```r
#Univariate Forecasting in R
#Justin S. Eloriaga

#Installing the Required Packages

install.packages("tidyverse")
install.packages("urca")
install.packages("forecast")
install.packages("tseries")
install.packages("TSstudio")
help(TSstudio)

#Calling the Required Packages
library(tidyverse)
library(forecast)
library(tseries)
library(urca)
library(TSstudio)

#-----Loading the Dataset
inflation <- read_csv(file.choose())
head(inflation)
nrow(inflation)

#-----Declaring the Time Series Object
inf <- ts(inflation$Rate, start = c(2000,1,5), frequency = 12)

#Plotting the Time Series Object
autoplot(inf) + ggtitle("Inflation Rate (Philippines), January 2000 to April 2020") + labs(x = "Time", y = "Inflation Rate")
ts_plot(inf, line.mode = "lines", title = "Inflation Rate (Philippines), January 2000 to April 2020")

#Generating some Summary Statistics

summary(inf)

#-----Looking at the ACF and PACF

ggAcf(inf) + ggtitle("ACF of Inflation")
ggPacf(inf) + ggtitle("PACF of Inflation")

#Differencing the Series

dinf <- diff(inf)

#Looking at the ACF and PACF of the Differenced Series

ggAcf(dinf) + ggtitle("ACF of Inflation (Differenced)")
ggPacf(dinf) + ggtitle("PACF of Inflation (Differenced)")

#Graphing Levels and Differenced

combo <- cbind(inf, dinf)
autoplot(combo, facets = TRUE) + ggtitle("Inflation (Rate (Philippines), Level and Difference") + labs(y ="Rate")

#-----Decomposing the Series

ts_decompose(inf, type = "additive", showline = TRUE)

#-----Testing for Non-Stationarity

#Using the Augmented Dickey Fuller Test
```

```
adf.test(inf)
adf.test(inf, k = 2)
adf.test(inf, k = 1)
adf.test(dinf)

#Using the Phillips Perron Test

pp.test(inf)
pp.test(dinf)

#Using the KPSS Test

kpss.test(inf)
kpss.test(dinf)

#----- In Sample Forecasting and Validation

#Partition the data into test data and training data

split_inf <- ts_split(inf, sample.out = 12)

training <- split_inf$train
testing <- split_inf$test

length(training)
length(testing)

#Using an ARIMA Diagnostic Plot on the Training Dataset

arima_diag(training)

#Trying out some Models

#For Model 1
arima211 <- arima(training, order = c(2,1,1))
autoplot(arima211)
check_res(arima211)

#For Model 2
sarima2111 <- arima(training, order = c(2,1,1), seasonal = list(order = c(1,0,0)))
autoplot(sarima2111)
check_res(sarima2111)

#For Model 3
auto <- auto.arima(training, seasonal = TRUE)
auto #We obtained a SARIMA(2,1,0)(2,0,1)
autoplot(auto)
check_res(auto)

#Forecasting Values and Diagnostics

#For Model 1
fcast1 <- forecast(arima211, h = 12)
test_forecast(actual = inf, forecast.obj = fcast1, test = testing)
accuracy(fcast1,testing)

#For Model 2
fcast2 <- forecast(sarima2111, h = 12)
test_forecast(actual = inf, forecast.obj = fcast2, test = testing)
accuracy(fcast2,testing)

#For Model 3
fcasta <- forecast(auto, h = 12)
```

```
test_forecast(actual = inf, forecast.obj = fcasta, test = testing)
accuracy(fcasta,testing)

#Graphing All Models in One

# Defining the models and their arguments
methods <- list(Model1 = list(method = "arima",
                 method_arg = list(order = c(2,1,1)),
                 notes = "ARIMA(2,1,1)"),
         Model2 = list(method = "arima",
                 method_arg = list(order = c(2,1,1),
                            seasonal = list(order = c(1,0,0))),
                 notes = "SARIMA(2,1,1)(1,0,0)"),
         Model3 = list(method = "arima",
                 method_arg = list(order = c(2,1,0),
                            seasonal = list(order = c(2,0,1))),
                 notes = "SARIMA(2,1,0)(2,0,1)"))

# Training the models with backtesting
md <- train_model(input = inf,
          methods = methods,
          train_method = list(partitions = 2,
                      sample.out = 12,
                      space = 3),
          horizon = 12,
          error = "RMSE")

# Plot the models performance on the testing partitions
plot_model(model.obj = md)

#------ Out of Sample Forecasting

#Building the Final Forecast Model
finalfit <- auto.arima(inf, seasonal = TRUE)
autoplot(finalfit)
check_res(finalfit)

#Generating the Forecast
fcastf <- forecast(inf, model = finalfit, h = 4)
plot_forecast(fcastf)
summary(fcastf)
```