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

BARI ANKIT (56)

Experiment No.4
Time Series Analysis in Python/R.
Date of Performance:
Date of Submission:

Experiment No- 4

Aim :- Implement Time Series Analysis for rainfall in R Programming. **Objective:-**

To understand the use of time series models for prediction.

Description:-

- Time series analysis is a specific way of analyzing a sequence of data points collected over an interval of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points randomly.

The basic syntax for ts() function in time series analysis is -

- timeseries.object.name <- ts(data, start, end, frequency)

Following is the description of the parameters used -

- data is a vector or matrix containing the values used in the time series.
- start specifies the start time for the first observation in time series.
- end specifies the end time for the last observation in time series.
- frequency specifies the number of observations per unit time.

- Except the parameter "data" all other parameters are optional.

Different Time Intervals

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

The value of the frequency parameter in the ts() function decides the time intervals at which the data points are measured. A value of 12 indicates that the time series is for 12 months. Other values and its meaning is as below -

- frequency = 12 pegs the data points for every month of a year.
- frequency = 4 pegs the data points for every quarter of a year.
- frequency = 6 pegs the data points for every 10 minutes of an hour.
- frequency = 24*6 pegs the data points for every 10 minutes of a day.

Program :-

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm

# 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()
# Decompose the time series into trend, seasonality, and residuals
decomposition = sm.tsa.seasonal_decompose(data, model='additive')
fig = decomposition.plot()
fig.set_size_inches(12, 8)
```

```
plt.show()


# Fit an ARIMA model
from statsmodels.tsa.arima.model import ARIMA


# ARIMA Model Fitting
model = ARIMA(data, order=(1, 1, 1))  # Adjust orders as needed
model_fit = model.fit()


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


# Plot the forecast
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 required libraries
library(ggplot2)
library(forecast)


# 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')


# Decompose the time series into its components
decomposed_data <- decompose(data)
autoplot(decomposed_data) +
  ggtitle('Decomposition of Time Series') +
  xlab('Year') +
  ylab('Value')


# Fit an ARIMA model
model <- auto.arima(data)


# Summary of the model
```

```
summary(model)


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

Conclusion :-

The R implementation of time series analysis effectively showcases how to utilize the forecast and ggplot2 libraries for visualizing and modeling time-based data. Decomposing the time series allows for a more profound understanding of its components, helping to identify trends and seasonal patterns.

By using the ARIMA model, we can create reliable forecasts that can guide strategic decisions across various domains such as finance, supply chain management, and public policy. Effective time series analysis facilitates better preparation for future trends and unforeseen events, solidifying its importance in data-driven forecasting.