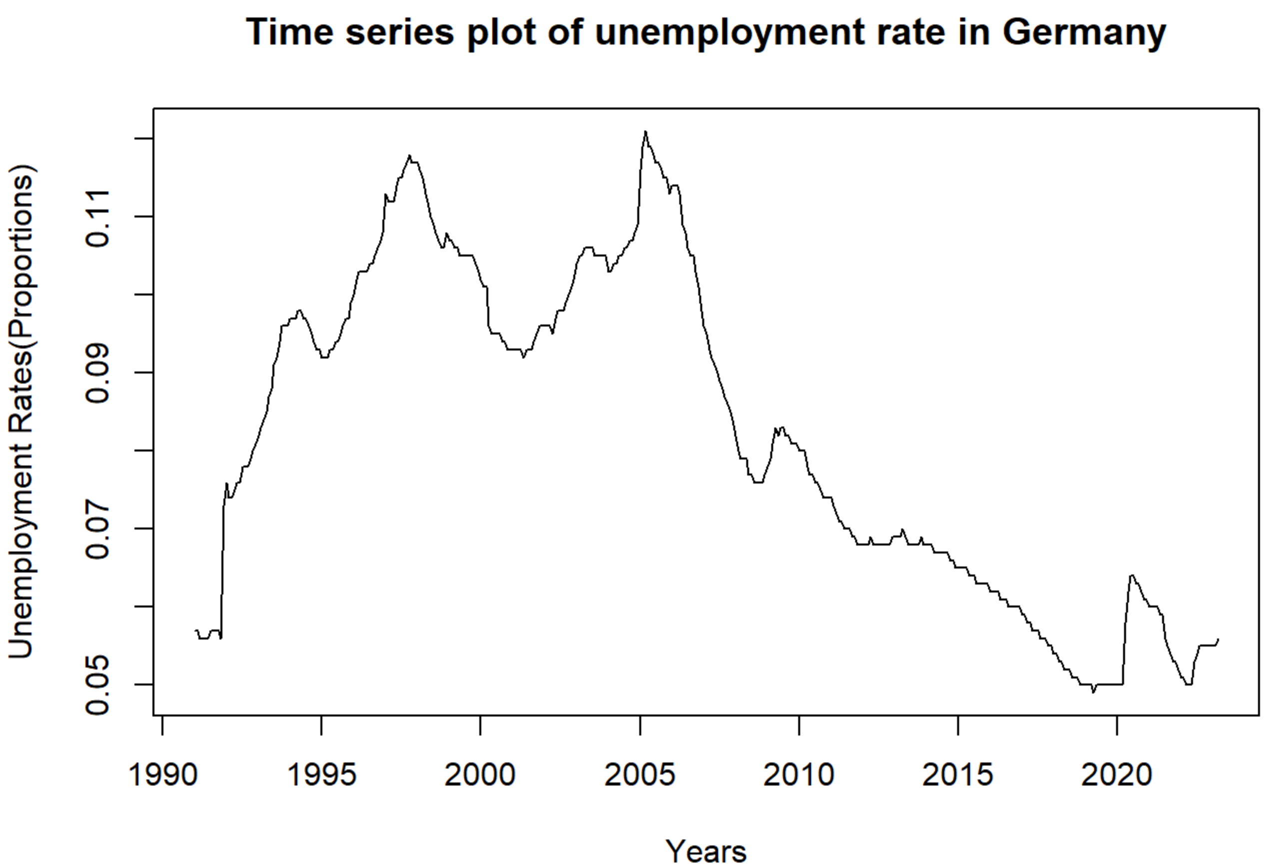
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## ── Attaching packages ────────────────────────────────────────────── fpp2 2.5 ──
```

```
## ✔ ggplot2   3.3.6     ✔ fma       2.4
## ✔ forecast  8.18      ✔ expsmooth 2.3
```

```
## ── Conflicts ───────────────────────────────────────────────── fpp2_conflicts ──
## ✖ ggplot2::margin() masks randomForest::margin()
```

```
suppressWarnings(library(urca))
df.ts=ts(data_germany$Unemployment_Rates, frequency = 12, start = c(1991,1))
plot(df.ts,xlab="Years",ylab="Unemployment Rates(Proportions)")
title(main="Time series plot of unemployment rate in Germany")
```




# Testing stationarity:

```
df_germany_train1[,"Unemployment_Rates"] %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 3.8739
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

This series is non-stationary - 1st order differencing would be necessary.

# Testing stationarity after 1st order differencing:

```
diff(df_germany_train1[,"Unemployment_Rates"]) %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.672
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

1st order differences are also non-stationary. 2nd order differencing would be necessary.

# Testing stationarity after 2nd order differencing:

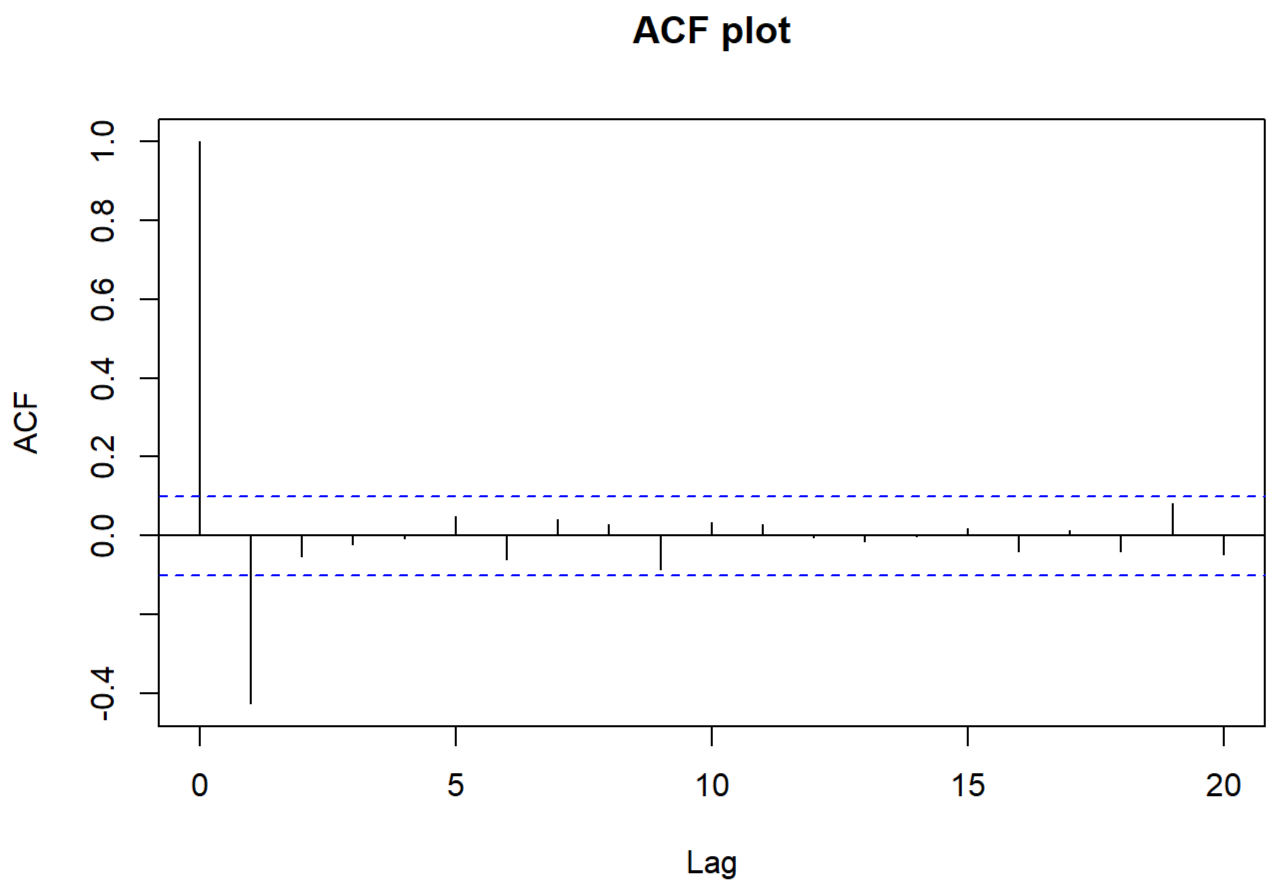
```
diff(diff(df_germany_train1[,"Unemployment_Rates"])) %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.0089
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

2nd order differences are stationary.

# ACF plot:

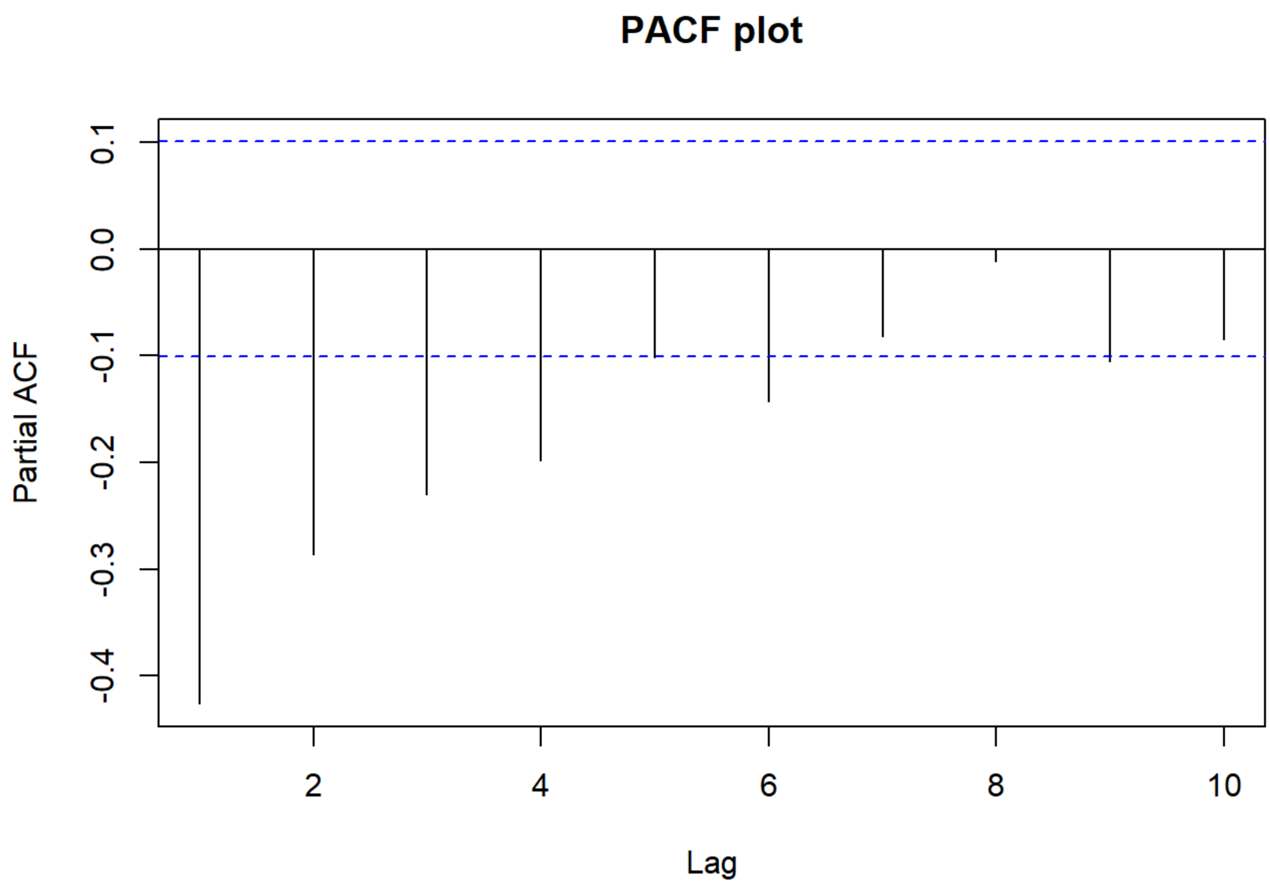
```
par(mfrow=c(1,1))
acf(diff(diff(df_germany_train1$Unemployment_Rates)), lag.max = 20, main = "ACF plot")
```




p can be taken as 0/1 based on the no. of significant lags.

# PACF plot:

```
par(mfrow=c(1,1))
pacf(diff(diff(df_germany_train1$Unemployment_Rates)), lag.max = 10, main = "PACF plot")
```




q can be 0/1/2/3/4, based on the no. of significant lags.

# Fitting ARIMAX model ignoring the variables that were eliminated due to high VIF:

## Starting with the value of p as 1 & q as 4 and with the rest of the regressors:

```
est_train=arima(df_germany_train1$Unemployment_Rates, order=c(1,2,4), xreg = as.matrix(df_germany_
train1[,c(3,6,7)]), method = "ML")
summary(est_train)
```

```
##
## Call:
## arima(x = df_germany_train1$Unemployment_Rates, order = c(1, 2, 4), xreg = as.matrix(df_germany
_train1[,
##     c(3, 6, 7)]), method = "ML")
##
## Coefficients:
```

```
## Warning in sqrt(diag(x$var.coef)): NaNs produced
```

```
##          ar1      ma1     ma2      ma3     ma4  Unemployed_all
##       0.8794  -1.9950  1.2661  -0.3434  0.0723               0
## s.e.  0.0623   0.0807  0.1249   0.1199  0.0552             NaN
##       Orders_received_constant_price_index  Gross_wages_salaries_main_industry
##                                     0e+00                                  0
## s.e.                                1e-04                                NaN
##
## sigma^2 estimated as 3.439e-07:  log likelihood = 2280.02,  aic = -4542.04
##
## Training set error measures:
##                         ME         RMSE          MAE         MPE      MAPE
## Training set -1.317463e-05 0.0005848607 0.0004188509 -0.01048982 0.5327821
##                  MASE        ACF1
## Training set 0.564409 -0.00132728
```

## Test of significance of individual coefficients:

```
suppressWarnings(library(lmtest))
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'lmtest'
```

```
## The following object is masked from 'package:VGAM':
##
##     lrtest
```

```
coeftest(est_train)
```

```
## Warning in sqrt(diag(se)): NaNs produced
```

```
##
## z test of coefficients:
##
##                                      Estimate  Std. Error  z value  Pr(>|z|)
## ar1                                 8.7940e-01  6.2316e-02  14.1120 < 2.2e-16
## ma1                                -1.9950e+00  8.0699e-02 -24.7209 < 2.2e-16
## ma2                                 1.2661e+00  1.2489e-01  10.1379 < 2.2e-16
## ma3                                -3.4339e-01  1.1991e-01  -2.8637  0.004188
## ma4                                 7.2257e-02  5.5231e-02   1.3083  0.190783
## Unemployed_all                      1.7081e-08         NaN      NaN       NaN
## Orders_received_constant_price_index -1.4610e-05  6.6961e-05  -0.2182  0.827285
## Gross_wages_salaries_main_industry   -1.8898e-10         NaN      NaN       NaN
##
## ar1                                ***
## ma1                                ***
## ma2                                ***
## ma3                                **
## ma4
## Unemployed_all
## Orders_received_constant_price_index
## Gross_wages_salaries_main_industry
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We need to remove the variables producing NaNs & the insignificant variables.

## After doing that, the summary & test of significances of the final model would look like:

```
est_1=arima(df_germany_train1$Unemployment_Rates, order=c(1,2,2), method = "ML")
summary(est_1)
```

```
##
## Call:
## arima(x = df_germany_train1$Unemployment_Rates, order = c(1, 2, 2), method = "ML")
##
## Coefficients:
##          ar1      ma1     ma2
##       0.8096  -1.6101  0.6148
## s.e.  0.1098   0.1421  0.1381
##
## sigma^2 estimated as 1.91e-06:  log likelihood = 1956.4,  aic = -3904.81
##
## Training set error measures:
##                         ME        RMSE          MAE        MPE     MAPE
## Training set -6.823572e-05 0.001378249 0.0007321423 -0.0470984 0.915872
##                   MASE       ACF1
## Training set 0.9865747 0.03148091
```

## Test of significance of coefficients:

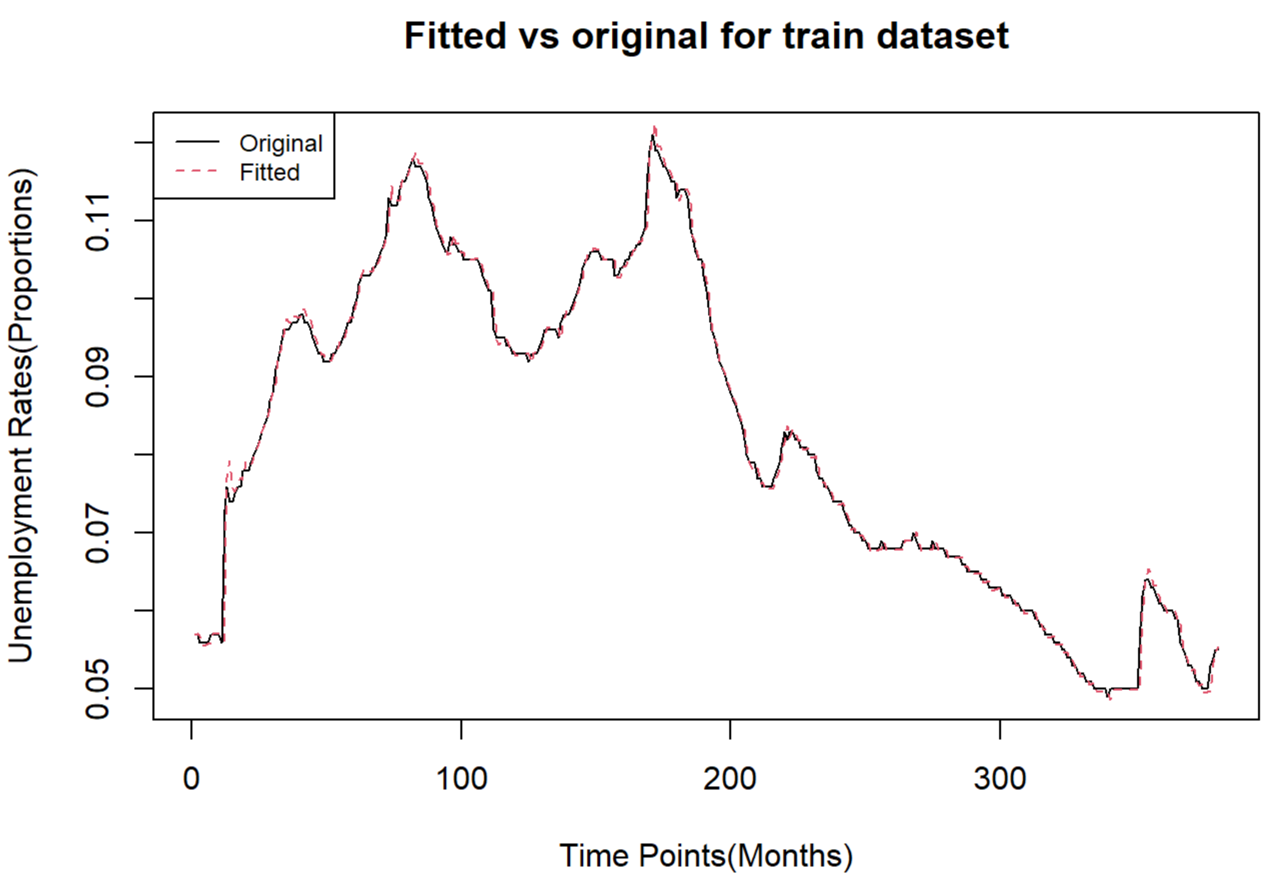
```
suppressWarnings(library(lmtest))
coeftest(est_1)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.80959    0.10980   7.3732 1.665e-13 ***
## ma1 -1.61010    0.14209 -11.3316 < 2.2e-16 ***
## ma2  0.61481    0.13807   4.4530 8.470e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus only those parameters are kept as final, which are significant in prediction of the target variable.

# Plot of Fitted vs Original values for train dataset:

```
res=residuals(est_1)
data_fit=df_germany_train1$Unemployment_Rates-res
ts.plot(df_germany_train1$Unemployment_Rates, type="l", xlab="Time Points(Months)", ylab="Unemploy
ment Rates(Proportions)", main="Fitted vs original for train dataset")
points(data_fit, type="l", col=2, lty=2)
legend("topleft",c("Original","Fitted"), col=c(1,2), lty=c(1,2), cex=0.75)
```



# Predictions of unemployment rates for the test dataset using above fitted model:

```
test_pred=predict(est_1, n.ahead=6, se.fit=FALSE, method="ML")
```

## Predicted values:

```
print(as.vector(test_pred))
```

```
## [1] 0.05526293 0.05546042 0.05560493 0.05570655 0.05577344 0.05581222
```

## Original values:

```
print(df_germany_test1$Unemployment_Rates)
```

```
## [1] 0.055 0.055 0.055 0.055 0.055 0.056
```

# Performance on test dataset:

MAPE (in %):

```
(1/length(df_germany_test1$Unemployment_Rates))*(sum(abs(df_germany_test1$Unemployment_Rates-as.ve
ctor(test_pred))/abs(df_germany_test1$Unemployment_Rates)))*100
```

```
## [1] 0.9068755
```

RMSE:

```
sqrt(mean((df_germany_test1$Unemployment_Rates-as.vector(test_pred))^2))
```

```
## [1] 0.0005446309
```

Thus, the fitted model is working well, more or less, for future dataset.

# Now going with the same approach with the actual dataset for getting the future forecast of March,23:

## Checking stationarity:

```
data_germany[,"Unemployment_Rates"] %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 4.0121
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
diff(data_germany[,"Unemployment_Rates"]) %>%
  ur.kpss() %>%
  summary()
```

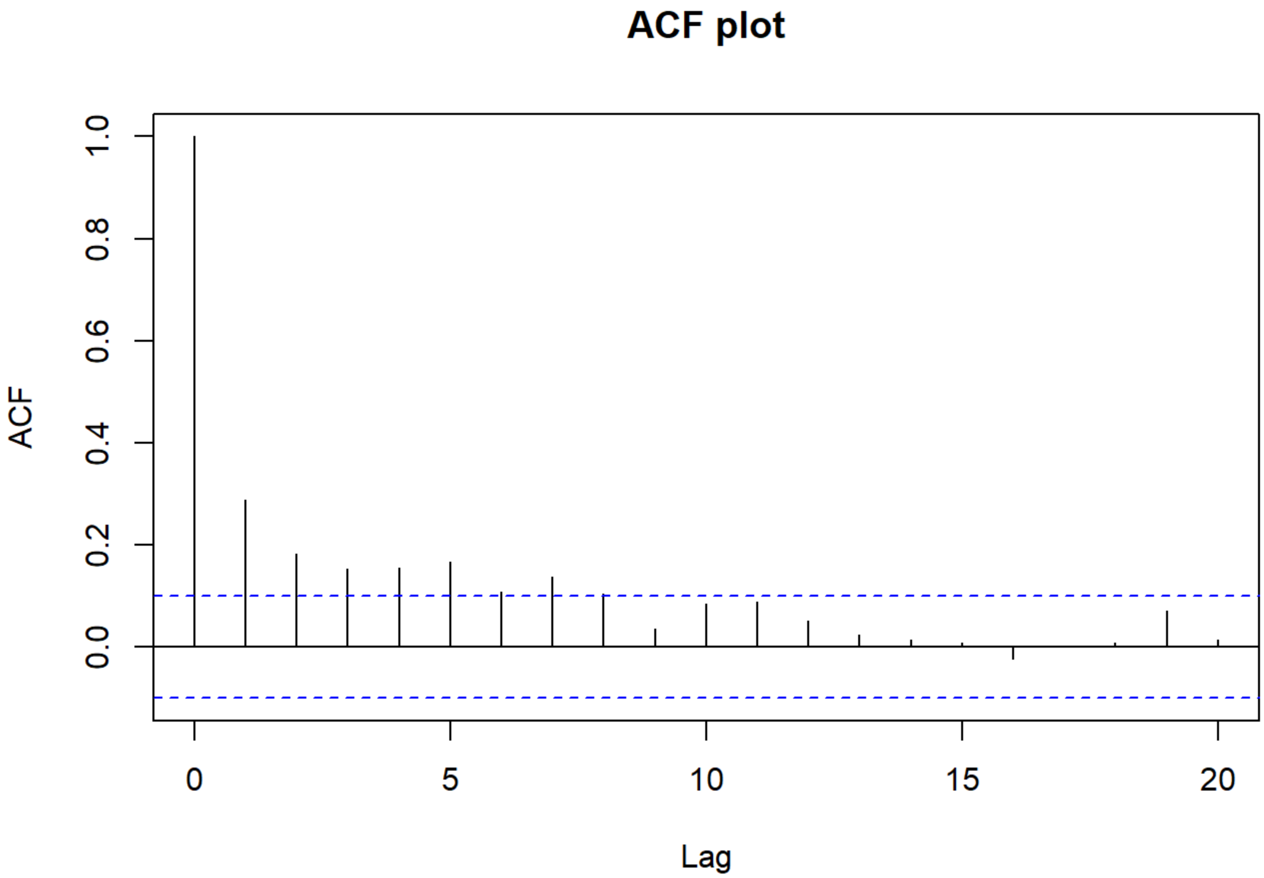
```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.6502
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

```
diff(diff(data_germany[,"Unemployment_Rates"])) %>%
  ur.kpss() %>%
  summary()
```

```
##
## #######################
## # KPSS Unit Root Test #
## #######################
##
## Test is of type: mu with 5 lags.
##
## Value of test-statistic is: 0.0107
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

# ACF plot:

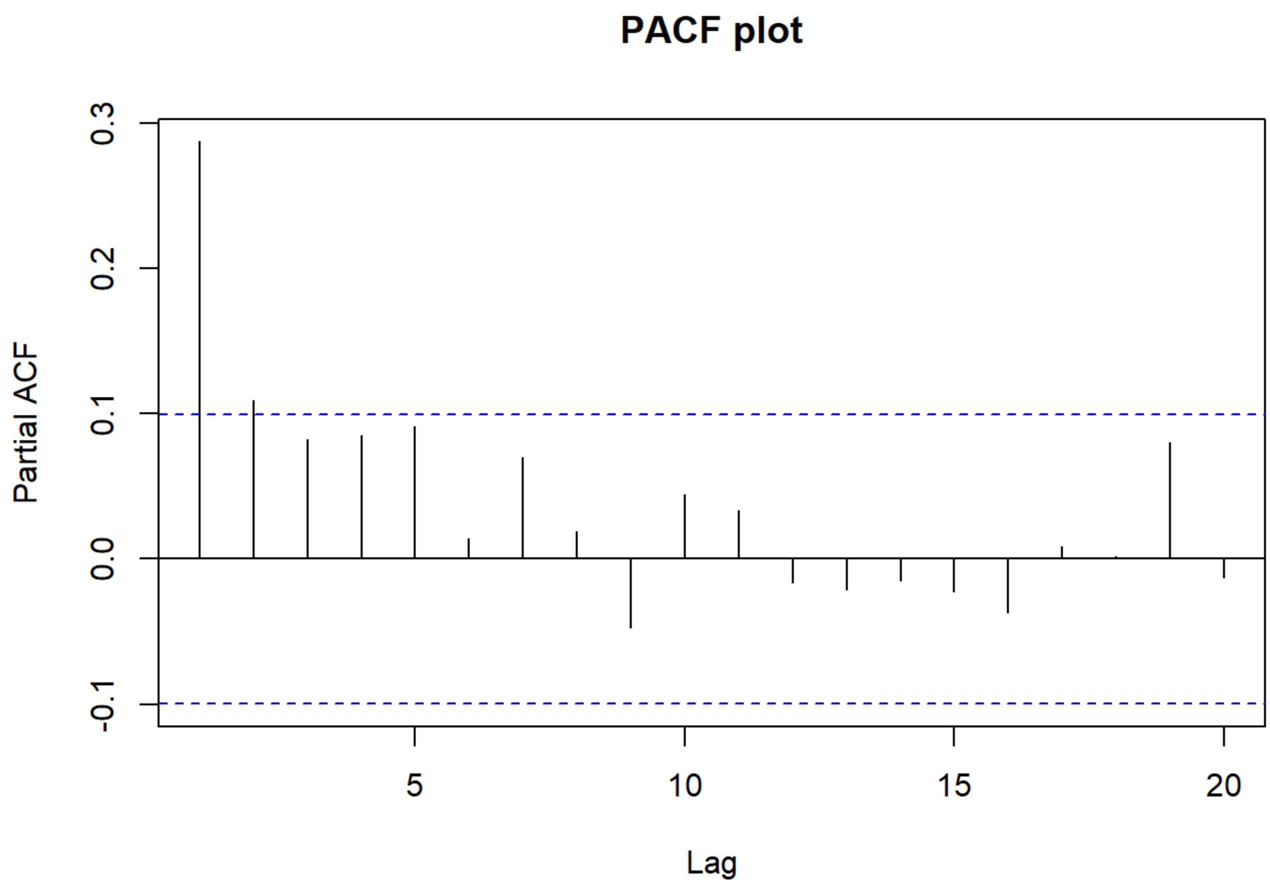
```
par(mfrow=c(1,1))
acf(diff(data_germany$Unemployment_Rates), lag.max = 20, main = "ACF plot")
```




p can be taken as 0/1, based on the no. of significant lags.

## PACF plot:

```
par(mfrow=c(1,1))
pacf(diff(data_germany$Unemployment_Rates), lag.max = 20, main = "PACF plot")
```




q can be taken as 0/1/2/3/4, based on the no. of significant lags.

# Fitting the model that we tested before - on the actual data:

```
est_actual=arima(data_germany$Unemployment_Rates, order=c(1,2,2), method = "ML")
summary(est_actual)
```

```
##
## Call:
## arima(x = data_germany$Unemployment_Rates, order = c(1, 2, 2), method = "ML")
##
## Coefficients:
##          ar1      ma1     ma2
##       0.8061  -1.6061  0.6109
## s.e.  0.1121   0.1448  0.1407
##
## sigma^2 estimated as 1.883e-06:  log likelihood = 1990.12,  aic = -3972.24
##
## Training set error measures:
##                         ME        RMSE          MAE         MPE     MAPE
## Training set -6.588141e-05 0.001368594 0.0007248823 -0.04392085 0.908913
##                   MASE       ACF1
## Training set 0.9887087 0.03124127
```

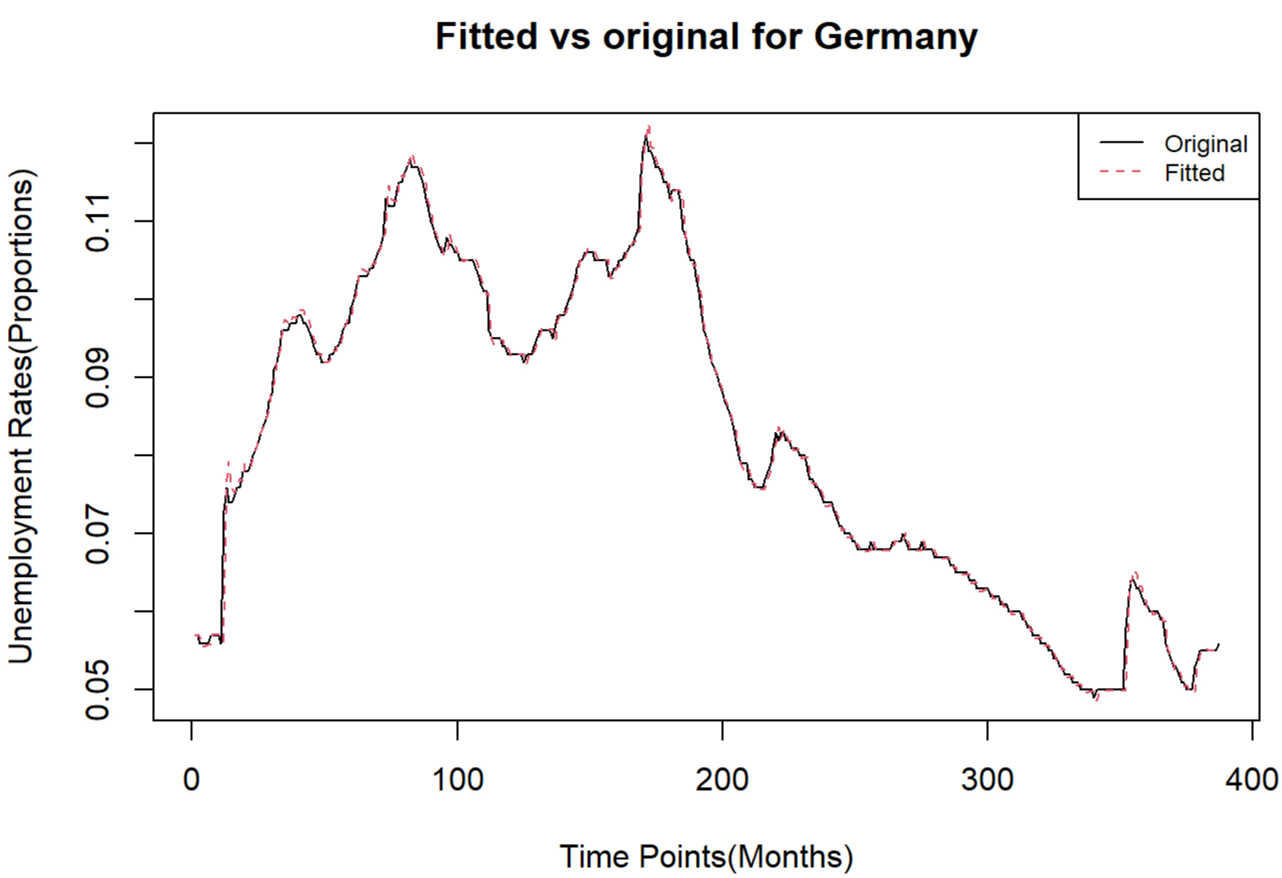
# Test of significance of individual coefficients:

```
suppressWarnings(library(lmtest))
coeftest(est_actual)
```

```
##
## z test of coefficients:
##
##      Estimate Std. Error  z value  Pr(>|z|)
## ar1  0.80605    0.11207   7.1926 6.357e-13 ***
## ma1 -1.60607    0.14482 -11.0902 < 2.2e-16 ***
## ma2  0.61088    0.14065   4.3432 1.404e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

# Plot of Fitted vs Original on the actual data:

```
res=residuals(est_actual)
data_fit=data_germany$Unemployment_Rates-res
ts.plot(data_germany$Unemployment_Rates, type="l", xlab="Time Points(Months)", ylab="Unemployment
Rates(Proportions)", main="Fitted vs original for Germany")
points(data_fit, type="l", col=2, lty=2)
legend("topright",c("Original","Fitted"), col=c(1,2), lty=c(1,2), cex=0.75)
```



# Obtaining prediction of Unemployment rate for May 2023:

```
future_unemp_pred=predict(est_actual, n.ahead=2, se.fit=FALSE, method="ML")
print(as.vector(future_unemp_pred)[2])
```

```
## [1] 0.0562936
```

# Upper & Lower limits (95% C.I.s):

```
upper=as.vector(future_unemp_pred)+(1.96*(sqrt(est_actual$sigma2)))
lower=as.vector(future_unemp_pred)-(1.96*(sqrt(est_actual$sigma2)))
```

# Upper limit for May 2023 forecast:

```
print(as.vector(upper)[2])
```

```
## [1] 0.05898299
```

# Lower limit for May 2023 forecast:

```
print(as.vector(lower)[2])
```

```
## [1] 0.05360421
```

## May,23 forecast - 5.629 %

## Upper & Lower limits - (5.36 %, 5.898 %)