Capstone Project: BAN 693

FORECASTING AMAZON STOCK PRICE

Prepared by:
1. Reena Namani (NetId – AH7791)
2. Bhargav Silaparasetty (NetId – RM2790)

Instructor name: Jiming Wu

Affiliations: California State University - East Bay

<u>Summary:</u>

The objective of the project is to forecast stock price of AMAZONusing the historical dataset available, which contains information on Amazon’s Stock price, that encompasses the time period from May 1997 to September 2023. The dataset will consist of several columns, each of which offers details regarding its stock performance on a daily basis such as opening price, closing price, day high and day low.

We have used regression analysis, time series analysis, and predictive modeling to analyze the historical dataset of Amazon's stock price and forecast its future values.

We have also used different models such as two-level forecasting, and the advanced Exponential Smoothing & Arima model to improve the accuracy of the forecast. By comparing the RMSE and MAPE values of each model, we have selected the best model for forecasting future stock prices.

The model with the lowest RMSE and MAPE values is generally considered the best fit for forecasting future stock prices.

It is important to note that no forecasting model can provide perfectly accurate predictions, and some degree of error is inevitable. However, by selecting the model with the lowest RMSE and MAPE values, you can minimize the error and increase the likelihood of making informed investment decisions.

The forecasting results can provide valuable insights into the trends and patterns of Amazon's stock performance and help investors and analysts make informed decisions regarding their investment in the stock. Visualizations such as line graphs and charts can also help in understanding the data and communicate the findings effectively.

Introduction

Amazon, one of the biggest and most prosperous technological businesses in the world, is best recognized for their e-commerce & cloud business. Starting out in 1994, it has expanded quickly, entering a variety of new markets and sectors, such as cloud computing, e-commerce & digital content.

The popularity of Amazon's stock, or AMZN, among investors is a result of the firm's solid financial results and its position as a pioneer in the technological sector. The corporation significantly invests in emerging technology, and its advertising division generates most of its revenue. Even though it is prone to market fluctuations, Amazon's stock is regarded as a solid long-term investment because of the company's financial health and dominant market share.

For this project we got the data from Kaggle &we have conducted a thorough analysis of Amazon's stock price using various statistical techniques such as regression analysis, time series analysis, and predictive modeling.

***Regression analysis*:** Regression analysis is a statistical technique used to examine the relationship between two or more variables. In finance and investing, regression analysis is commonly used to analyze the relationship between a stock's price and other factors that may influence it, such as company earnings, industry trends, or macroeconomic indicators. Regression analysis can help investors and analysts understand how changes in these factors are likely to impact the stock's price.

***Time series analysis*:** Time series analysis is a statistical technique used to analyze data that is collected over time, such as stock prices or economic indicators. The goal of time series analysis is to identify patterns or trends in the data and make predictions about future values based on those patterns. Time series analysis involves techniques such as trend analysis, seasonal analysis, and forecasting.

Predictive modeling**:** Predictive modeling is a statistical technique used to make predictions about future events based on historical data. In finance and investing, predictive modeling is commonly used to forecast future stock prices or other financial metrics such as earnings or revenue. Predictive modeling involves using statistical algorithms and techniques to analyze historical data and identify patterns or trends that can be used to make predictions about the future.

We have also used different models such as two-level forecasting, advanced Exponential Smoothing, and ARIMA models can be used to improve the accuracy of forecasting future stock prices.

Two-Level Forecasting**:** Two-level forecasting is a statistical technique that uses a combination of regression analysis and time series analysis to forecast future values. The technique involves first using regression analysis to model the relationship between the stock price and external factors that may influence it, such as earnings, industry trends, or macroeconomic indicators. Next, the time series analysis is used to model the time-dependent factors that may impact the stock price. By combining these two models, two-level forecasting can provide a more accurate forecast of future stock prices.

Advanced Exponential Smoothing**:** Exponential smoothing is a time series analysis technique used to model time-dependent data, such as stock prices. Advanced Exponential Smoothing techniques build on this basic method by introducing additional parameters that can help capture more complex patterns and trends in the data. For example, Holt's Exponential Smoothing adds a trend component to the model, while Winter's Exponential Smoothing adds a seasonal component. By adjusting these parameters, advanced Exponential Smoothing techniques can provide more accurate forecasts of future stock prices.

ARIMA Models**:** ARIMA (Autoregressive Integrated Moving Average) models are another type of time series analysis technique that can be used to model the time-dependent factors that impact stock prices. ARIMA models are particularly useful when the data exhibits trends or seasonality. ARIMA models involve selecting the appropriate parameters for the autoregressive

(AR), integrated (I), and moving average (MA) components of the model. By selecting the appropriate parameters and fitting the model to historical data, ARIMA models can provide more accurate forecasts of future stock prices.

After creating these models, the accuracy of each model is assessed by comparing the Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE) values of each model. RMSE measures the difference between predicted and actual values, while MAPE measures the percentage difference between predicted and actual values. The model with the lowest RMSE and MAPE values is generally considered the best fit for forecasting future stock prices.

In summary, different models such as two-level forecasting, advanced Exponential Smoothing, and ARIMA models can be used to improve the accuracy of forecasting future stock prices. By comparing the RMSE and MAPE values of each model, wehave selected the best model for forecasting future stock prices.

Eight steps of forecasting

The above summary describes a project for forecasting the stock price of Amazon using historical data. Let's apply the eight steps of forecasting to this project:

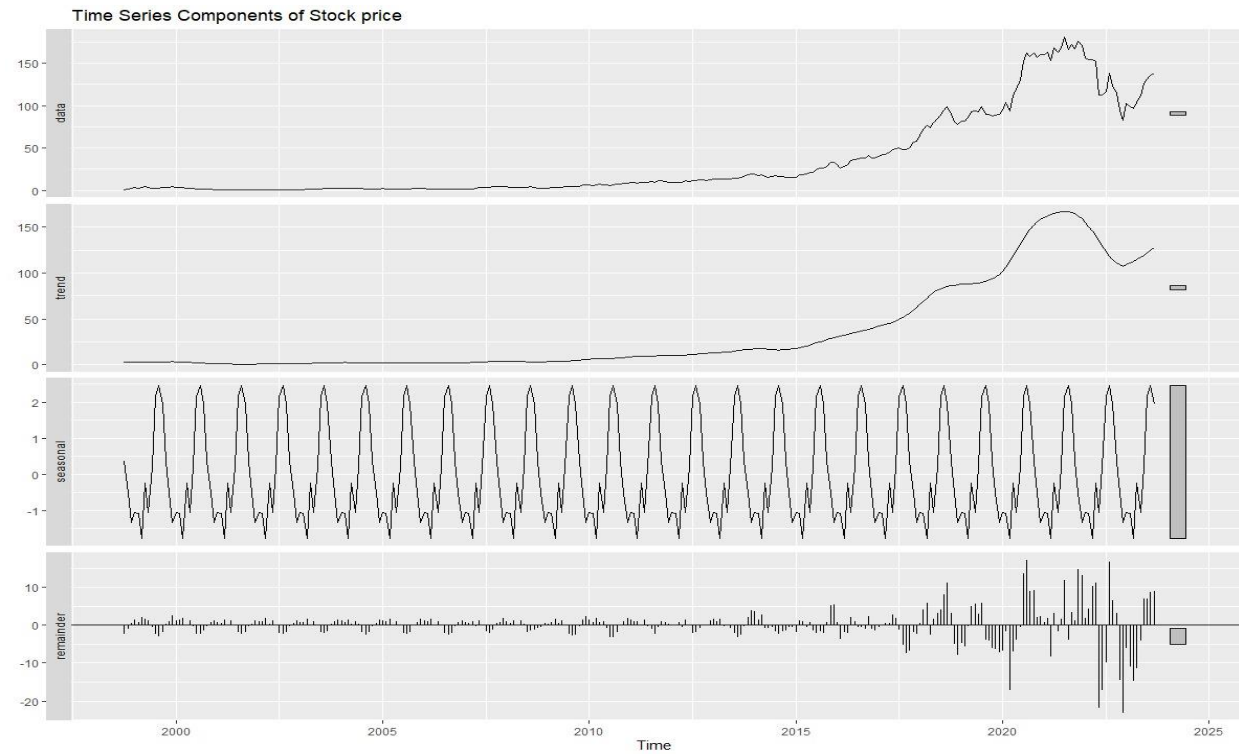
Step 1: Define the Goal - The goal of the project is to forecast the stock price of Amazon using historical data and various statistical techniques. The forecasting models developed for this project were done in R language.

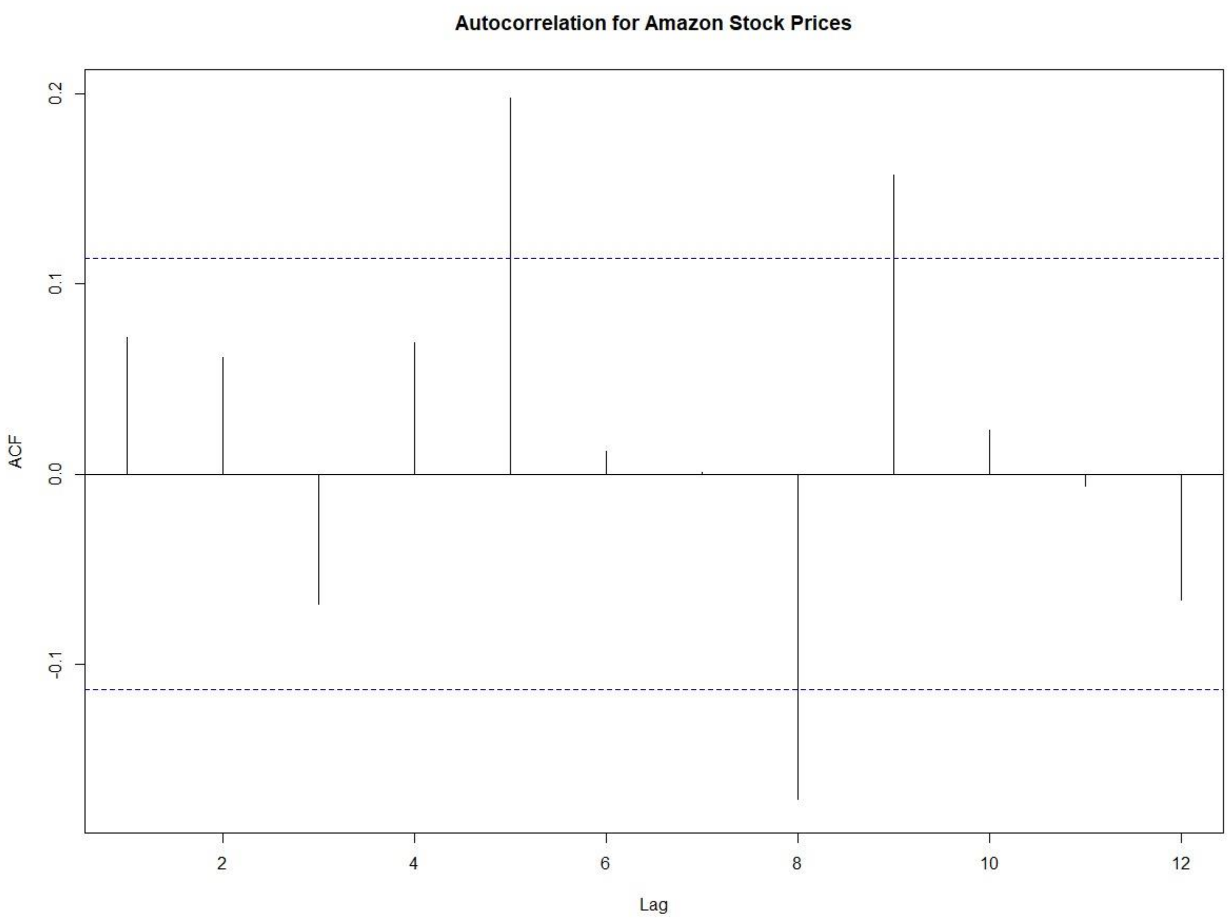
Step 2: Get the Data -We have obtained the data from the Kaggle, and it deals with historical data on Amazon's stock price, containing information on the stock performance on a monthly basis has been obtained. We have data such as monthly high, monthly low, average stock price for the month. The data spans from May 1997 to the September 2023

The first column is the 'date' column, which provides the date on which the stock price was recorded. The second column is the 'price' column, which displays the opening price of the stock on the given date. The third column is the 'high' column, which represents the highest price at which the stock was traded during the day. The fourth column is the 'low' column, which displays the lowest price at which the stock was traded during the day.

Using the day high and day low, we calculated day average ((day high + day low)/2). We then calculated monthly average by taking average of the day averages we have. Also we set the date as 15th of the month, so that we can forecast on monthly basis going forward.

Step 3: Explore and visualize the series - The project involves exploring and visualizing the historical data to identify various time series components of trends, patterns, and outliers.


Time Series Components of Stock price
data
trend
seasonal
remainder
Time


Autocorrelation for Amazon Stock Prices
ACF
Lag

***Step 4: Data pre-processing* -** The historical data may require pre-processing steps such as handling missing data, smoothing, and scaling.

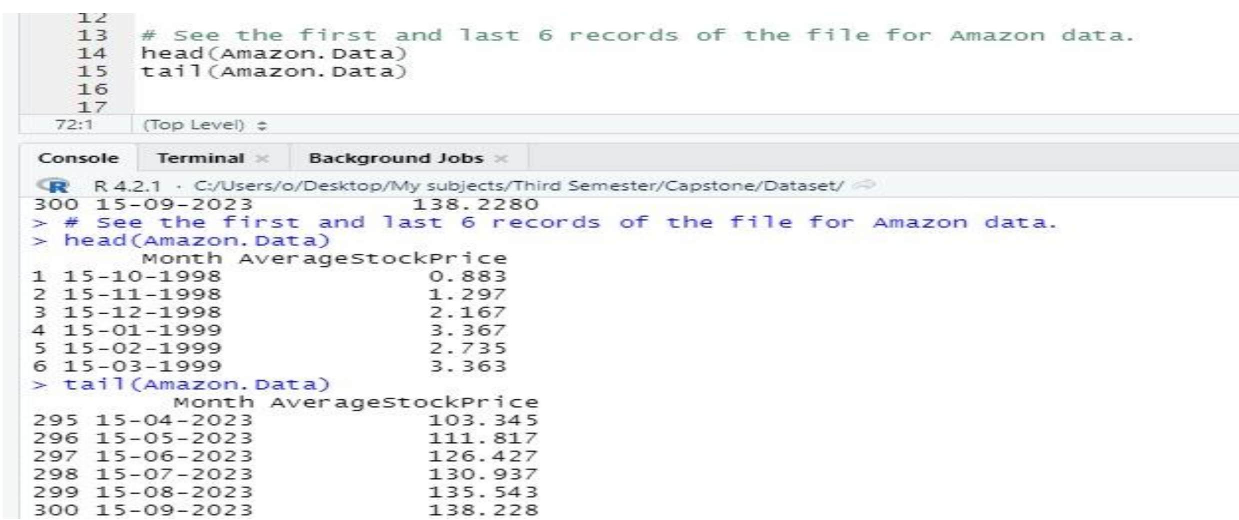
***Step 5: Partition Series* -** The dataset may be partitioned into training and test sets for modelling and evaluating the forecast accuracy.In this model, the partition set was 80% (240 months) for training data & 20% (60 months) for validation data.

***Step 6 & 7: Apply forecasting and compare performances* -** The projectinvolves applying various forecasting models such as regression analysis, time series analysis, and predictive modelling. The performance of each model can be compared based on metrics such as RMSE and MAPE.

Introductory R code to start forecasting in R environment.

library(forecast)
library(zoo)

 # See the first and last 6 records of the dataset.
head(Amazon.Data)
tail(Amazon.Data)



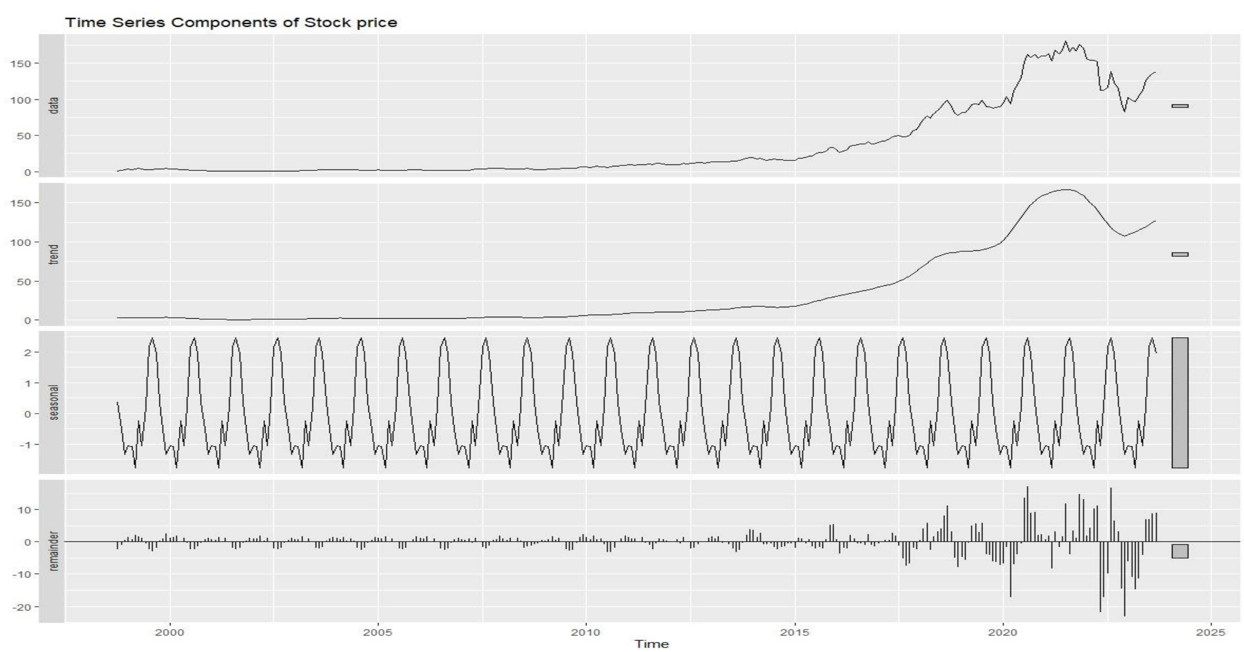
```
# See the first and last 6 records of the file for Amazon data.
head(Amazon.Data)
tail(Amazon.Data)

72:1   (Top Level) ÷
Console   Terminal ×   Background Jobs ×
R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/
300 15-09-2023          138.2280
> # See the first and last 6 records of the file for Amazon data.
> head(Amazon.Data)
       Month AverageStockPrice
1 15-10-1998             0.883
2 15-11-1998             1.297
3 15-12-1998             2.167
4 15-01-1999             3.367
5 15-02-1999             2.735
6 15-03-1999             3.363
> tail(Amazon.Data)
         Month AverageStockPrice
295 15-04-2023           103.345
296 15-05-2023           111.817
297 15-06-2023           126.427
298 15-07-2023           130.937
299 15-08-2023           135.543
300 15-09-2023           138.228
```

##ts() function is used to create time series data set.

#Amazon.ts <- ts(Amazon.Data$Average.Stock.Price,start = c(1998,10), end = c(2023,9),freq=12)

```
> Amazon.ts <- ts(Amazon.Data$AverageStockPrice,
+                 start = c(1998, 10), end = c(2023,9) , freq = 12)
> Amazon.ts
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1998                                                                                    0.8830   1.2970   2.1670
1999   3.3670   2.7350   3.3630   4.5530   3.3610   2.7860   3.0010   2.6400   3.2460   3.9800   3.8100   4.6080
2000   3.4190   3.6190   3.3200   2.8250   2.6720   2.2800   1.8370   1.8050   2.0970   1.5310   1.5210   1.0570
2001   0.8900   0.6810   0.5490   0.6700   0.7880   0.7120   0.7390   0.5250   0.3720   0.3810   0.4360   0.5580
2002   0.6010   0.6530   0.7660   0.7250   0.9060   0.8660   0.7390   0.7230   0.8140   0.9160   1.0650   1.0940
2003   1.0700   1.0700   1.2460   1.3130   1.5960   1.7640   1.9350   2.1000   2.3670   2.7950   2.6330   2.5650
2004   2.6930   2.2730   2.1200   2.3250   2.1610   2.5360   2.3100   1.8960   2.0290   1.9180   1.9140   2.0300
2005   2.1220   1.8080   1.7260   1.6840   1.7290   1.7340   1.7590   1.7560   1.8740   2.1820   2.2280   2.4400
2006   2.2610   1.9380   1.8220   1.8170   1.7050   1.7590   1.6700   1.3850   1.5740   1.6930   2.0170   1.9690
2007   1.8790   1.9710   1.9300   2.3110   3.2180   3.5180   3.7230   3.8400   4.4030   4.5700   4.1510   4.5790
2008   4.0860   3.6300   3.4220   3.8450   3.8680   4.0000   3.6160   4.1080   3.8150   2.8300   2.2270   2.4660
2009   2.6070   3.1600   3.4210   3.9230   3.8850   4.1600   4.1700   4.2000   4.3250   5.0550   6.4030   6.7610
2010   6.3860   5.8890   6.5240   7.0520   6.3810   6.0980   5.8380   6.3000   7.3490   8.0210   8.4180   8.9460
2011   9.1105   9.0470   8.4350   9.2000   9.9050   9.5820  10.7490   9.9670  11.1370  11.3170  10.2880   9.2030
2012   9.2250   9.1280   9.4540   9.7580  11.0610  10.9440  11.1870  11.9470  12.7850  12.2420  11.7020  12.6380
2013  13.4330  13.1930  13.2600  13.1560  13.1270  13.6970  14.9160  14.5670  15.2130  16.2610  18.2090  19.5730
2014  19.7600  17.6830  18.1760  16.0620  15.1240  16.2250  16.9570  16.3670  16.6060  15.4100  15.8930  15.4730
2015  15.1450  18.7420  18.7800  19.7270  21.3630  21.6300  23.9790  25.9700  26.0470  28.2300  32.8360  33.4590
2016  30.0780  26.5250  28.5170  30.6690  34.8280  35.7970  37.0550  38.2120  38.3890  41.2650  38.2090  38.1760
2017  40.2550  41.7160  42.6720  45.2250  48.0380  49.5150  50.3940  48.5780  48.4760  50.0090  56.8970  58.4640
2018  65.2190  72.1620  76.8450  73.5120  79.6010  84.9070  89.1620  94.7400  98.2700  89.4120  81.1700  78.2450
2019  81.7500  81.3060  86.0660  93.1330  93.4730  92.6200  98.1500  89.7200  89.9250  87.5070  88.7130  89.2740
2020  94.2090 103.4830  93.3880 111.0700 119.5840 130.5250 152.4850 162.3110 158.2720 161.7300 157.0490 159.8240
2021 160.3220 163.2500 153.5360 167.5910 162.6750 168.1900 180.5600 165.4550 171.4710 166.6350 176.2980 170.9810
2022 155.6440 153.7990 154.2340 151.7980 112.3100 112.5740 116.4810 137.8070 123.1970 114.3210  94.2520  83.2600
2023 102.2400  99.0360  96.3980 103.3450 111.8170 126.4270 130.9370 135.5430 138.2280
```

#stl() component is used to plot time series components(seasonality, trend and level) of the original data:
#(Visualization)
Stock.stl<- stl(Amazon.ts, s.window="periodic")
autoplot(Stock.stl, main="Time Series Components of Stock price")



 # Use Arima() function to fit AR(1) model.

 # The ARIMA model of order = c(1,0,0) gives an AR(1) model.
Stock.ar1<- Arima(Amazon.ts, order = c(1,0,0))
summary(Stock.ar1)

```
# Use Arima() function to fit AR(1) model.
# The ARIMA model of order = c(1,0,0) gives an AR(1) model.
# ARIMA (1,0,0) model is an autoregressive model of order 1 without differencing and without average moving component.
Stock.ar1<- Arima(Amazon.ts, order = c(1,0,0))
summary(Stock.ar1)
```

```
Console   Terminal   Background Jobs
R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/
> # The ARIMA model of order = c(1,0,0) gives an AR(1) model.
> # ARIMA (1,0,0) model is an autoregressive model of order 1 without differencing and without average moving component.
> Stock.ar1<- Arima(Amazon.ts, order = c(1,0,0))
> summary(Stock.ar1)
Series: Amazon.ts
ARIMA(1,0,0) with non-zero mean

Coefficients:
         ar1     mean
      0.9971  58.3698
s.e.  0.0032  53.1644

sigma^2 = 23.51:  log likelihood = -900.85
AIC=1807.7   AICc=1807.78   BIC=1818.81

Training set error measures:
                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 0.3742477 4.832293 2.191709 -5.561658 11.73532 0.2181803 0.07629686
```

The above code Stock.ar1<- Arima(Amazon.ts, order = c(1,0,0)) fits an ARIMA (Autoregressive Integrated Moving Average) model of order (1,0,0) to the Amazon.ts time series data. This model is an AR(1) model, which means it has a single autoregressive term and no moving average term.

The summary(Stock.ar1) command provides a summary of the model's coefficients and other key statistics.

The output shows that the AR(1) model has an autoregressive coefficient (AR1) of 0.9971 and a mean of 58.3698. The standard error of the coefficient estimate (s.e.) is 0.0032, and the sigma squared is estimated to be 23.51. The log-likelihood of the model is -900.85, and the Akaike information criterion (AIC), corrected AIC (AICc), and Bayesian information criterion (BIC) values are also provided.

###Checking predictability using hypothesis testing method:

#Predictability test approach-1

 # Apply z-test to test the null hypothesis that beta
 # coefficient of AR(1) is equal to 1.

```
ar1 <- 0.9971
s.e.<- 0.0032
null_mean<- 1
alpha <- 0.05
z.stat<- (ar1-null_mean)/s.e.
z.stat
p.value<- pnorm(z.stat)
p.value
if (p.value<alpha) {
  "Reject null hypothesis"
} else {
  "Accept null hypothesis"
}
```

```
# Apply z-test to test the null hypothesis that beta
# coefficient of AR(1) is equal to 1.
ar1 <- 0.9971
s.e. <- 0.0032
null_mean <- 1
alpha <- 0.05
z.stat <- (ar1-null_mean)/s.e.
z.stat
p.value <- pnorm(z.stat)
p.value
if (p.value<alpha) {
  "Reject null hypothesis"
} else {
  "Accept null hypothesis"
}
```

Console Terminal Background Jobs

R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/

```
Training set 0.3742477 4.832293 2.191709 -5.561658 11.73532 0.2181803 0.07629686
> # Apply z-test to test the null hypothesis that beta
> # coefficient of AR(1) is equal to 1.
> ar1 <- 0.9971
> s.e. <- 0.0032
> null_mean <- 1
> alpha <- 0.05
> z.stat <- (ar1-null_mean)/s.e.
> z.stat
[1] -0.90625
> p.value <- pnorm(z.stat)
> p.value
[1] 0.1824018
> if (p.value<alpha) {
+   "Reject null hypothesis"
+ } else {
+   "Accept null hypothesis"
+ }
[1] "Accept null hypothesis"
```

This code is performing a predictability test on the AR(1) model for Amazon stock price. The null hypothesis is that the beta coefficient of AR(1) is equal to 1, which means that the stock price is predictable based on its past values. The alternative hypothesis is that stock price is not predictable based on its past values and it is a random walk.

The code first sets the value of the AR(1) coefficient (ar1) to 0.9971 and the standard error (s.e.) to 0.0032. The null hypothesis is set to a value of 1. The significance level (alpha) is set to 0.05.

The z-statistic is calculated as (ar1-null_mean)/s.e., which gives a value of -0.90625. The p-value is then calculated using the pnorm() function, which gives a value of 0.1824.

The code then checks if the p-value is greater than the significance level (alpha). **Since the p-value is greater than the significance level, the null hypothesis is accepted, which means that the stock price is random and not predictable.**

#Predictability test approach-2

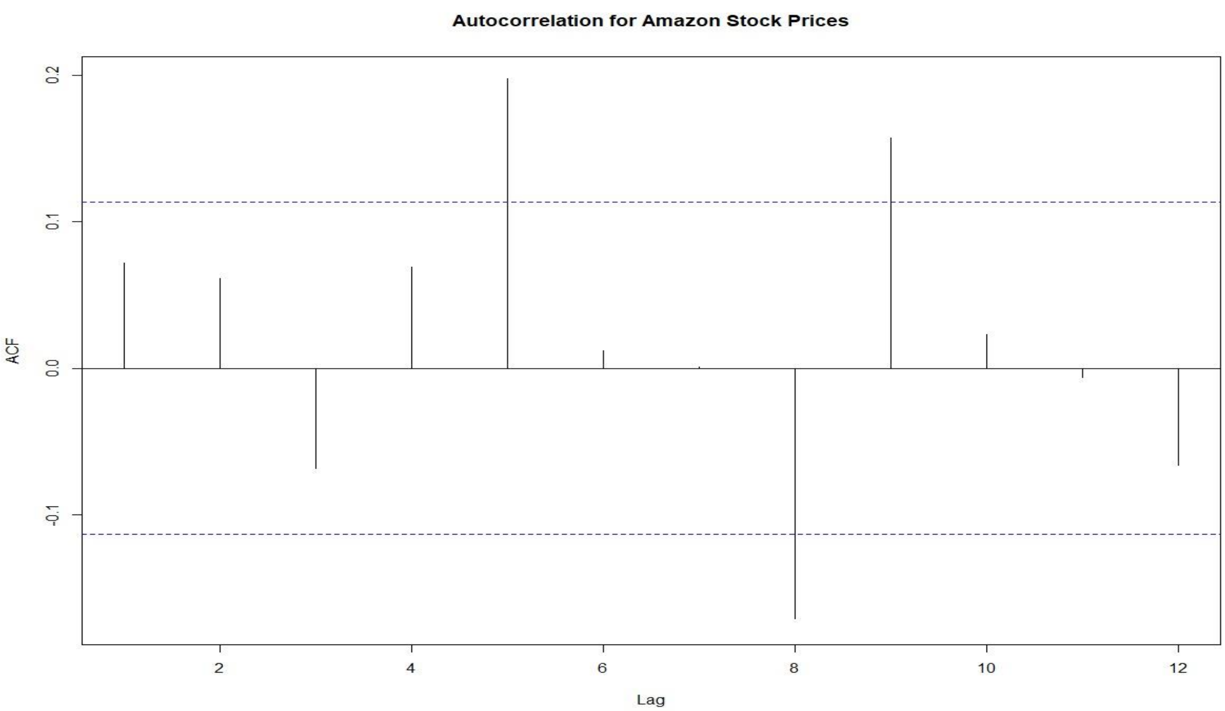
 # Create first difference of ClosePrice data using diff() function.
diff.stock.price<- diff(Amazon.ts, lag = 1)
diff.stock.price

```
#Predictability test approach-2

# Create first difference of ClosePrice data using diff() function.
diff.stock.price <- diff(Amazon.ts, lag = 1)
diff.stock.price
```

```
R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/
+ }
[1] "Accept null hypothesis"
> # Create first difference of ClosePrice data using diff() function.
> diff.stock.price <- diff(Amazon.ts, lag = 1)
> diff.stock.price
         Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
1998                                                                                              0.4140   0.8700
1999  1.2000  -0.6320   0.6280   1.1900  -1.1920  -0.5750   0.2150  -0.3610   0.6060   0.7340  -0.1700   0.7980
2000 -1.1890   0.2000  -0.2990  -0.4950  -0.1530  -0.3920  -0.4430  -0.0320   0.2920  -0.5660  -0.0100  -0.4640
2001 -0.1670  -0.2090  -0.1320   0.1210   0.1180  -0.0760   0.0270  -0.2140  -0.1530   0.0090   0.0550   0.1220
2002  0.0430   0.0520   0.1130  -0.0410   0.1810  -0.0400  -0.1270  -0.0160   0.0910   0.1020   0.1490   0.0290
2003 -0.0240   0.0000   0.1760   0.0670   0.2830   0.1680   0.1710   0.1650   0.2670   0.4280  -0.1620  -0.0680
2004  0.1280  -0.4200  -0.1530   0.2050  -0.1640   0.3750  -0.2260  -0.4140   0.1330  -0.1110  -0.0040   0.1160
2005  0.0920  -0.3140  -0.0820  -0.0420   0.0450   0.0050   0.0250  -0.0030   0.1180   0.3080   0.0460   0.2120
2006 -0.1790  -0.3230  -0.1160  -0.0050  -0.1120   0.0540  -0.0890  -0.2850   0.1890   0.1190   0.3240  -0.0480
2007 -0.0900   0.0920  -0.0410   0.3810   0.9070   0.3000   0.2050   0.1170   0.5630   0.1670  -0.4190   0.4280
2008 -0.4930  -0.4560  -0.2080   0.4230   0.0230   0.1320  -0.3840   0.4920  -0.2930  -0.9850  -0.6030   0.2390
2009  0.1410   0.5530   0.2610   0.5020  -0.0380   0.2750   0.0100   0.0300   0.1250   0.7300   1.3480   0.3580
2010 -0.3750  -0.4970   0.6350   0.5280  -0.6710  -0.2830  -0.2600   0.4620   1.0490   0.6720   0.3970   0.5280
2011  0.1645  -0.0635  -0.6120   0.7650   0.7050  -0.3230   1.1670  -0.7820   1.1700   0.1800  -1.0290  -1.0850
2012  0.0220  -0.0970   0.3260   0.3040   1.3030  -0.1170   0.2430   0.7600   0.8380  -0.5430  -0.5400   0.9360
2013  0.7950  -0.2400   0.0670  -0.1040  -0.0290   0.5700   1.2190  -0.3490   0.6460   1.0480   1.9480   1.3640
2014  0.1870  -2.0770   0.4930  -2.1140  -0.9380   1.1010   0.7320  -0.5900   0.2390  -1.1960   0.4830  -0.4200
2015 -0.3280   3.5970   0.0380   0.9470   1.6360   0.2670   2.3490   1.9910   0.0770   2.1830   4.6060   0.6230
2016 -3.3810  -3.5530   1.9920   2.1520   4.1590   0.9690   1.2580   1.1570   0.1770   2.8760  -3.0560  -0.0330
2017  2.0790   1.4610   0.9560   2.5530   2.8130   1.4770   0.8790  -1.8160  -0.1020   1.5330   6.8880   1.5670
2018  6.7550   6.9430   4.6830  -3.3330   6.0890   5.3060   4.2550   5.5780   3.5300  -8.8580  -8.2420  -2.9250
2019  3.5050  -0.4440   4.7600   7.0670   0.3400  -0.8530   5.5300  -8.4300   0.2050  -2.4180   1.2060   0.5610
2020  4.9350   9.2740 -10.0950  17.6820   8.5140  10.9410  21.9600   9.8260  -4.0390   3.4580  -4.6810   2.7750
2021  0.4980   2.9280  -9.7140  14.0550  -4.9160   5.5150  12.3700 -15.1050   6.0160  -4.8360   9.6630  -5.3170
2022 -15.3370 -1.8450   0.4350  -2.4360 -39.4880   0.2640   3.9070  21.3260 -14.6100  -8.8760 -20.0690 -10.9920
2023 18.9800  -3.2040  -2.6380   6.9470   8.4720  14.6100   4.5100   4.6060   2.6850
```

In this code, the first difference of the Amazon.ts time series data is calculated using the diff() function. The lag argument is set to 1, which means that the first difference will be taken with respect to the previous value. The resulting data is printed to the console, showing the differences in the data from one time period to the next. This approach can be used to test for predictability in the time series data, as it can reveal whether the changes in the data are random or follow some pattern or trend.

```
#UseAcf()functiontoidentify autocorrelation forfirstdifferencedStockPriceandplotauto-correlation for different lags
# (up to maximum of 12).
Acf(diff.stock.price, lag.max = 12,
   main = "Autocorrelation for Amazon Stock Prices")
```



We can see significant correlations existing at lags 5, 8&9. Therefore, we can say that this is not random walk and we can predict the data.

```
## Creating Time Series Partition.

# Define the numbers of months in the training and validation sets,
# nTrain and nValid, respectively.
# Total number of period length(ridership.ts) = 300.
# nTrain = 240 months, from October 1998 to September  2018.
# nvalid = 60 months, from October 2018 to September 2023.
```

nValid<- 60
nTrain<- length(Amazon.ts) - nValid
train.ts<- window(Amazon.ts, start = c(1998, 10), end = c(1998, nTrain))
valid.ts<- window(Amazon.ts, start = c(1998, nTrain + 1),
end = c(1998, nTrain + nValid))

MODEL ONE:

Use two-level model, regression with linear trend and Seasonality and trailing ma for residuals for entire data set.

Use two-level (combined) forecast to forecast 12 future periods from September 2023.

Fit a regression model with linear trend and seasonality for entire data set.

tot.trend.seas<- tslm(Amazon.ts ~ trend + season)
summary(tot.trend.seas)

```
####MODEL ONE:


## USE TWO-LEVEL MODEL, REGRESSION WITH LINEAR TREND AND
## SEASONALITY AND TRAILING MA FOR RESIDUALS FOR ENTIRE
## DATA SET. USE TWO-LEVEL (COMBINED) FORECAST TO FORECAST
## 12 FUTURE PERIODS FROM APRIL 2023.

# Fit a regression model with linear trend and seasonality for
# entire data set.
tot.trend.seas <- tslm(Amazon.ts ~ trend  + season)
summary(tot.trend.seas)
```

```
Console   Terminal   Background Jobs
R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/
> # Fit a regression model with linear trend and seasonality for
> # entire data set.
> tot.trend.seas <- tslm(Amazon.ts ~ trend  + season)
> summary(tot.trend.seas)

Call:
tslm(formula = Amazon.ts ~ trend + season)

Residuals:
    Min      1Q  Median      3Q     Max 
-40.338 -24.717  -3.177  16.730  86.056 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -35.86983    6.68291  -5.367 1.65e-07 ***
trend         0.46609    0.01999  23.315  < 2e-16 ***
season2      -0.03707    8.47446  -0.004    0.997    
season3      -0.84423    8.47453  -0.100    0.921    
season4       0.58244    8.47465   0.069    0.945    
season5      -0.36816    8.47481  -0.043    0.965    
season6       0.75255    8.47502   0.089    0.929    
season7       2.66659    8.47528   0.315    0.753    
season8       2.92502    8.47559   0.345    0.730    
season9       2.45169    8.47594   0.289    0.773    
season10      1.53116    8.47465   0.181    0.857    
season11      0.60675    8.47453   0.072    0.943    
season12     -0.25237    8.47446  -0.030    0.976    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 29.96 on 287 degrees of freedom
Multiple R-squared:  0.6553,    Adjusted R-squared:  0.6409 
F-statistic: 45.47 on 12 and 287 DF,  p-value: < 2.2e-16
```

The output is the summary of a multiple linear regression model using the tslm() function from the forecast package. The response variable is the Amazon stock price (Amazon.ts), and the predictors are a linear trend (trend) and seasonal dummies (season2, season3, ..., season12), which represent the 12 months of the year. The estimates for the coefficients are given in the "Coefficients" table.

The intercept estimate is -35.869, which represents the expected value of the response variable when all predictors are equal to zero. The estimate for the trend variable is 0.466, which represents the expected change in the response variable for a unit increase in time (in months). The estimates for the seasonal dummies represent the expected difference in the response variable between the given month and the base month (January, represented by season1).

The p-values in the "Pr(>|t|)" column indicate the significance of each coefficient, with smaller p-values indicating stronger evidence against the null hypothesis (that the coefficient is zero). The p-value for trend is very small, indicating that it is highly significant. However, most of the seasonal dummies have large p-values, indicating that they are not significant predictors of the response variable.

 # Create regression forecast for future 12 periods.
tot.trend.seas.pred<- forecast(tot.trend.seas, h = 12, level = 0)
tot.trend.seas.pred

```
> # Create regression forecast for future 12 periods.
> tot.trend.seas.pred <- forecast(tot.trend.seas, h = 12,
+                                 level = 0)
> tot.trend.seas.pred
         Point Forecast     Lo 0     Hi 0
Oct 2023       105.9531 105.9531 105.9531
Nov 2023       105.4948 105.4948 105.4948
Dec 2023       105.1018 105.1018 105.1018
Jan 2024       105.8202 105.8202 105.8202
Feb 2024       106.2493 106.2493 106.2493
Mar 2024       105.9082 105.9082 105.9082
Apr 2024       107.8009 107.8009 107.8009
May 2024       107.3164 107.3164 107.3164
Jun 2024       108.9032 108.9032 108.9032
Jul 2024       111.2833 111.2833 111.2833
Aug 2024       112.0079 112.0079 112.0079
Sep 2024       112.0006 112.0006 112.0006
```

The above output shows the Point Forecast column of the predicted values for each period, while the Lo 0 and Hi 0 columns show the lower and upper bounds of the 0% prediction interval, respectively. The forecast predicts an overall upward trend, with some fluctuations around the trend. It is important to note that this forecast assumes that the underlying trend and seasonality patterns will continue into the future, and that there will be no major unexpected events that could affect the stock price.

 # Identify and display regression residuals for entire data set.
tot.trend.seas.res <- tot.trend.seas$residuals
tot.trend.seas.res

```
> # Identify and display regression residuals for entire data set.
> tot.trend.seas.res <- tot.trend.seas$residuals
> tot.trend.seas.res
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1998										34.7555838	35.6279038	36.8909438
1999	37.3724838	36.3114638	37.2805438	36.5777838	35.8703038	33.7085038	31.5433838	30.4578638	31.0711038	32.2595552	32.5478752	33.7389152
2000	31.8314552	31.6024352	31.6445152	29.2567552	29.5882752	27.6094752	24.7863552	24.0298352	24.3290752	24.2175265	24.6658465	24.5948865
2001	23.7094265	23.0714065	23.2804865	21.5087265	22.1112465	20.4484465	18.0953265	17.1568065	17.0110465	17.4744979	17.9878179	18.5028579
2002	17.8273979	17.4503779	17.9044579	15.9706979	16.6362179	15.0094179	12.5022979	11.7617779	11.8600179	12.4164692	13.0237892	13.4458292
2003	12.7033692	12.2743492	12.7914292	10.9656692	11.7331892	10.3143892	8.1052692	7.5457492	7.8199892	8.7024406	8.9987606	9.3238006
2004	8.7333406	7.8843206	8.0724006	6.3846406	6.7051606	5.4933606	2.8872406	1.7487206	1.8889606	2.2324119	2.6867319	3.1957719
2005	2.5693119	1.8262919	2.0853719	0.1506119	0.6801319	-0.9016681	-3.2567881	-3.9843081	-3.8590681	-3.0966167	-2.5922967	-1.9872567
2006	-2.8847167	-3.6367367	-3.4116567	-5.3094167	-4.9368967	-6.4696967	-8.9388167	-9.9483367	-9.7520967	-9.1786454	-8.3963254	-8.0512854
2007	-8.8597454	-9.1967654	-8.8966854	-10.4084454	-9.0169254	-10.3037254	-12.4788454	-13.0863654	-12.5161254	-11.8946740	-11.8553540	-11.0343140
2008	-12.2457740	-13.1307940	-12.9977140	-14.4674740	-13.9599540	-15.4147540	-18.1788740	-18.4113940	-18.6971540	-19.2277027	-19.3723827	-18.7403427
2009	-19.3178027	-19.1938227	-18.5917427	-19.9825027	-19.5359827	-20.8477827	-23.2179027	-23.9124227	-23.7801827	-22.5957313	-20.7894113	-20.0383713
2010	-21.1318313	-22.0578513	-21.0817713	-22.4465313	-22.6330113	-24.5028113	-27.1429313	-27.4054513	-26.3492113	-25.2227600	-24.3674400	-23.4464000
2011	-24.0003600	-24.4928800	-24.7638000	-25.8915600	-24.7020400	-26.6118400	-27.8249600	-29.3314800	-28.1542400	-27.5197887	-28.0904687	-28.7824287
2012	-29.4788887	-30.0049087	-29.3378287	-30.9265887	-29.1390687	-30.8428687	-32.9799887	-32.9445087	-32.0992687	-32.1878173	-32.2694973	-30.9404573
2013	-30.8639173	-31.5329373	-31.1248573	-33.1216173	-32.6660973	-33.6828973	-34.8440173	-35.9175373	-35.2642973	-33.7618460	-31.3555260	-29.5984860
2014	-30.1299460	-32.6359660	-31.8018860	-35.8086460	-36.2621260	-36.7479260	-38.3960460	-39.7105660	-39.4643260	-40.2058746	-39.2645546	-39.2915146
2015	-40.3379746	-37.1699946	-36.7909146	-37.7366746	-35.6161546	-36.9359546	-36.9670746	-35.7005946	-35.6163546	-32.9789033	-27.9145833	-26.8985433
2016	-30.9980033	-34.9800233	-32.6469433	-32.3877033	-27.7441833	-28.3619833	-29.4841033	-29.0516233	-28.8673833	-25.5369319	-28.1346119	-27.7745719
2017	-26.4140319	-25.3820519	-24.0849719	-23.4247319	-20.1272119	-20.2370119	-21.7381319	-24.2786519	-24.3734119	-22.3859606	-15.0396406	-13.0796006
2018	-7.0430606	-0.5290806	4.4949994	-0.7307606	5.8427594	9.5619594	11.4368394	16.2903194	19.8275594	11.4240108	3.6403308	1.1083708
2019	3.8949108	3.0218908	8.1229708	13.2972108	14.1217308	11.6819308	14.8318108	5.6772908	5.8895308	3.9259821	5.5903021	6.5443421
2020	10.7608821	19.6058621	9.8519421	25.6411821	34.6397021	43.9939021	63.5737821	72.6752621	68.6435021	72.5559535	68.3332735	71.5013135
2021	71.2808535	73.7798335	64.4069135	76.5691535	72.1376735	76.0658735	86.0557535	70.2262335	76.2494735	71.8679248	81.9892448	77.0652848
2022	61.0098248	58.7358048	59.5118848	55.1831248	16.1796448	14.8568448	16.3837248	36.9852048	22.3824448	13.9608962	-5.6497838	-16.2487438
2023	2.0127962	-1.6202238	-3.9171438	1.1370962	10.0936162	23.1168162	25.2466962	29.1281762	31.8204162			

These residuals are the differences between the actual values of the time series and the values predicted by the trend and seasonal components of the model. They represent the unexplained variability in the data after accounting for the trend and seasonal patterns. The residuals are displayed in the output above, which shows the values for each month from October1998 to September 2023. The blank spaces in the last row indicate that the time series model has not yet made predictions for those months.

 # Use trailing MA to forecast residuals for entire data set.

tot.ma.trail.res <- rollmean(tot.trend.seas.res, k = 4, align = "right")
tot.ma.trail.res

```r
# Use trailing MA to forecast residuals for entire data set.
tot.ma.trail.res <- rollmean(tot.trend.seas.res, k = 4,
                             align = "right")
tot.ma.trail.res
```

R 4.2.1 · C:/users/manu7/Desktop/MSBA Sem-2/Sem-2/BAN 673/Final Project/OurProject/OurProject/

```
> # Use trailing MA to forecast residuals for entire data set.
> tot.ma.trail.res <- rollmean(tot.trend.seas.res, k = 4,
+                              align = "right")
> tot.ma.trail.res
             Jan          Feb          Mar          Apr          May          Jun          Jul
2013                                                                                  11.903409091
2014  12.032159091  12.493159091  13.042409091  11.854511364  10.440113636   9.276465909   7.676818182
2015   2.023068182   0.681568182   0.433318182  -0.564579545  -1.246477273  -1.905125000  -2.852272727
2016   1.258977273   0.592477273   0.009227273  -1.408670455  -2.620568182  -3.496715909  -5.153863636
2017  -6.075113636  -6.926613636  -6.502363636  -6.990261364  -6.429659091  -5.513306818  -5.732954545
2018  -4.261704545  -4.205704545  -4.006454545  -5.416852273  -7.298750000  -7.394897727  -7.674545455
2019 -14.000795455 -15.157295455 -14.095545455 -13.125943182 -13.242840909 -14.496488636 -16.471136364
2020 -12.359886364 -10.608886364 -12.552136364 -15.187534091 -16.941931818 -18.138079545 -15.575227273
2021  -3.833977273   1.467022727   4.866272727   9.918375000  15.193977273  18.907829545  23.795681818
2022  41.726931818  39.912931818  36.479681818  31.126784091  24.232386364  17.343738636  10.084090909
2023 -16.509659091 -18.248659091 -17.674409091
             Aug          Sep          Oct          Nov          Dec
2013  11.148159091  10.134659091   9.731909091   9.856909091  10.999659091
2014   6.839068182   6.393068182   5.515318182   4.332818182   3.513068182
2015  -2.660022727  -2.471022727  -1.508772727  -0.426272727   0.701477273
2016  -5.911613636  -6.037613636  -5.627863636  -5.740363636  -5.885113636
2017  -6.063204545  -7.324204545  -7.774454545  -7.421954545  -6.119204545
2018  -6.467295455  -6.138295455  -7.231045455  -9.606045455 -12.130795455
2019 -18.333886364 -18.599886364 -17.515136364 -16.317636364 -14.497386364
2020 -12.947977273 -12.338977273 -11.711727273  -9.501727273  -6.851477273
2021  28.640431818  34.056931818  38.011681818  41.976681818  43.549431818
2022   5.756340909   2.325340909  -1.889909091  -7.152409091 -13.279659091
2023
>
```

Environment | History | Connections | Tutorial

Import Dataset · 233 MiB · List

R · Global Environment

alpha	0.05
ar1	0.9943
diff.stock.price	Time-Series [1:119] from 2013 to 2023: 2.14 0.04 0.59 -0.62 0.2 ...
Google.ts	Time-Series [1:120] from 2013 to 2023: 19.8 21.9 22 22.6 21.9 ...
nTrain	96
null_mean	1
nValid	24
p.value	0.186565191332239
s.e.	0.0064
tot.ma.trail.res	Time-Series [1:117] from 2014 to 2023: 11.9 11.15 10.13 9.73 9.86 ...
tot.trend.seas.res	Time-Series [1:120] from 2013 to 2023: 10.95 13.4 12.48 10.79 7.93 ...
train.fst.2level	Time-Series [1:117] from 2014 to 2023: 23.7 25.2 22.9 22.3 24.4 ...
train.ts	Time-Series [1:93] from 2013 to 2021: 19.8 21.9 22 22.6 21.9 ...
valid.ts	Time-Series [1:24] from 2021 to 2023: 90 103 102 114 115 ...
z.stat	-0.890625000000006

This code uses a rolling mean to forecast the residuals for the entire data set. The rollmean() function calculates the mean of a rolling window, with a window size of 4 and the align parameter set to "right", which means that the mean is calculated using the **4 most recent residual values** to the right of the current data point.

The resulting tot.ma.trail.res object is a matrix that contains the forecasted residuals for each month in the data set. The matrix has rows for each year and columns for each month, with missing values for the last few months of the last year.

 # Identify two-level forecast, regression with linear trend and
 # seasonality and trailing MA for regression residuals, for
 # entire data set (training period).
train.fst.2level <- tot.trend.seas.pred$fitted+tot.ma.trail.res
train.fst.2level

```
# Identify two-level forecast, regression with linear trend and
# seasonality and trailing MA for regression residuals, for
# entire data set (training period).
train.fst.2level <- tot.trend.seas.pred$fitted+tot.ma.trail.res
train.fst.2level
```

```
109:17   (Top Level) ÷                                          R Script ÷
```

```
R 4.2.1 · C:/users/manu7/Desktop/MSBA Sem-2/Sem-2/BAN 673/Final Project/OurProject/OurProject/
> # Identify two-level forecast, regression with linear trend and
> # seasonality and trailing MA for regression residuals, for
> # entire data set (training period).
> train.fst.2level <- tot.trend.seas.pred$fitted+tot.ma.trail.res
> train.fst.2level
           Jan       Feb       Mar       Apr       May       Jun       Jul       Aug       Sep       Oct
2013                                                             23.66225  25.15300  22.92050  22.32475
2014  26.98800  29.55200  29.22425  31.44994  29.72355  29.51990  30.19725  31.60550  29.94050  28.86975
2015  27.74050  28.50200  27.37675  29.79244  28.79855  29.09990  30.42975  32.86800  31.83800  32.60725
2016  37.73800  39.17450  37.71425  39.70994  38.18605  38.26990  38.88975  40.37800  39.03300  39.24975
2017  41.16550  42.41700  41.96425  44.88994  45.13855  47.01490  49.07225  50.98800  48.50800  47.86475
2018  53.74050  55.89950  55.22175  57.22494  55.03105  55.89490  57.89225  61.34550  60.45550  59.16975
2019  54.76300  55.70950  55.89425  60.27744  59.84855  59.55490  59.85725  60.24050  58.75550  59.64725
2020  67.16550  71.01950  68.19925  68.97744  66.91105  66.67490  71.51475  76.38800  75.77800  76.21225
2021  86.45300  93.85700  96.37925 104.84494 109.80855 114.48240 121.64725 128.73800 132.93550 136.69725
2022 142.77550 143.06450 138.75425 136.81494 129.60855 123.67990 118.69725 116.61550 111.96550 107.55725
2023  95.30050  95.66450  95.36175
           Nov       Dec
2013  24.37575  25.61350
2014  29.61325  28.88850
2015  35.61575  36.83850
2016  41.06325  41.01350
2017  50.14325  51.54100
2018  58.72075  56.29100
2019  62.77075  64.68600
2020  80.34825  83.09350
2021 142.58825 144.25600
2022 104.22075  98.18850
2023
>
```

The code is generating a two-level forecast by adding the fitted values obtained from a regression model with a linear trend and seasonality to the trailing moving average of the regression residuals. The resulting forecast is then calculated for the entire dataset, including the training period. The code is stored in the object train.fst.2level and the resulting forecast is shown as a table of values for each month from January to December for the years 1998 to 2023.

Specifically, tot.trend.seas.pred$fitted is the fitted values obtained from a regression model with a linear trend and seasonality, and tot.ma.trail.res is the trailing moving average of the regression residuals. These two components are added together to obtain the two-level forecast for the training period. The resulting forecast is shown as a table of values for each month from January to December for the years 2013 to 2023.

 # Create forecast for trailing MA residuals for future 12 periods.
tot.ma.trail.res.pred<- forecast(tot.ma.trail.res, h = 12,
level = 0)
tot.ma.trail.res.pred

```
> # Create forecast for trailing MA residuals for future 12 periods.
> tot.ma.trail.res.pred <- forecast(tot.ma.trail.res, h = 12,
+                                   level = 0)
> tot.ma.trail.res.pred
         Point Forecast     Lo 0     Hi 0
Oct 2023       31.71293 31.71293 31.71293
Nov 2023       35.25265 35.25265 35.25265
Dec 2023       38.11012 38.11012 38.11012
Jan 2024       40.41682 40.41682 40.41682
Feb 2024       42.27892 42.27892 42.27892
Mar 2024       43.78211 43.78211 43.78211
Apr 2024       44.99557 44.99557 44.99557
May 2024       45.97515 45.97515 45.97515
Jun 2024       46.76592 46.76592 46.76592
Jul 2024       47.40427 47.40427 47.40427
Aug 2024       47.91958 47.91958 47.91958
Sep 2024       48.33557 48.33557 48.33557
```

The forecast for the trailing MA residuals for the next 12 periods is obtained using the forecast() function. The resulting forecast shows the point forecast for each month as well as the lower and upper bounds of the forecast interval at level 0. The forecast values suggest that the residuals are expected to increase gradually in the next 12 months.

 # Develop 2-level forecast for future 12 periods by combining
 # regression forecast and trailing MA for residuals for future
 # 12 periods.
tot.fst.2level <- tot.trend.seas.pred$mean +
tot.ma.trail.res.pred$mean
tot.fst.2level

```
> # Develop 2-level forecast for future 12 periods by combining
> # regression forecast and trailing MA for residuals for future
> # 12 periods.
> tot.fst.2level <- tot.trend.seas.pred$mean +
+   tot.ma.trail.res.pred$mean
> tot.fst.2level
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov      Dec
2023                                                                                  137.6661 140.7475 143.2119
2024 146.2371 148.5282 149.6903 152.7965 153.2916 155.6691 158.6876 159.9274 160.3362
```

The 2-level forecast is the sum of the point forecasts for the regression model and the trailing MA for residuals model for each period in the forecast horizon.

 # Create a table with regression forecast, trailing MA for residuals,
 # and total forecast for future 12 periods.
future12.df <- round(data.frame(tot.trend.seas.pred$mean,
tot.ma.trail.res.pred$mean, tot.fst.2level), 3)
names(future12.df) <- c("Regression.Fst", "MA.Residuals.Fst",
"Combined.Fst")
future12.df

Amazon.Data	300 obs. of 2 variables
future12.df	12 obs. of 3 variables
HW.ZZZ	List of 19
HW.ZZZ.pred	List of 10
lin.season	List of 16
lin.season.pred	List of 11
Stock.ar1	List of 18
Stock.stl	List of 8
tot.ma.trail.res.pred	List of 10
tot.trend.seas	List of 16
tot.trend.seas.pred	List of 11
Values	
alpha	0.05
Amazon.ts	Time-Series [1:300] from 1999 to 2024: 0.883 1.297 2.167 3.367 2.735 ...
ar1	0.9971
diff.stock.price	Time-Series [1:299] from 1999 to 2024: 0.414 0.87 1.2 -0.632 0.628 ...
nTrain	240
null_mean	1
nValid	60
p.value	0.182401771838493
s.e.	0.0032
tot.fst.2level	Time-Series [1:12] from 2024 to 2025: 138 141 143 146 149 ...
tot.ma.trail.res	Time-Series [1:297] from 1999 to 2024: 36.2 36.6 37 36.9 36.5 ...
tot.trend.seas.res	Time-Series [1:300] from 1999 to 2024: 34.8 35.6 36.9 37.4 36.3 ...
train.fst.2level	Time-Series [1:297] from 1999 to 2024: 2.16 2.97 3.05 4.86 4 ...
train.ts	Time-Series [1:231] from 1999 to 2018: 0.883 1.297 2.167 3.367 2.735 ...
valid.ts	Time-Series [1:60] from 2018 to 2023: 65.2 72.2 76.8 73.5 79.6 ...
z.stat	-0.906250000000004

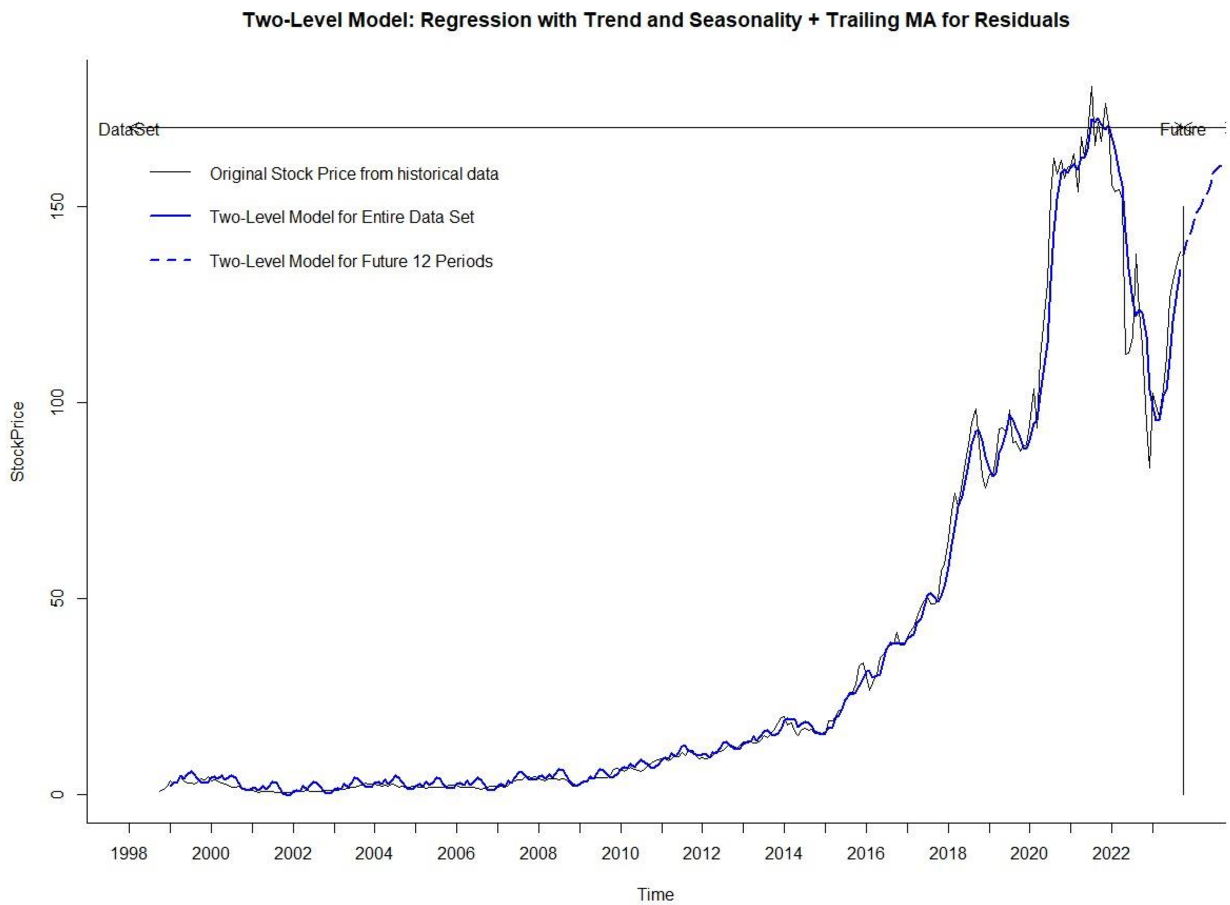
```
> # Create a table with regression forecast, trailing MA for residuals,
> # and total forecast for future 12 periods.
> future12.df <- round(data.frame(tot.trend.seas.pred$mean,
+                              tot.ma.trail.res.pred$mean, tot.fst.2level), 3)
> names(future12.df) <- c("Regression.Fst", "MA.Residuals.Fst",
+                        "Combined.Fst")
> future12.df
   Regression.Fst MA.Residuals.Fst Combined.Fst
1         105.953           31.713      137.666
2         105.495           35.253      140.747
3         105.102           38.110      143.212
4         105.820           40.417      146.237
5         106.249           42.279      148.528
6         105.908           43.782      149.690
7         107.801           44.996      152.797
8         107.316           45.975      153.292
9         108.903           46.766      155.669
10        111.283           47.404      158.688
11        112.008           47.920      159.927
12        112.001           48.336      160.336
```

Now we have a table that shows the regression forecast, trailing MA for residuals forecast, and combined forecast for the future 12 periods. The combined forecast is obtained by adding the regression forecast and the trailing MA for residuals forecast. The combined forecast can be more accurate than each forecast method on its own, as it takes into account the strengths of each method.

```
# Plot historical data set and two-level model's forecast for
# entire data set and future 12 monthly periods.

plot(Amazon.ts,
xlab = "Time", ylab = "StockPrice", ylim = c(0, 180),
bty = "l", xlim = c(1998, 2023.75), lwd =1, xaxt = "n",
    main = "Two-Level Model: Regression with Trend and Seasonality + Trailing MA for Residuals")
axis(1, at = seq(1998, 2023.75, 1), labels = format(seq(1998, 2023.75, 1)))
lines(train.fst.2level, col = "blue", lwd = 2)
lines(tot.fst.2level, col = "blue", lty =5, lwd = 2)
legend(1998,170, legend = c("Original Stock Price from historical data", "Two-Level Model for Entire Data Set",
                        "Two-Level Model for Future 12 Periods"),
      col = c("black", "blue" , "blue"),
lty = c(1, 1, 2), lwd =c(1, 2, 2), bty = "n")
```

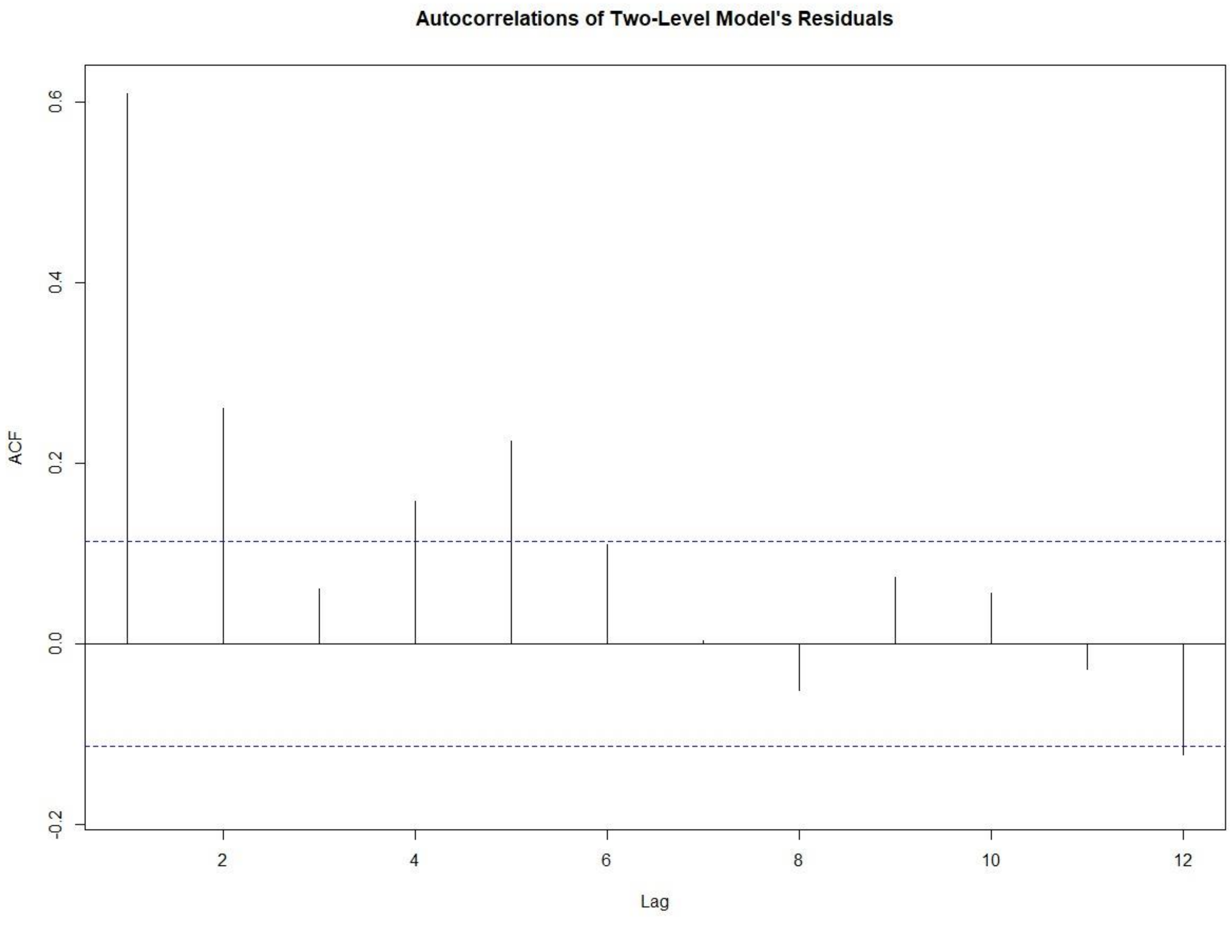
 # Plot on chart vertical lines and horizontal arrows describingentire data set and future #prediction intervals.

```
lines(c(2023.25, 2023.25), c(0, 150))
text(2017, 150 , "DataSet")
text(2023.5, 150, "Future")
arrows(2013.25, 140, 2023.25, 140, code = 3, length = 0.1,
lwd = 1, angle = 30)
arrows(2023.25, 140, 2024, 140, code = 3, length = 0.1,
lwd = 1, angle = 30)
```



The code is creating a plot of the historical stock price data along with the two-level model's forecast for the entire data set and the future 12 monthly periods . It also includes vertical lines and horizontal arrows to describe the entire data set and future prediction intervals, Overall, this code creates a well-labeled and informative plot of the historical data and future predictions using the two-level model.

```
# Use Acf() function to create autocorrelation chart of
# two-level model's residuals.
Acf((Amazon.ts - train.fst.2level), lag.max = 12,
   main = "Autocorrelations of Two-Level Model's Residuals")
```



This will produce a plot of the ACF of the residuals. The plot will show the autocorrelation coefficients on the y-axis and the lag on the x-axis. The blue shaded area represents the 95% confidence intervals for the ACF. Any autocorrelation coefficient that falls outside of these confidence intervals is considered statistically significant.

###**MODEL TWO:**

FORECAST WITH HOLT-WINTER'S MODEL USING ENTIRE DATA SET INTO
THE FUTURE FOR 12 MONTHLY PERIODS IN 2019.

 # Create Holt-Winter's (HW) exponential smoothing for
 # entire data set. Use ets() function with model = "ZZZ" to
 # identify the best HW options and optimal alpha, beta, & gamma
 # to fit HW for the entire data set.
HW.ZZZ <- ets(Amazon.ts, model = "ZZZ")
HW.ZZZ

```
### Model 2:


## FORECAST WITH HOLT-WINTER'S MODEL USING ENTIRE DATA SET INTO
## THE FUTURE FOR 12 MONTHLY PERIODS IN Oct 2023- Sep 24.

# Create Holt-Winter's (HW) exponential smoothing for
# entire data set. Use ets() function with model = "ZZZ" to
# identify the best HW options and optimal alpha, beta, & gamma
# to fit HW for the entire data set.
HW.ZZZ <- ets(Amazon.ts, model = "ZZZ")
HW.ZZZ

# Use forecast() function to make predictions using this
# HW model for upcoming 12 months with no confidence interval.
HW.ZZZ.pred <- forecast(HW.ZZZ, h = 12 , level = 0)
HW.ZZZ.pred
```

182:1 (Top Level) ÷

Console Terminal × Background Jobs ×

R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/

```
> # Create Holt-Winter's (HW) exponential smoothing for
> # entire data set. Use ets() function with model = "ZZZ" to
> # identify the best HW options and optimal alpha, beta, & gamma
> # to fit HW for the entire data set.
> HW.ZZZ <- ets(Amazon.ts, model = "ZZZ")
> HW.ZZZ
ETS(M,A,N)

Call:
 ets(y = Amazon.ts, model = "ZZZ")

  Smoothing parameters:
    alpha = 0.9999
    beta  = 0.0431

  Initial states:
    l = 0.8671
    b = 0.3298

  sigma:  0.1137

     AIC     AICc      BIC
1799.452 1799.656 1817.971
```

The ets() function determined that the best Holt-Winter's (HW) model for the entire data set is a (M,Ad,M) model with smoothing parameters alpha = 0.9999 and beta = 0.0431. The initial states are l = 0.8671 and b = 0.3298, and the estimated sigma value is 0.1137. The AIC, AICc, and BIC values are also provided. However, it appears that the statement regarding beta, gamma, and phi is incorrect and not applicable in this case as the model parameter was set to "ZZZ" and not "AAM", "MAM", or "ZAM".

 # Use forecast() function to make predictions using this
 # HW model for upcoming 12 months with no confidence interval.
HW.ZZZ.pred<- forecast(HW.ZZZ, h = 12 , level = 0)
HW.ZZZ.pred

Amazon.Data	300 obs. of 2 variables
future12.df	12 obs. of 3 variables
HW.ZZZ	List of 19
HW.ZZZ.pred	List of 10
lin.season	List of 16
lin.season.pred	List of 11
Stock.ar1	List of 18
Stock.stl	List of 8
tot.ma.trail.res.pred	List of 10
tot.trend.seas	List of 16
tot.trend.seas.pred	List of 11
Values	
alpha	0.05
Amazon.ts	Time-Series [1:300] from 1999 to 2024: 0.883 1.297 2.167 3.367 2.735 ...
ar1	0.9971
diff.stock.price	Time-Series [1:299] from 1999 to 2024: 0.414 0.87 1.2 -0.632 0.628 ...
nTrain	240
null_mean	1
nValid	60
p.value	0.182401771838493
s.e.	0.0032
tot.fst.2level	Time-Series [1:12] from 2024 to 2025: 138 141 143 146 149 ...
tot.ma.trail.res	Time-Series [1:297] from 1999 to 2024: 36.2 36.6 37 36.9 36.5 ...
tot.trend.seas.res	Time-Series [1:300] from 1999 to 2024: 34.8 35.6 36.9 37.4 36.3 ...
train.fst.2level	Time-Series [1:297] from 1999 to 2024: 2.16 2.97 3.05 4.86 4 ...
train.ts	Time-Series [1:231] from 1999 to 2018: 0.883 1.297 2.167 3.367 2.735 ...
valid.ts	Time-Series [1:60] from 2018 to 2023: 65.2 72.2 76.8 73.5 79.6 ...
z.stat	-0.906250000000004

```
# Use forecast() function to make predictions using this
# HW model for upcoming 12 months with no confidence interval.
HW.ZZZ.pred <- forecast(HW.ZZZ, h = 12 , level = 0)
HW.ZZZ.pred

# Plot HW model's predictions for historical data set and
```

187:1 (Top Level)

Console Terminal Background Jobs

R 4.2.1 · C:/Users/o/Desktop/My subjects/Third Semester/Capstone/Dataset/

```
> # Use forecast() function to make predictions using this
> # HW model for upcoming 12 months with no confidence interval.
> HW.ZZZ.pred <- forecast(HW.ZZZ, h = 12 , level = 0)
> HW.ZZZ.pred
         Point Forecast     Lo 0     Hi 0
Oct 2023       138.9389 138.9389 138.9389
Nov 2023       139.6501 139.6501 139.6501
Dec 2023       140.3613 140.3613 140.3613
Jan 2024       141.0724 141.0724 141.0724
Feb 2024       141.7836 141.7836 141.7836
Mar 2024       142.4947 142.4947 142.4947
Apr 2024       143.2059 143.2059 143.2059
May 2024       143.9170 143.9170 143.9170
Jun 2024       144.6282 144.6282 144.6282
Jul 2024       145.3393 145.3393 145.3393
Aug 2024       146.0505 146.0505 146.0505
Sep 2024       146.7616 146.7616 146.7616
> 
```

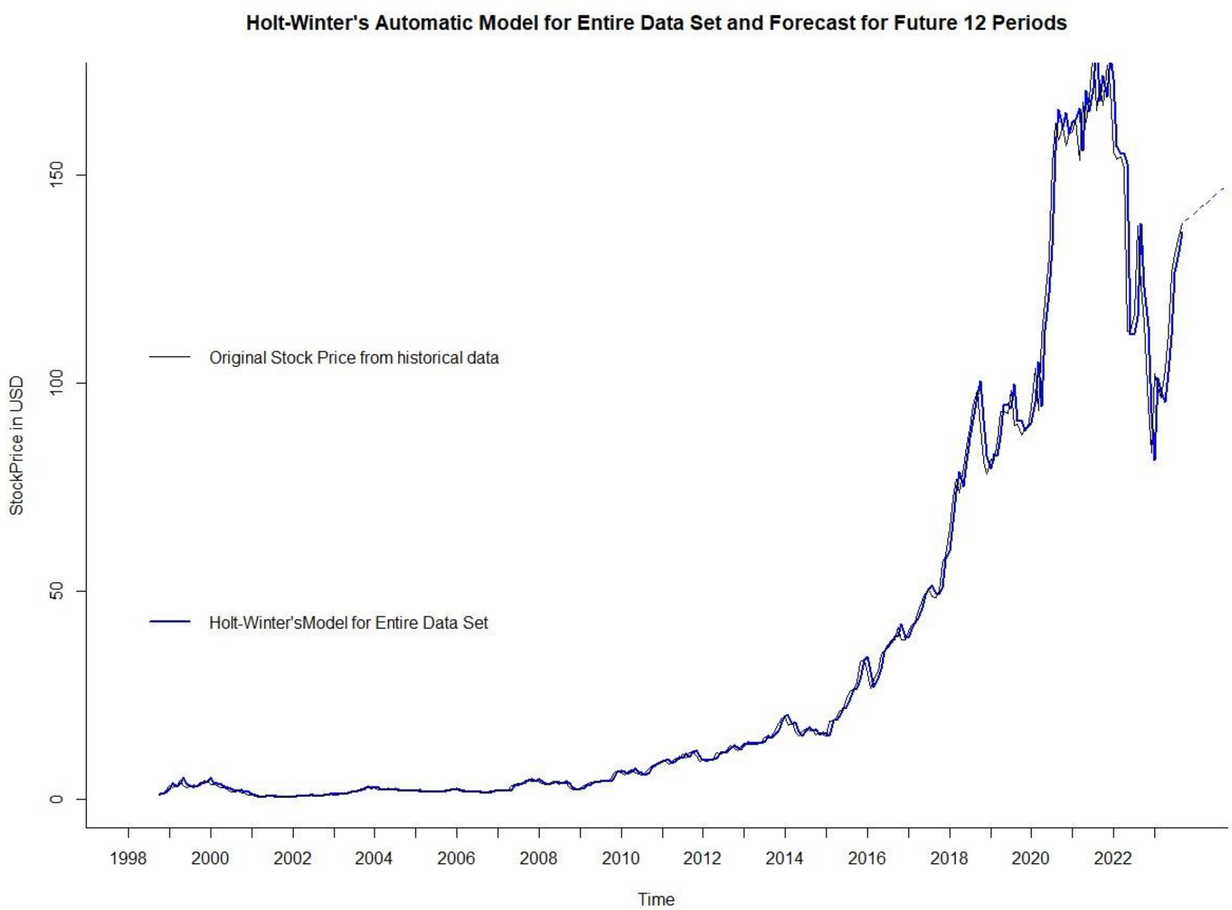
The result shows the point forecast for the upcoming 12 months (October 2023 to September 2024) based on the Holt-Winter's (HW) exponential smoothing model, which was fitted using the entire historical data set.

The point forecast is the estimated value of the stock price for each month, which is calculated based on the fitted HW model. In this case, the point forecast starts at 138.9389 for October 2023 and increases steadily over the forecast horizon, reaching 146.7616 for September 2024.

The Lo 0 and Hi 0 columns show the lower and upper bounds of the 0% confidence interval for each forecasted value. Since we specified level = 0 in the forecast() function, these columns are not displayed in the output. However, by default, the forecast() function produces 80%, 95%, and 99% prediction intervals in addition to the point forecasts, which can be useful for assessing the uncertainty of the forecasts.

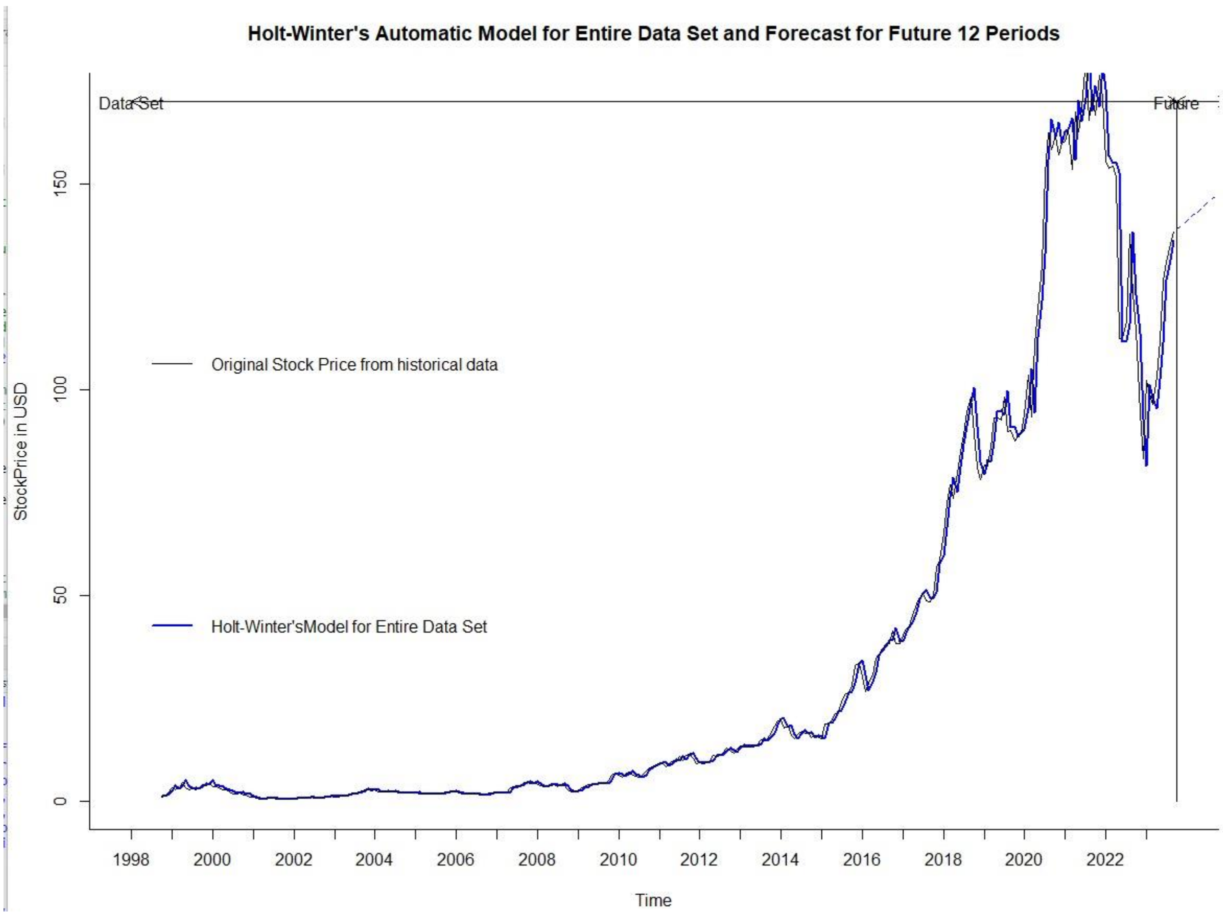
 # Plot HW model's predictions for historical data set and future 12 months.

```
plot(HW.ZZZ.pred$mean,
xlab = "Time", ylab = "StockPrice in USD", ylim = c(0, 170),
bty = "l", xlim = c(1998, 2023.75),lwd=1, xaxt = "n",
    main = "Holt-Winter's Automatic Model for Entire Data Set and Forecast for Future 12 Periods",
lty = 2, col = "blue")
axis(1, at = seq(1998, 2023.75, 1), labels = format(seq(1998, 2023.75, 1)))
lines(HW.ZZZ.pred$fitted, col = "blue", lwd = 2)
lines(Amazon.ts)
legend(1998,170,
      legend = c("Original Stock Price from historical data",
              "Holt-Winter'sModel for Entire Data Set",
              "Holt-Winter's Model Forecast, Future 12 Periods"),
      col = c("black", "blue" , "blue"),
lty = c(1, 1, 2), lwd =c(1, 2, 2), bty = "n")
```



 # Plot on chart vertical lines and horizontal arrows describing
 # entire data set and future prediction intervals.

```
lines(c(2023.75, 2023.75), c(0, 170))
text(1998, 170, "Data Set")
text(2023.75, 170, "Future")
arrows(1998, 170, 2023.75, 170, code = 3, length = 0.1,
lwd = 1, angle = 30)
arrows(2023.75, 170, 2025, 170, code = 3, length = 0.1,
lwd = 1, angle = 30)
```



The code above creates a plot showing the predictions of the Holt-Winter's exponential smoothing model for the entire data set and the next 12 months. The plot includes the following elements:

A blue line representing the predicted values for the entire data set, overlaid on the original Amazon stock price time series.

A blue dotted line representing the predicted values for the next 12 months, extending beyond the end of the time series.

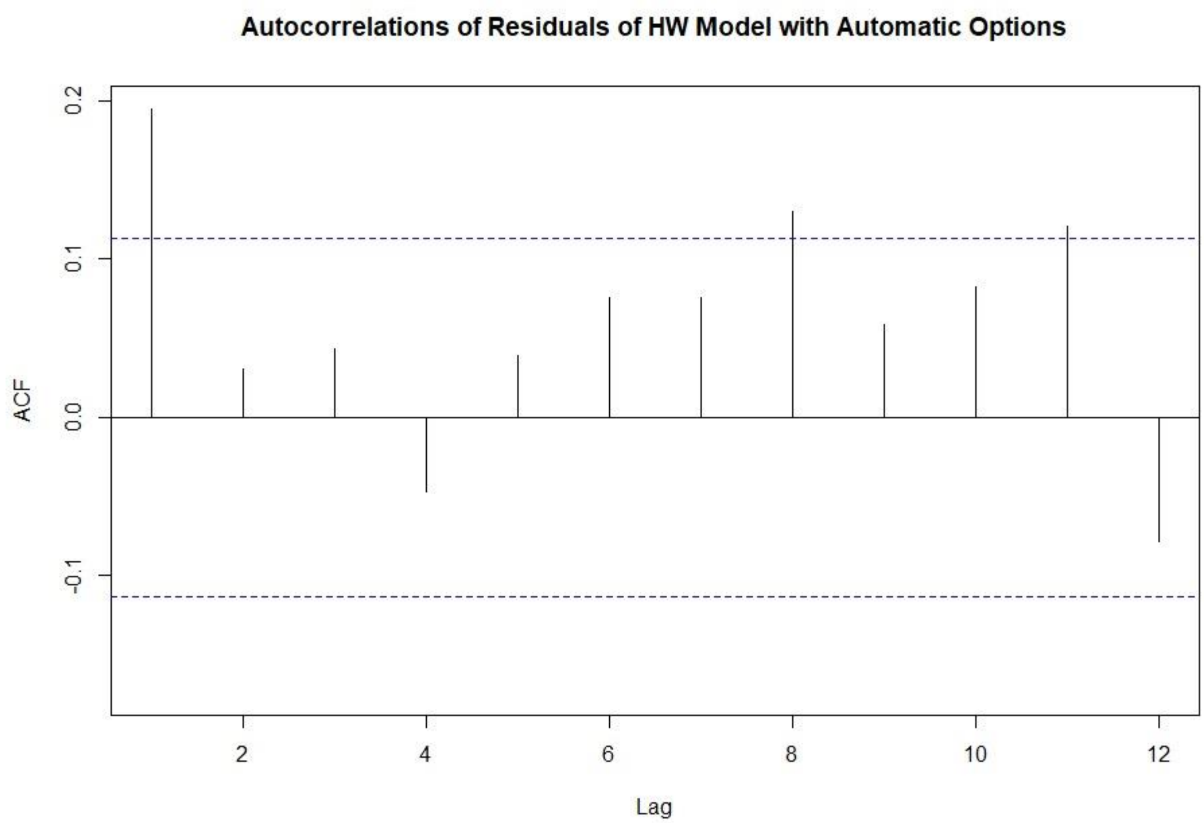
Vertical lines marking the end of the time series and the beginning of the future predictions.

Text labels indicating "Data Set" and "Future".

Horizontal arrows indicating the range of the data set and the future predictions.

The plot also includes a legend describing the three lines and their styles, and axes labels and limits for the x-axis (time) and y-axis (stock price in USD).

```
# Use Acf() function to create autocorrelation chart of
# residuals of HW model's with automated options and parameters.
Acf(HW.ZZZ.pred$residuals, lag.max = 12,
   main = "Autocorrelations of Residuals of HW Model with Automatic Options")
```



The autocorrelation function (ACF) plot shows the correlation of the residuals of the HW model at various lags. If the residuals are uncorrelated, we expect the ACF plot to be close to zero for all lags. If the residuals are correlated, we will see patterns in the ACF plot.

By examining the ACF plot, we can assess whether there is any remaining autocorrelation in the residuals of the HW model. If there is remaining autocorrelation, it suggests that the model has not captured all the information in the data, and we may need to consider using a more complex model.

###MODEL 3:

FIT TWO-LEVEL MODEL, LINEAR TREND AND SEASONALITY REGRESSION
AND AR(1) FOR REGRESSION RESIDUALS, FOR ENTIRE DATA SET.

 # Use tslm() function to create linear trend and seasonality model.
lin.season<- tslm(Amazon.ts ~ trend + season)

Amazon.Data	300 obs. of 2 variables
future12.df	12 obs. of 3 variables
HW.ZZZ	List of 19
HW.ZZZ.pred	List of 10
lin.season	List of 16
lin.season.pred	List of 11
Stock.ar1	List of 18
Stock.stl	List of 8
tot.ma.trail.res.pred	List of 10
tot.trend.seas	List of 16
tot.trend.seas.pred	List of 11
Values	
alpha	0.05
Amazon.ts	Time-Series [1:300] from 1999 to 2024: 0.883 1.297 2.167 3.367 2.735 ...
ar1	0.9971
diff.stock.price	Time-Series [1:299] from 1999 to 2024: 0.414 0.87 1.2 -0.632 0.628 ...
nTrain	240
null_mean	1
nValid	60
p.value	0.182401771838493
s.e.	0.0032
tot.fst.2level	Time-Series [1:12] from 2024 to 2025: 138 141 143 146 149 ...
tot.ma.trail.res	Time-Series [1:297] from 1999 to 2024: 36.2 36.6 37 36.9 36.5 ...
tot.trend.seas.res	Time-Series [1:300] from 1999 to 2024: 34.8 35.6 36.9 37.4 36.3 ...
train.fst.2level	Time-Series [1:297] from 1999 to 2024: 2.16 2.97 3.05 4.86 4 ...
train.ts	Time-Series [1:231] from 1999 to 2018: 0.883 1.297 2.167 3.367 2.735 ...
valid.ts	Time-Series [1:60] from 2018 to 2023: 65.2 72.2 76.8 73.5 79.6 ...
z.stat	-0.906250000000004

 # See summary of linear trend equation and associated parameters.

summary(lin.season)

```
> # See summary of linear trend equation and associated parameters.
> summary(lin.season)

Call:
tslm(formula = Amazon.ts ~ trend + season)

Residuals:
    Min      1Q  Median      3Q     Max 
-40.338 -24.717  -3.177  16.730  86.056 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -35.86983    6.68291  -5.367 1.65e-07 ***
trend         0.46609    0.01999  23.315  < 2e-16 ***
season2      -0.03707    8.47446  -0.004    0.997    
season3      -0.84423    8.47453  -0.100    0.921    
season4       0.58244    8.47465   0.069    0.945    
season5      -0.36816    8.47481  -0.043    0.965    
season6       0.75255    8.47502   0.089    0.929    
season7       2.66659    8.47528   0.315    0.753    
season8       2.92502    8.47559   0.345    0.730    
season9       2.45169    8.47594   0.289    0.773    
season10      1.53116    8.47465   0.181    0.857    
season11      0.60675    8.47453   0.072    0.943    
season12     -0.25237    8.47446  -0.030    0.976    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 29.96 on 287 degrees of freedom
Multiple R-squared:  0.6553,	Adjusted R-squared:  0.6409 
F-statistic: 45.47 on 12 and 287 DF,  p-value: < 2.2e-16
```

From above table we did, two-level model is fit for the entire data set of Amazon's stock price. The model includes a linear trend and seasonality regression, as well as an autoregressive component (AR(1)) for regression residuals.

The tslm() function is used to create the linear trend and seasonality model, and the resulting model summary is printed using the summary() function. The output shows the estimated coefficients for the intercept, trend, and seasonality terms, as well as their standard errors, t-values, and p-values.

The R-squared value of the model is 0.6553, indicating that the model explains about 65.53% of the variation in the data. The F-statistic is 45.47 with a very low p-value, suggesting that the overall model is significant.

 # Apply forecast() function to make predictions with linear trend and seasonal
 # model into the future 12 months.
lin.season.pred<- forecast(lin.season, h = 12, level = 0)
lin.season.pred

Amazon.Data	300 obs. of 2 variables
future12.df	12 obs. of 3 variables
HW.ZZZ	List of 19
HW.ZZZ.pred	List of 10
lin.season	List of 16
lin.season.pred	List of 11
Stock.ar1	List of 18
Stock.stl	List of 8
tot.ma.trail.res.pred	List of 10
tot.trend.seas	List of 16
tot.trend.seas.pred	List of 11
Values	
alpha	0.05
Amazon.ts	Time-Series [1:300] from 1999 to 2024: 0.883 1.297 2.167 3.367 2.735 ...
ar1	0.9971
diff.stock.price	Time-Series [1:299] from 1999 to 2024: 0.414 0.87 1.2 -0.632 0.628 ...
nTrain	240
null_mean	1
nValid	60
p.value	0.182401771838493
s.e.	0.0032
tot.fst.2level	Time-Series [1:12] from 2024 to 2025: 138 141 143 146 149 ...
tot.ma.trail.res	Time-Series [1:297] from 1999 to 2024: 36.2 36.6 37 36.9 36.5 ...
tot.trend.seas.res	Time-Series [1:300] from 1999 to 2024: 34.8 35.6 36.9 37.4 36.3 ...
train.fst.2level	Time-Series [1:297] from 1999 to 2024: 2.16 2.97 3.05 4.86 4 ...
train.ts	Time-Series [1:231] from 1999 to 2018: 0.883 1.297 2.167 3.367 2.735 ...
valid.ts	Time-Series [1:60] from 2018 to 2023: 65.2 72.2 76.8 73.5 79.6 ...
z.stat	-0.906250000000004

```
> # Apply forecast() function to make predictions with linear trend and seasonal
> # model into the future 12 months of 2023-2024
> lin.season.pred <- forecast(lin.season, h = 12, level = 0)
> lin.season.pred
         Point Forecast     Lo 0     Hi 0
Oct 2023       105.9531 105.9531 105.9531
Nov 2023       105.4948 105.4948 105.4948
Dec 2023       105.1018 105.1018 105.1018
Jan 2024       105.8202 105.8202 105.8202
Feb 2024       106.2493 106.2493 106.2493
Mar 2024       105.9082 105.9082 105.9082
Apr 2024       107.8009 107.8009 107.8009
May 2024       107.3164 107.3164 107.3164
Jun 2024       108.9032 108.9032 108.9032
Jul 2024       111.2833 111.2833 111.2833
Aug 2024       112.0079 112.0079 112.0079
Sep 2024       112.0006 112.0006 112.0006
```

The output shows the point forecast for the next 12 months (October 2023 to September 2024) based on the linear trend and seasonal model. The "Lo 0" and "Hi 0" columns indicate the upper and lower bounds of the 0% prediction interval, which is equivalent to the point forecast in this case since the level parameter was set to 0.

According to this model, the Amazon stock price is expected to increase slightly in the coming months, reaching a peak in August 2024 before declining slightly again. However, it's important to keep in mind that these are only predictions and that the actual stock price may differ from these forecasts.

```
# Use Arima() function to fit AR(1) model for regression residuals.
# The ARIMA model order of order = c(1,0,0) gives an AR(1) model.
# Use forecast() function to make prediction of residuals into
# the future 12 months of 2023-2024
residual.ar1 <- Arima(lin.season$residuals, order = c(1,0,0))
residual.ar1.pred <- forecast(residual.ar1, h = 12, level = 0)
residual.ar1.pred
```

Amazon.Data	300 obs. of 2 variables
future12.df	12 obs. of 3 variables
HW.ZZZ	List of 19
HW.ZZZ.pred	List of 10
lin.season	List of 16
lin.season.pred	List of 11
Stock.ar1	List of 18
Stock.stl	List of 8
tot.ma.trail.res.pred	List of 10
tot.trend.seas	List of 16
tot.trend.seas.pred	List of 11
Values	
alpha	0.05
Amazon.ts	Time-Series [1:300] from 1999 to 2024: 0.883 1.297 2.167 3.367 2.735 ...
ar1	0.9971
diff.stock.price	Time-Series [1:299] from 1999 to 2024: 0.414 0.87 1.2 -0.632 0.628 ...
nTrain	240
null_mean	1
nValid	60
p.value	0.182401771838493
s.e.	0.0032
tot.fst.2level	Time-Series [1:12] from 2024 to 2025: 138 141 143 146 149 ...
tot.ma.trail.res	Time-Series [1:297] from 1999 to 2024: 36.2 36.6 37 36.9 36.5 ...
tot.trend.seas.res	Time-Series [1:300] from 1999 to 2024: 34.8 35.6 36.9 37.4 36.3 ...
train.fst.2level	Time-Series [1:297] from 1999 to 2024: 2.16 2.97 3.05 4.86 4 ...
train.ts	Time-Series [1:231] from 1999 to 2018: 0.883 1.297 2.167 3.367 2.735 ...
valid.ts	Time-Series [1:60] from 2018 to 2023: 65.2 72.2 76.8 73.5 79.6 ...
z.stat	-0.906250000000004

```
> residual.ar1 <- Arima(lin.season$residuals, order = c(1,0,0))
> residual.ar1.pred <- forecast(residual.ar1, h = 12, level = 0)
> residual.ar1.pred
         Point Forecast     Lo 0     Hi 0
Oct 2023       31.56488 31.56488 31.56488
Nov 2023       31.31256 31.31256 31.31256
Dec 2023       31.06340 31.06340 31.06340
Jan 2024       30.81736 30.81736 30.81736
Feb 2024       30.57441 30.57441 30.57441
Mar 2024       30.33451 30.33451 30.33451
Apr 2024       30.09761 30.09761 30.09761
May 2024       29.86369 29.86369 29.86369
Jun 2024       29.63270 29.63270 29.63270
Jul 2024       29.40461 29.40461 29.40461
Aug 2024       29.17938 29.17938 29.17938
Sep 2024       28.95698 28.95698 28.95698
```

These are the predicted values of the regression residuals using an AR(1) model. These predictions are for the same 12 months from October 2023 to September 2024 as the previous forecast. The "Point Forecast" column gives the predicted values of the residuals, and the "Lo 0" and "Hi 0" columns give the upper and lower bounds of the prediction interval, respectively.

 # Use summary() to identify parameters of AR(1) model.
summary(residual.ar1)

```
> # Use summary() to identify parameters of AR(1) model.
> summary(residual.ar1)
Series: lin.season$residuals
ARIMA(1,0,0) with non-zero mean

Coefficients:
         ar1     mean
      0.9875  11.4475
s.e.  0.0081  18.1711

sigma^2 = 22.2:  log likelihood = -891.52
AIC=1789.04   AICc=1789.12   BIC=1800.15

Training set error measures:
                     ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set -0.1419506 4.695648 2.476047 0.7877519 32.83916 0.2257669 0.07668675
```

The AR(1) model that was fitted for the regression residuals has the following parameters:

AR coefficient (AR1): 0.9875

Mean: 11.4475

The standard errors for these parameters are also provided.

The estimated variance of the error term is 22.2. The log likelihood of the model is -891.52, and various information criteria are provided as well, including AIC (Akaike information criterion), AICc (corrected AIC), and BIC (Bayesian information criterion).

The training set error measures indicate the model's performance on the data that was used to fit the model. The mean error (ME) is -0.1419506, the root mean squared error (RMSE) is 4.695648, and the mean absolute error (MAE) is 2.476047. The mean percentage error (MPE) is 0.7877 and the mean absolute percentage error (MAPE) is 32.839%. The MASE (mean absolute scaled error) is 0.2257669, and the ACF1 (autocorrelation of residuals at lag 1) is 0.076686.

 # Create two-level model's forecast for the entire data set (training
 # period).
train.lin.season.ar1.pred <- lin.season$fitted + residual.ar1$fitted
train.lin.season.ar1.pred

Amazon.Data	300 obs. of 2 variables
future12.df	12 obs. of 3 variables
HW.ZZZ	List of 19
HW.ZZZ.pred	List of 10
lin.season	List of 16
lin.season.pred	List of 11
residual.ar1	List of 18
residual.ar1.pred	List of 10
Stock.ar1	List of 18
Stock.stl	List of 8
tot.ma.trail.res.pred	List of 10
tot.trend.seas	List of 16
tot.trend.seas.pred	List of 11
Values	
alpha	0.05
Amazon.ts	Time-Series [1:300] from 1999 to 2024: 0.883 1.297 2.167 3.367 2.735 ...
ar1	0.9971
diff.stock.price	Time-Series [1:299] from 1999 to 2024: 0.414 0.87 1.2 -0.632 0.628 ...
nTrain	240
null_mean	1
nValid	60
p.value	0.182401771838493
s.e.	0.0032
tot.fst.2level	Time-Series [1:12] from 2024 to 2025: 138 141 143 146 149 ...
tot.ma.trail.res	Time-Series [1:297] from 1999 to 2024: 36.2 36.6 37 36.9 36.5 ...
tot.trend.seas.res	Time-Series [1:300] from 1999 to 2024: 34.8 35.6 36.9 37.4 36.3 ...
train.fst.2level	Time-Series [1:297] from 1999 to 2024: 2.16 2.97 3.05 4.86 4 ...
train.lin.season.ar1.pred	Time-Series [1:300] from 1999 to 2024: -2.797 0.132 0.601 2.566 3.471 ...
train.ts	Time-Series [1:231] from 1999 to 2018: 0.883 1.297 2.167 3.367 2.735 ...
valid.ts	Time-Series [1:60] from 2018 to 2023: 65.2 72.2 76.8 73.5 79.6 ...
z.stat	-0.906250000000004

```
> train.lin.season.ar1.pred
             Jan          Feb          Mar          Apr          May          Jun          Jul          Aug          Sep          Oct          Nov          Dec
1998                                                                                                                        -2.79702778   0.13233401   0.60067279
1999   2.56633090   3.47085110   2.08205912   4.93174427   3.75327876   4.64147245   4.88690718   3.47346356   2.39431888   2.54541587   3.26064087   3.15230456
2000   5.04686575   3.59235038   3.02512290   4.95943511   2.11710399   4.03126584   4.45740527   2.39421473   1.63994351   1.48097889   0.91250936   0.96216623
2001   1.61055626   1.16522229   0.19412476   2.29334234   0.05928493   2.24104773   2.97922367   1.38013808   0.44614962  -0.15223351  -0.15291508  -0.03907348
2002   1.18796654   0.94999862   0.23662745   2.57777208   0.18374659   2.42771919   3.20144360   1.45028957   0.71181767   0.35437413   0.44552609   0.65218868
2003   1.78739517   1.48326760   0.71854866   3.12190310   0.83452300   3.17921625   4.15833178   2.70144004   2.14169791   1.95804687   2.37110992   2.27067328
2004   3.31009641   3.15606235   1.97661133   4.05509230   1.90398131   3.80728114   4.99080034   3.14188804   2.01040812   1.69443779   1.57526134   1.63084295
2005   2.85195823   2.66237571   1.58759517   3.73618561   1.34117266   3.45085107   4.26901106   2.66795055   1.94231559   1.61153340   1.90610157   2.01105605
2006   3.32696724   2.86978378   1.78611612   3.90113301   1.54265599   3.49730359   4.36384899   2.65021834   1.64612042   1.38544770   1.49338643   1.87285405
2007   2.93202633   2.56272658   1.88885371   4.07792990   2.10061141   5.06147803   6.17093794   4.74761975   4.14047967   4.24911599   4.40445268   4.05023950
2008   5.57944147   4.81219641   3.59719692   5.62136774   3.68552245   5.77347680   6.71704387   4.71211331   4.47526973   3.73864258   2.75642842   2.22052309
2009   3.56309562   3.42189851   3.20324347   5.69053178   3.83269562   5.86041507   6.94518854   5.32931617   4.63626730   4.31239731   5.02367248   6.41429641
2010   7.87437637   7.22365127   5.96816603   8.82476338   6.99260112   8.39526007   8.92903235   7.04654650   6.78007920   7.36861979   8.02262243   8.47417443
2011  10.10212214   9.98413027   9.15670779  10.78194585   9.18381096  11.94521121  12.43948520  11.96610096  10.47123674  11.17925966  11.34743329  10.39087114
2012  10.42605017  10.16734563   9.30684333  11.85831637   9.80496365  13.15686339  13.85455358  12.46875877  12.49655375  12.87674084  12.33098283  11.85728731
2013  13.88811760  14.39271759  13.39100888  15.68673047  13.23049517  15.26710173  16.64317512  16.22113866  15.15384346  15.34443875  16.36972534  18.35282367
2014  20.80628570  20.71051163  17.89484381  20.61122222  16.17019768  17.30920552  19.20961875  18.30669057  17.00141816  16.79011831  15.59955065  16.13602399
2015  16.82786214  16.22354754  19.01071262  21.27779794  19.85938030  23.54010332  24.61697713  25.31076746  26.55412242  26.18285448  28.32890515  32.93666507
2016  34.65842122  31.03939938  26.76624454  30.96282149  30.73428993  36.90636780  38.67643666  38.29291104  38.71272659  38.44020438  41.27056298  38.31242482
2017  39.38640896  41.15890416  41.83686037  45.01043153  45.17787036  50.02083066  52.29252785  51.53475590  49.01886079  48.47083799  49.97504140  56.83617895
2018  59.49009482  65.87994050  71.97113782  78.82496246  73.18022742  81.25809795  87.31076926  89.88665326  94.67201768  97.71043991  88.95397417  80.87488228
2019  79.09313981  82.27374917  81.07059917  88.00045802  92.62527925  95.02625757  94.99717917  98.83207127  89.78513337  89.54025997  87.14301945  88.39342444
2020  90.05395823  94.64663157 103.03959193  95.30077210 110.40745292 120.87990747 132.49690085 152.55571693 161.53580009 157.10015777 160.50521654 155.94246021
2021 159.78922454 160.00054970 162.12710582 154.76450724 166.28968031 163.50058289 169.75963278 180.34873304 164.71051745 170.20375860 165.41884626 175.02017791
2022 170.87643750 155.45137580 152.86479807 155.52390396 150.76491816 113.83744583 114.91135727 117.14360613 137.47944861 122.60539483 113.83115486  94.07340510
2023  84.32584433 102.78735591  98.85882367  98.48347317 102.98979952 113.42078087 128.66075520 131.48844082 135.31399671
```

The code creates a forecast for the entire training set using the AR(1) model with seasonal adjustment. The forecast is created by adding the fitted values of the seasonal component and the fitted values of the residual AR(1) model. The resulting train.lin.season.ar1.pred object contains the forecasted values for each month of each year in the training set.

 # Identify forecast for the future 12 periods of 2023-2024 as sum of
 # linear trend and seasonality model and AR(1) model for residuals.
lin.season.ar1.pred <- lin.season.pred$mean + residual.ar1.pred$mean
lin.season.ar1.pred

```
> lin.season.ar1.pred <- lin.season.pred$mean + residual.ar1.pred$mean
> lin.season.ar1.pred
          Jan      Feb      Mar      Apr      May      Jun      Jul      Aug      Sep      Oct      Nov
2023                                                                                     137.5180 136.8074
2024 136.6376 136.8237 136.2427 137.8985 137.1801 138.5359 140.6879 141.1872 140.9576
          Dec
2023 136.1652
2024
```

Based on the two-level model, the forecast for the future 12 periods (October 2023- September 2024) is given by lin.season.ar1.pred. This forecast is obtained by adding the predicted values from the linear trend and seasonality model (lin.season.pred$mean) and the predicted values from the AR(1) model for the residuals (residual.ar1.pred$mean).

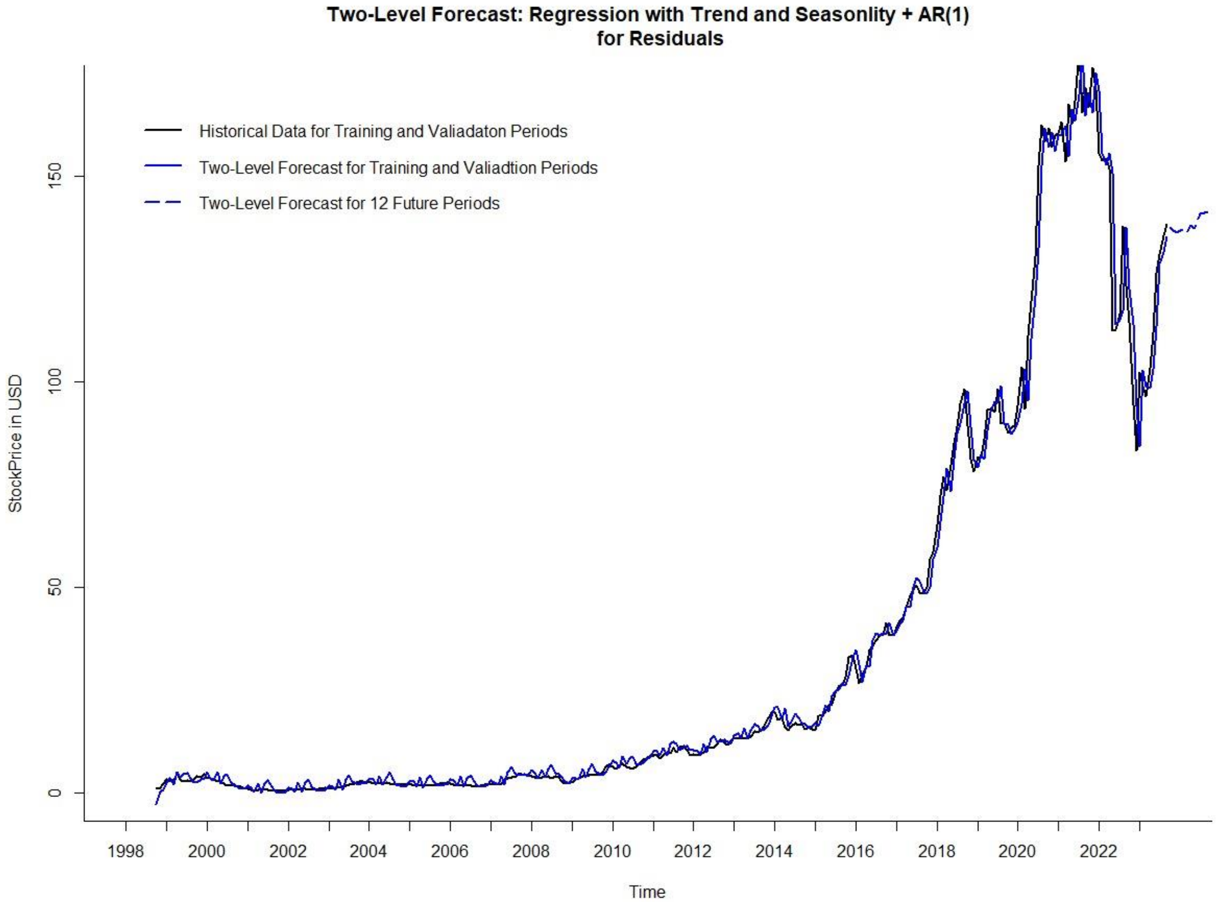
 # Create a data table with linear trend and seasonal forecast
 # for 12 future periods,
 # AR(1) model for residuals for 12 future periods, and combined
 # two-level forecast for 12 future periods.
table.df<- round(data.frame(lin.season.pred$mean,

residual.ar1.pred$mean, lin.season.ar1.pred),3)
names(table.df) <- c("Reg.Forecast", "AR(1)Forecast","Combined.Forecast")
table.df

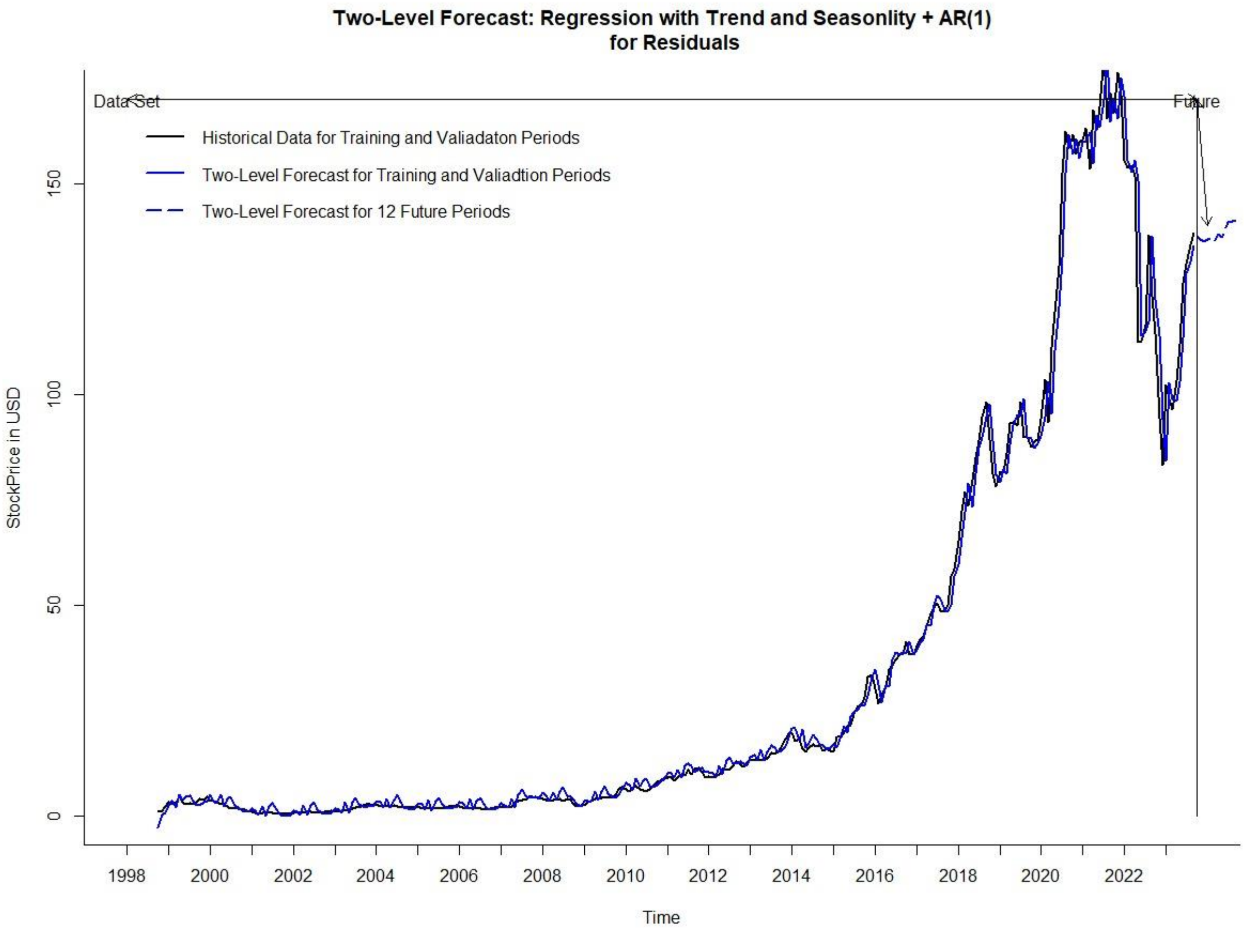
```
> # Create a data table with linear trend and seasonal forecast
> # for 12 future periods,
> # AR(1) model for residuals for 12 future periods, and combined
> # two-level forecast for 12 future periods.
> table.df <- round(data.frame(lin.season.pred$mean,
+                             residual.ar1.pred$mean, lin.season.ar1.pred),3)
> names(table.df) <- c("Reg.Forecast", "AR(1)Forecast","Combined.Forecast")
> table.df
   Reg.Forecast AR(1)Forecast Combined.Forecast
1       105.953        31.565           137.518
2       105.495        31.313           136.807
3       105.102        31.063           136.165
4       105.820        30.817           136.638
5       106.249        30.574           136.824
6       105.908        30.335           136.243
7       107.801        30.098           137.899
8       107.316        29.864           137.180
9       108.903        29.633           138.536
10      111.283        29.405           140.688
11      112.008        29.179           141.187
12      112.001        28.957           140.958
```

Great! You now have a data table with three columns: one for the linear trend and seasonal forecast, one for the AR(1) model forecast, and one for the combined forecast using both models. The table provides the forecasted values for the 12 future periods, from October 2023 to September 2024. It looks like the combined forecast values are obtained by adding the linear trend and seasonal forecast and the AR(1) model forecast for each month.

```
# Plot historical data, predictions for historical data, and forecast
# for 12 future periods.
plot(Amazon.ts,
xlab = "Time", ylab = "StockPrice in USD",
ylim = c(0, 170), xaxt = "n",
bty = "l", xlim = c(1998, 2023.75), lwd = 2,
    main = "Two-Level Forecast: Regression with Trend and Seasonlity + AR(1)
    for Residuals")
axis(1, at = seq(1998, 2023.75, 1), labels = format(seq(1998, 2023.75, 1)))
lines(lin.season$fitted + residual.ar1$fitted, col = "blue", lwd = 2)
lines(lin.season.ar1.pred, col = "blue", lty = 5, lwd = 2)
legend(1998,170, legend = c("Historical Data for Training and Valiadaton Periods",
                       "Two-Level Forecast for Training and Valiadtion Periods",
                       "Two-Level Forecast for 12 Future Periods"),
      col = c("black", "blue" , "blue"),
lty = c(1, 1, 5), lwd =c(2, 2, 2), bty = "n")
```



 # Plot on chart vertical lines and horizontal arrows describing
 # entire data set and future prediction intervals.
lines(c(2023.75, 2023.75), c(0, 170))
text(1998, 170, "Data Set")
text(2023.75, 170, "Future")
arrows(1998, 170, 2023.75, 170, code = 3, length = 0.1,
lwd = 1, angle = 30)
arrows(2023.75, 170, 2024, 170, code = 3, length = 0.1,
lwd = 1, angle = 30)



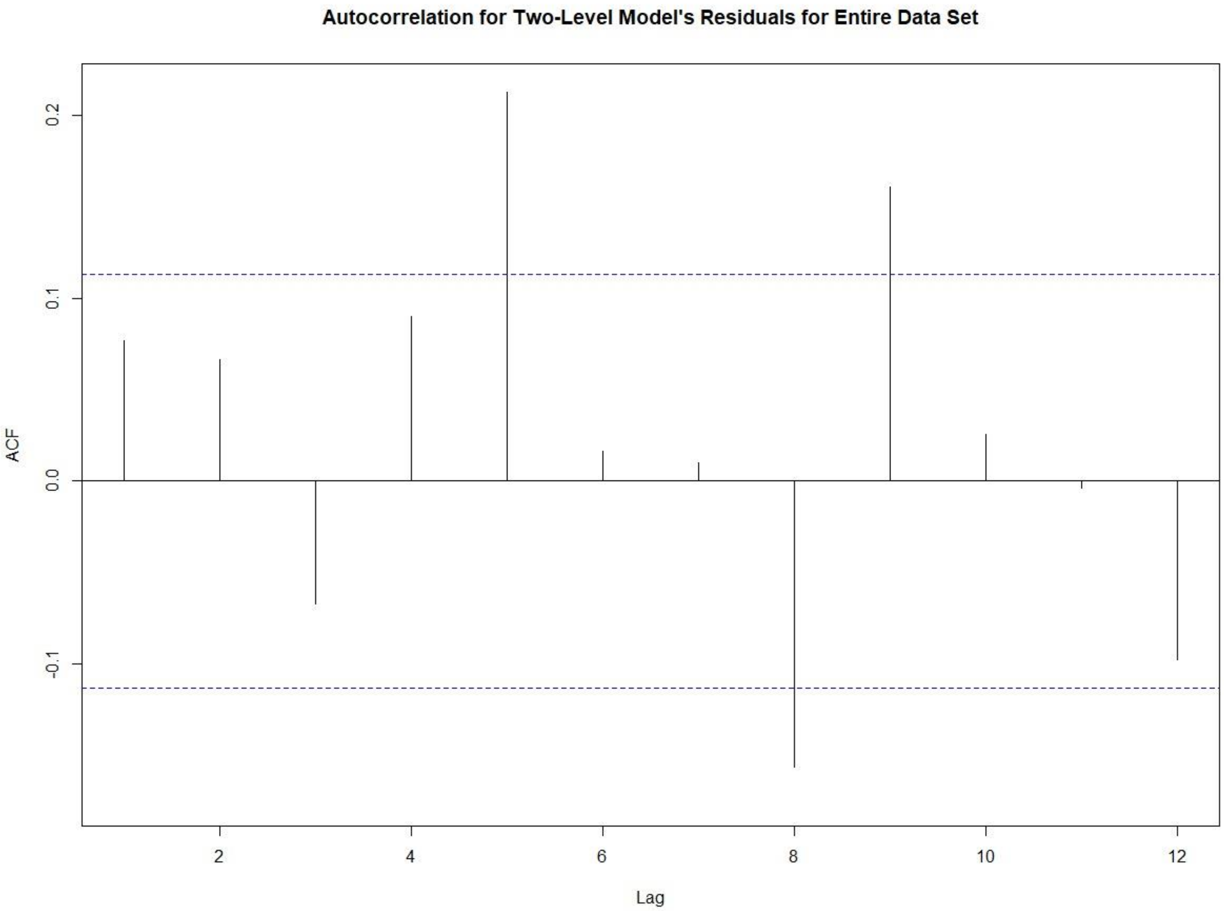
We are creating a plot of the historical data, the fitted values from the two-level forecast model for the historical data, and the forecasted values for the 12 future periods. The x-axis shows the time period from 1998 to 2024, with the training and validation data shown in black and the forecasted data shown in blue.

The plot() function is used to create the initial plot with appropriate labels and limits for the axes. The axis() function is used to label the x-axis with the appropriate time periods.

The lines() function is used to add two lines to the plot: one for the fitted values from the two-level forecast model for the historical data, and one for the forecasted values for the 12 future periods.
The legend() function is used to add a legend to the plot that identifies the different lines and colors.

Finally, the lines() and text() functions are used to add vertical lines and arrows to the plot that describe the entire data set and future prediction intervals.

 # Use Acf() function to identify autocorrelation for residuals of
 # two-level model for different lags (up to maximum of 12).
Acf((Amazon.ts - train.lin.season.ar1.pred), lag.max = 12,
main = "Autocorrelation for Two-Level Model's Residuals for Entire Data Set")



This code generates a plot of the autocorrelation function (ACF) of the residuals of the two-level model that was built using linear regression with trend and seasonality, and an autoregressive (AR(1)) model for the residuals. The Acf() function is used with the residual time series obtained by subtracting the two-level model's predicted values from the original Amazon stock price time series. The lag.max parameter specifies the maximum number of lags to include in the plot. The resulting plot shows the autocorrelation at different lags, which can help identify whether there is any remaining autocorrelation in the residuals that the model did not capture. The plot can be used to assess the adequacy of the model and to guide the selection of additional model components, such as higher-order autoregressive terms.

###**Model Four:**

FIT AUTO ARIMA MODEL FOR ENTIRE DATA SET.
FORECAST AND PLOT DATA.

 # Use auto.arima() function to fit ARIMA model for entire data set.
 # use summary() to show auto ARIMA model and its parameters
 # for entire data set.

auto.arima<- auto.arima(Amazon.ts)
summary(auto.arima)

```
> # Use auto.arima() function to fit ARIMA model for entire data set.
> # use summary() to show auto ARIMA model and its parameters
> # for entire data set.
> auto.arima <- auto.arima(Amazon.ts)
> summary(auto.arima)
Series: Amazon.ts
ARIMA(0,1,0) with drift

Coefficients:
       drift
      0.4593
s.e.  0.2783

sigma^2 = 23.24:  log likelihood = -894.09
AIC=1792.19   AICc=1792.23   BIC=1799.59

Training set error measures:
                       ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 1.412174e-06 4.805073 2.217649 -12.91446 16.67734 0.2207627 0.07199954
```

The output shows that the model is an ARIMA(0,1,0) model with drift coefficients of 0.4593 for AR(1) . The estimated variance of the model is 23.24&the model also includes error measure values for our estimation for best models in future by comparing these values.

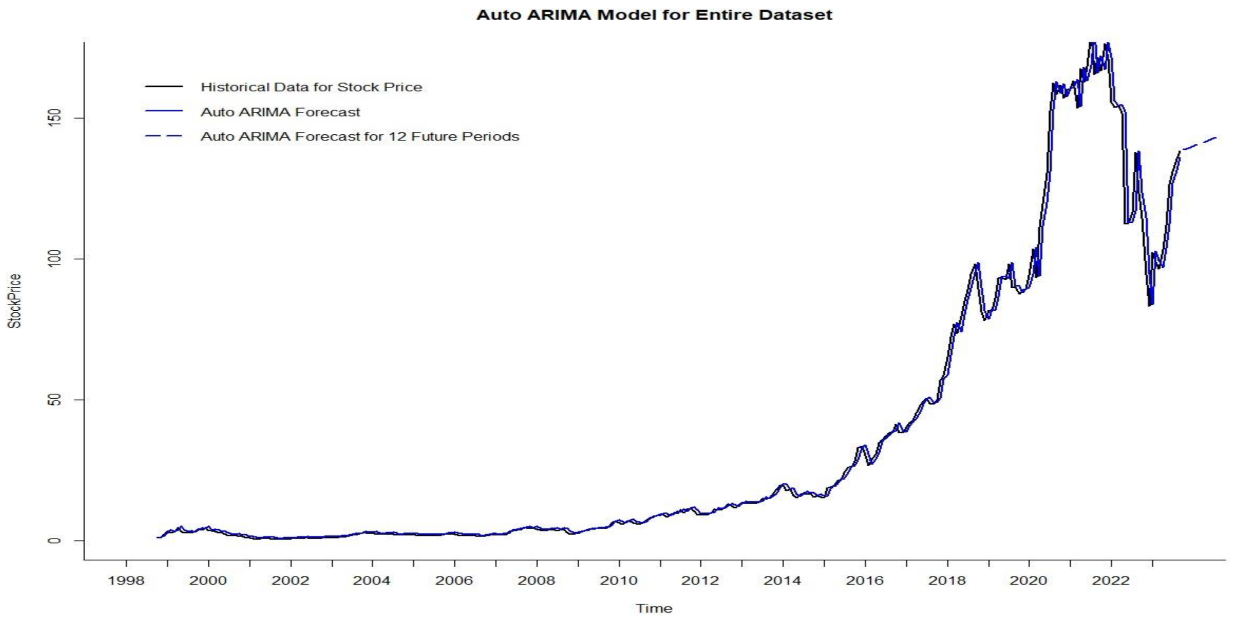
 # Apply forecast() function to make predictions for ts with
 # auto ARIMA model for the future 12 periods.
auto.arima.pred<- forecast(auto.arima, h = 12, level = 0)
auto.arima.pred

```
> # Apply forecast() function to make predictions for ts with
> # auto ARIMA model for the future 12 periods.
> auto.arima.pred <- forecast(auto.arima, h = 12, level = 0)
> auto.arima.pred
         Point Forecast     Lo 0     Hi 0
Oct 2023       138.6873 138.6873 138.6873
Nov 2023       139.1467 139.1467 139.1467
Dec 2023       139.6060 139.6060 139.6060
Jan 2024       140.0654 140.0654 140.0654
Feb 2024       140.5247 140.5247 140.5247
Mar 2024       140.9841 140.9841 140.9841
Apr 2024       141.4434 141.4434 141.4434
May 2024       141.9028 141.9028 141.9028
Jun 2024       142.3621 142.3621 142.3621
Jul 2024       142.8215 142.8215 142.8215
Aug 2024       143.2808 143.2808 143.2808
Sep 2024       143.7402 143.7402 143.7402
```

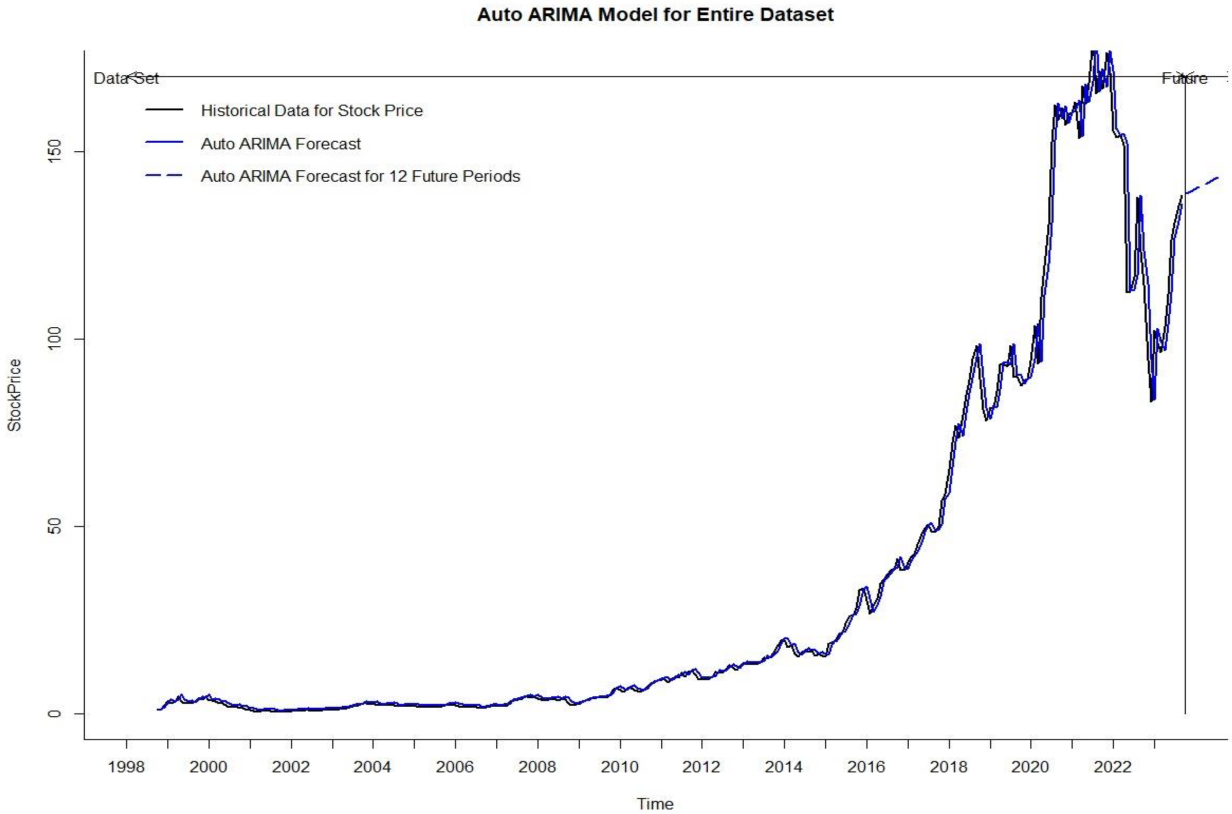
Here, we have made predictions for the next 12 periods using the h argument in the forecast() function.

The output of auto.arima.pred shows the point forecast (predicted values) for the next 12 periods, along with a zero confidence interval. The point forecast represents the most likely value for the time series at each future time point. The upper and lower bounds of the prediction interval provide a range of values within which the actual future value is likely to fall with a given confidence interval.

```
# Plot historical data, predictions for historical data, and Auto ARIMA
# forecast for 12 future periods.
plot(Amazon.ts,
xlab = "Time", ylab = "StockPrice",
ylim = c(0, 170), xaxt = "n",
bty = "l", xlim = c(1998, 2023.75), lwd = 2,
    main = "Auto ARIMA Model for Entire Dataset")
axis(1, at = seq(1998, 2023.75, 1), labels = format(seq(1998, 2023.75, 1)))
lines(auto.arima$fitted, col = "blue", lwd = 2)
lines(auto.arima.pred$mean, col = "blue", lty = 5, lwd = 2)
legend(1998,170, legend = c("Historical Data for Stock Price",
                        "Auto ARIMA Forecast",
                        "Auto ARIMA Forecast for 12 Future Periods"),
     col = c("black", "blue" , "blue"),
lty = c(1, 1, 5), lwd =c(2, 2, 2), bty = "n")
```




```
#plot on the chart vertical lines and horizontal arrowsdescribing training and future
#prediction intervals.
lines(c(2023.75, 2023.75), c(0, 170))
text(1998, 170, "Data Set")
text(2023.5, 170, "Future")
arrows(1998, 170, 2023.75, 170, code = 3, length = 0.1,
lwd = 1, angle = 30)
arrows(2023.75, 170, 2025, 170, code = 3, length = 0.1,
lwd = 1, angle = 30)
```



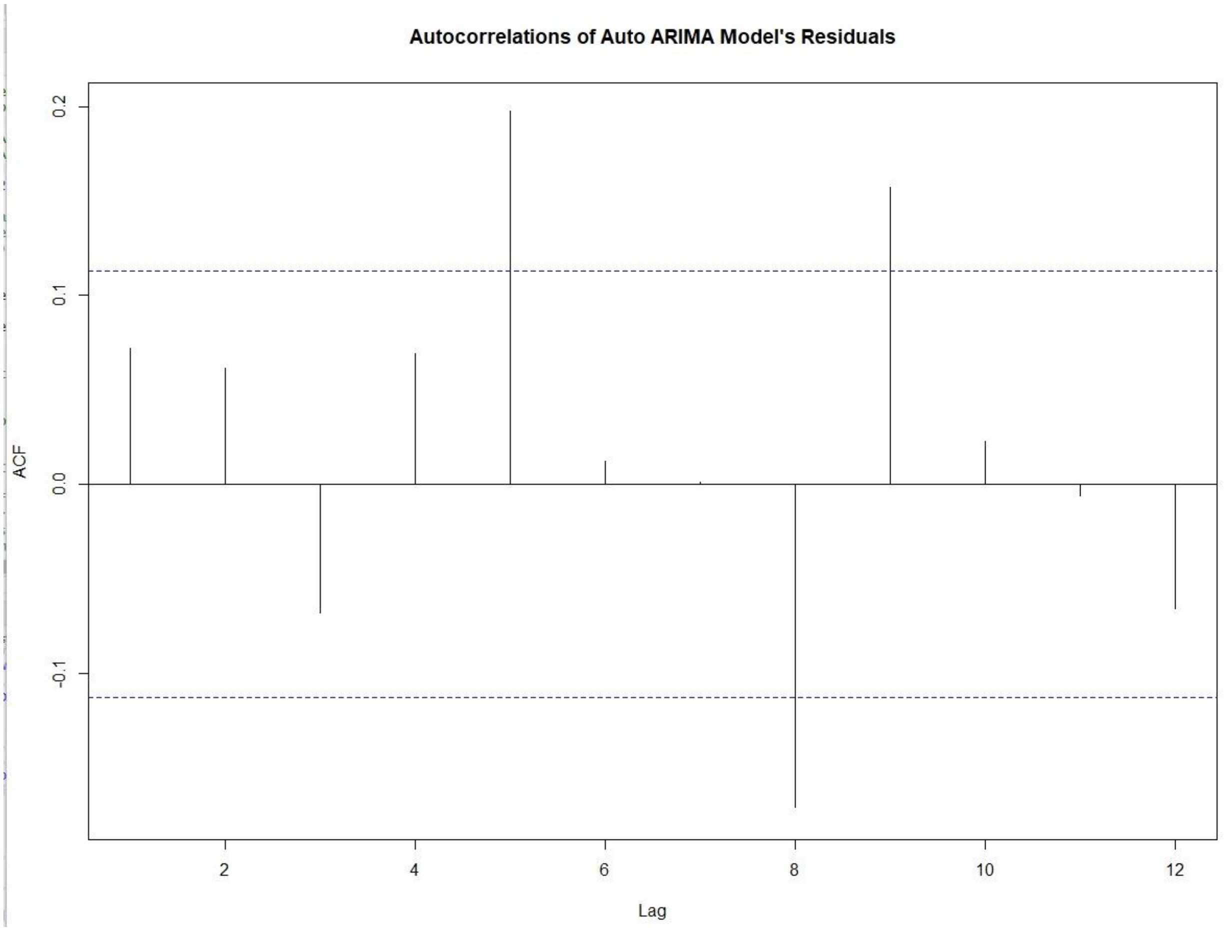
This code generates a plot that shows the historical data for the stock price of Amazon, the predictions for historical data based on the auto ARIMA model, and the auto ARIMA forecast for the next 12 periods.

The plot() function is used to plot the historical data with appropriate labels and limits. The axis() function is used to add x-axis labels with yearly ticks. The lines() function is used twice to add lines to the plot. The first lines() function adds a blue line for the predicted values for the historical data. The second lines() function adds a blue dotted line for the auto ARIMA forecast for the next 12 periods.

The legend() function is used to add a legend to the plot that describes the historical data, the auto ARIMA forecast, and the auto ARIMA forecast for the next 12 periods.

Finally, the lines() and arrows() functions are used to add vertical lines and horizontal arrows that describe the training and future prediction intervals. The text() function is used to add text labels to the chart.

Use Acf() function to create autocorrelation chart of auto ARIMA model residuals.
Acf(auto.arima$residuals, lag.max = 12,
main = "Autocorrelations of Auto ARIMA Model's Residuals")



The output is a plot showing the autocorrelation of the residuals of the auto ARIMA model. The x-axis shows the lag (number of time periods between the observations being compared), and the y-axis shows the correlation coefficient between the residuals at that lag. The blue shaded area represents the confidence interval, and any correlations outside of this area are considered statistically significant. The main title of the plot indicates that it shows the autocorrelations of the auto ARIMA model's residuals. This plot is useful for checking whether the residuals exhibit any significant autocorrelation, which would indicate that there is still some information in the data that the model has not captured. If there is significant autocorrelation, it may be necessary to modify the model or add additional terms to better capture the underlying patterns in the data.

Step 8: Evaluate Results and Refine the Model - The final step of the project is to evaluate the forecasting results and refine the model if necessary. The evaluation may involve comparing the forecasted values with the actual values, analysing the forecast errors, and updating the model parameters based on new data or changes in the underlying conditions.

MEASURE FORECAST ACCURACY FOR ENTIRE DATA SET USING VARIOUS METHODS.

Use accuracy() function to identify common accuracy measures for:

(1) Two-Level Model with Linear Trend & Seasonality Regression and Trailing MA for Regression Residuals.

round(accuracy(tot.trend.seas.pred$fitted+tot.ma.trail.res, Amazon.ts), 3)

```
> round(accuracy(tot.trend.seas.pred$fitted+tot.ma.trail.res, Amazon.ts), 3)
             ME  RMSE   MAE     MPE   MAPE ACF1 Theil's U
Test set -0.032 4.651 2.442 -20.026 30.768 0.61     5.328
```

(2) Holt-Winter's Model with Automatic Selection of Model Optionsand Parameters,

round(accuracy(HW.ZZZ.pred$fitted, Amazon.ts), 3)

```
> round(accuracy(HW.ZZZ.pred$fitted, Amazon.ts), 3)
           ME  RMSE   MAE    MPE  MAPE  ACF1 Theil's U
Test set 0.03 4.848 2.112 -1.264 8.768 0.048     0.944
```

(3) Two-Level Model with Linear Trend & Seasonality Regression and Ar(1) Model for Regression Residuals

round(accuracy(lin.season$fitted + residual.ar1$fitted, Amazon.ts), 3)

```
> round(accuracy(lin.season$fitted + residual.ar1$fitted, Amazon.ts), 3)
             ME  RMSE   MAE     MPE   MAPE  ACF1 Theil's U
Test set -0.142 4.696 2.476 -12.365 29.914 0.077     4.384
```

(4) Auto ARIMA Model

round(accuracy(auto.arima.pred$fitted, Amazon.ts), 3)

```
> round(accuracy(auto.arima.pred$fitted, Amazon.ts), 3)
         ME  RMSE   MAE     MPE   MAPE  ACF1 Theil's U
Test set  0 4.805 2.218 -12.914 16.677 0.072      2.05
```

<u>The four models for which the accuracy measures are computed are:</u>

<u>Model 1.</u>Two-Level Model with Linear Trend & Seasonality Regression and Trailing MA for Regression Residuals

<u>Model 2.</u>Holt-Winter's Model with Automatic Selection of Model Options and Parameters

<u>Model 3.</u>Two-Level Model with Linear Trend & Seasonality Regression and Ar(1) Model for Regression Residuals

<u>Model 4.</u>Auto ARIMA Model.

<u>Forecast of Stock Price Using Different Models:</u>

Month & Year	Model One	Model Two	Model Three	Model Four
Oct-23	137.666	138.939	137.518	138.6873
Nov-23	140.747	139.65	136.807	139.1467
Dec-23	143.212	140.361	136.165	139.606
Jan-24	146.237	141.072	136.638	140.065
Feb-24	148.528	141.783	136.824	140.5247
Mar-24	149.69	142.4947	136.243	140.984
Apr-24	152.797	143.206	137.899	141.4434
May-24	153.292	143.917	137.18	141.9028
Jun-24	155.669	144.628	138.536	142.3621
Jul-24	158.688	145.339	140.688	142.8215
Aug-24	159.927	146.0505	141.187	143.2808
Sep-24	160.336	156.7616	140.958	143.7402

Model 1. Two-Level Model with Linear Trend & Seasonality Regression and Trailing MA for Regression Residuals

Model 2. Holt-Winter's Model with Automatic Selection of Model Options and Parameters

Model 3. Two-Level Model with Linear Trend & Seasonality Regression and Ar(1) Model for Regression Residuals

Model 4. Auto ARIMA Model.

Accuracy Measures Using Different Models:

Measures	Model One	Model Two	Model Three	Model Four
ME	-0.032	0.03	-0.142	0
RMSE	4.651	4.848	4.696	4.805
MAE	2.442	2.112	2.476	2.218
MPE	-20.026	-1.264	-12.365	-12.914
MAPE	30.768	8.768	29.914	16.677
ACF1	0.61	0.048	0.077	0.072
Theil's U	5.328	0.944	4.384	2.05

Model 1. Two-Level Model with Linear Trend & Seasonality Regression and Trailing MA for Regression Residuals

Model 2. Holt-Winter's Model with Automatic Selection of Model Options and Parameters

Model 3. Two-Level Model with Linear Trend & Seasonality Regression and Ar(1) Model for Regression Residuals

Model 4. Auto ARIMA Model.

CONCLUSION

Based on the accuracy measures computed, we can compare the performance of each model. The best model would be the one that has the lowest RMSE,MAPE. In this case, the ***Holt-Winter's Model***has the lowest RMSE& MAPE values, indicating that it performs the best among the four models & the second-best model is the **A*uto ARIMA Model***as it has the second lowest values for MAPE & RMSE. Here, RMSE values are almost in the same range. Therefore, we are considering MAPE in this case.

Bibliography and Appendices

Book:

● Shmueli, G. and Lichtendahl Jr., K.C. *Practical Time Series Forecasting with R*, 2nd Edition, Axelrod Schnall Publishers, 2016. ISBN-13: 978-0-9978479-1-8.