Vidyavardhini's College of Engineering and Technology
Department of Artificial Intelligence & Data Science

BARI ANKIT (56)

Experiment No.5
Implementation of ARIMA model in python / R.
Date of Performance:
Date of Submission:

Experiment No 5

Aim- Implementation of ARIMA model in **R** Programming.

Objective- To Understand use of Auto-Regression Integrated Moving Average Time Series Model

Theory-

In R programming , data analysis and visualization is so easy to learn the behavior of the data. Moreover, the **R** language is used mostly in the data science field after Python. Time series analysis is a type of analysis of data used to check the behavior of data over a period of time. The data is collected over time sequentially by the **ts()** function along with some parameters. It helps in analyzing the pattern of the data over a graph. There are many techniques used to forecast the time series object over the plot graph but the **ARIMA model** is the most widely used approach out of them.

Time Series Forecasting

Time series forecasting is a process of predicting future values with the help of some statistical tools and methods used on a data set with historical data. Some of the applications of time series forecasting are:

- Predicting stock prices
- Forecast weather
- Forecast the sales of a product

ARIMA model

ARIMA stands for AutoRegressive Integrated Moving Average and is specified by three order parameters: *(p, d, q).*

- **AR(p) Autoregression:** A regression model that utilizes the dependent relationship between a current observation and observations over a previous period.An auto regressive *(AR(p))* component refers to the use of past values in the regression equation for the time series.
- **I(d) Integration:** Uses differencing of observations (subtracting an observation from observation at the previous time step) in order to make the time series stationary. Differencing involves the subtraction of the current values of a series with its previous

values d number of times.

1. Load the data set after installing the package forecast.
2. The Steps of Pre-processing are done, which creates a separate time-series or timestamp.
3. Making Time-series stationary and check the required transformations.
4. The difference value 'd' will be performed.
5. The core important step in ARIMA is plotting ACF and PACF.
6. Determine the two parameters p and q from the plots.
7. The previously created value fits the Aroma model and predicts the future values. The Fitting Process is also named as Box-Jenkins Method.
8. Doing Validation.auto. Arima() function is used for automatic prediction and ARIMA Models. This function uses unit root tests, minimization of the AIC and MLE to obtain an ARIMA model.

To make the series stationary, we need to differentiate a previous value from the current value.

d= pval-cval, if the value is already stationary the d=O.

predict() - Used to predict the model based on the results of the various fitting model used.

Implementation of ARIMA model in R

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Generate a time series dataset (e.g., monthly data)
np.random.seed(42)
```

```
date_range = pd.date_range(start='2020-01-01', periods=100, freq='M')
data = pd.Series(np.random.randn(100).cumsum(), index=date_range)

# Plot the time series data
plt.figure(figsize=(12, 6))
plt.plot(data, label='Time Series Data', color='blue')
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()

# ACF and PACF plots to determine p and q
plot_acf(data, lags=20)
plt.title('Autocorrelation Function (ACF)')
plt.show()

plot_pacf(data, lags=20)
plt.title('Partial Autocorrelation Function (PACF)')
plt.show()

# Fit an ARIMA model (You may want to change order based on ACF and PACF plots)
model = ARIMA(data, order=(1, 1, 1))
model_fit = model.fit()

# Summary of the model
print(model_fit.summary())
```

```
# Plot residuals
plt.figure(figsize=(12, 6))
plt.plot(model_fit.resid)
plt.title('Residuals of the Model')
plt.axhline(0, color='red', linestyle='--')
plt.show()

# Forecasting
forecast = model_fit.forecast(steps=10)
plt.figure(figsize=(12, 6))
plt.plot(data, label='Historical Data', color='blue')
plt.plot(pd.date_range(start=data.index[-1] + pd.offsets.MonthBegin(), periods=10, freq='M'),
forecast, label='Forecast', color='red')
plt.title('ARIMA Forecast')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.grid(True)
plt.show()
```

Output:

```
# Load necessary libraries
library(forecast)
library(ggplot2)

# Generate a time series dataset (e.g., monthly data)
set.seed(42)
data <- ts(rnorm(100, mean=10, sd=2), frequency=12, start=c(2020, 1))

# Plot the time series data
```

```
autoplot(data) +
  ggtitle('Time Series Data') +
  xlab('Year') +
  ylab('Value')

# ACF and PACF plots to determine p and q
Acf(data, lag.max = 20, main='Autocorrelation Function (ACF)')
Pacf(data, lag.max = 20, main='Partial Autocorrelation Function (PACF)')

# Fit an ARIMA model (You may want to change order based on ACF and PACF plots)
model <- auto.arima(data)

# Summary of the model
summary(model)

# Residual diagnostics
checkresiduals(model)

# Forecasting
forecasted_values <- forecast(model, h=10)
autoplot(forecasted_values) +
  ggtitle('ARIMA Forecast') +
  xlab('Year') +
  ylab('Value')
```

Conclusion:-

In this R implementation, we successfully configured and analyzed the ARIMA model for time series forecasting. After generating the dataset, we utilized ACF and PACF plots for identifying parameters. The auto.arima function facilitated the fitting process, yielding a concise summary of the model's performance. Finally, the forecasts provided valuable insights into expected future values, demonstrating the effectiveness of ARIMA in time series forecasting. This methodology is crucial for making informed decisions based on historical trends and patterns.