## Final Project in MSDS 511

# STOCK PRICE PREDICTION

Reynalyn I. Asilo

# Project Assignment

Develop a model to predict the future prices of stocks based on historical price data, trading volume, and other relevant factors. You can use machine learning algorithms such as ARIMA, LSTM, or Prophet for this project.

# Stock Market

The stock market refers to the collection of markets and exchanges where regular activities of buying, selling, and issuance of shares of publicly-held companies take place. Such financial activities are conducted through institutionalized formal exchanges or over-the-counter (OTC) marketplaces which operate under a defined set of regulations. There can be multiple stock trading venues in a country or a region which allow transactions in stocks and other forms of securities.

# Importance of Stock Market

- Helps companies to raise capital
- Helps create personal wealth
- Serves as an indicator of the state of the economy
- Helps to increase investment

# About the Dataset

Amazon.com, Inc. is an American multinational technology company which focuses on e-commerce, cloud computing, digital streaming, and artificial intelligence. It is one of the Big Five companies in the U.S. information technology industry, along with Google, Apple, Microsoft, and Facebook. The company has been referred to as "one of the most influential economic and cultural forces in the world", as well as the world's most valuable brand.

This dataset provides the history of daily prices of Amazon stock (AMZN). All the column descriptions are provided. Currency is USD.

# About the Dataset

The dataset includes the following columns:

**Date**: The date on which the stock market data was recorded.
**Open**: The opening price of the asset on the given date.
**High**: The highest price of the asset on the given date.
**Low**: The lowest price of the asset on the given date.
**Close***: The closing price of the asset on the given date. Note that this price does not take into account any after-hours trading that may have occurred after the market officially closed.*
**Volume**: The total number of shares of the asset that were traded on the given date.
**Adjusted**: The adjusted closing price of the asset on the given date. This price takes into account any dividends, stock splits, or other corporate actions that may have occurred, which can affect the stock price.

# Loading Required Libraries

```
library(quantmod)
library(forecast)
library(tseries)
library(timeSeries)
library(dplyr)
library(tsfknn)
library(prophet)

# Extracting stock data for Amazon
getSymbols("AMZN", from= "2019-01-01", to = "2024-04-01")

head(AMZN)

tail(AMZN)

# Separating Closing Prices of stocks from data
AMZN_CP = AMZN[,4]
```

# Dataset

```
> head(AMZN)
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2019-01-02   73.2600   77.6680  73.0465    76.9565   159662000       76.9565
2019-01-03   76.0005   76.9000  74.8555    75.0140   139512000       75.0140
2019-01-04   76.5000   79.7000  75.9155    78.7695   183652000       78.7695
2019-01-07   80.1155   81.7280  79.4595    81.4755   159864000       81.4755
2019-01-08   83.2345   83.8305  80.8305    82.8290   177628000       82.8290
2019-01-09   82.6490   83.3900  82.0700    82.9710   126976000       82.9710
```

- Amazon Stock Price Data from January 2, 2019 to March 28, 2024

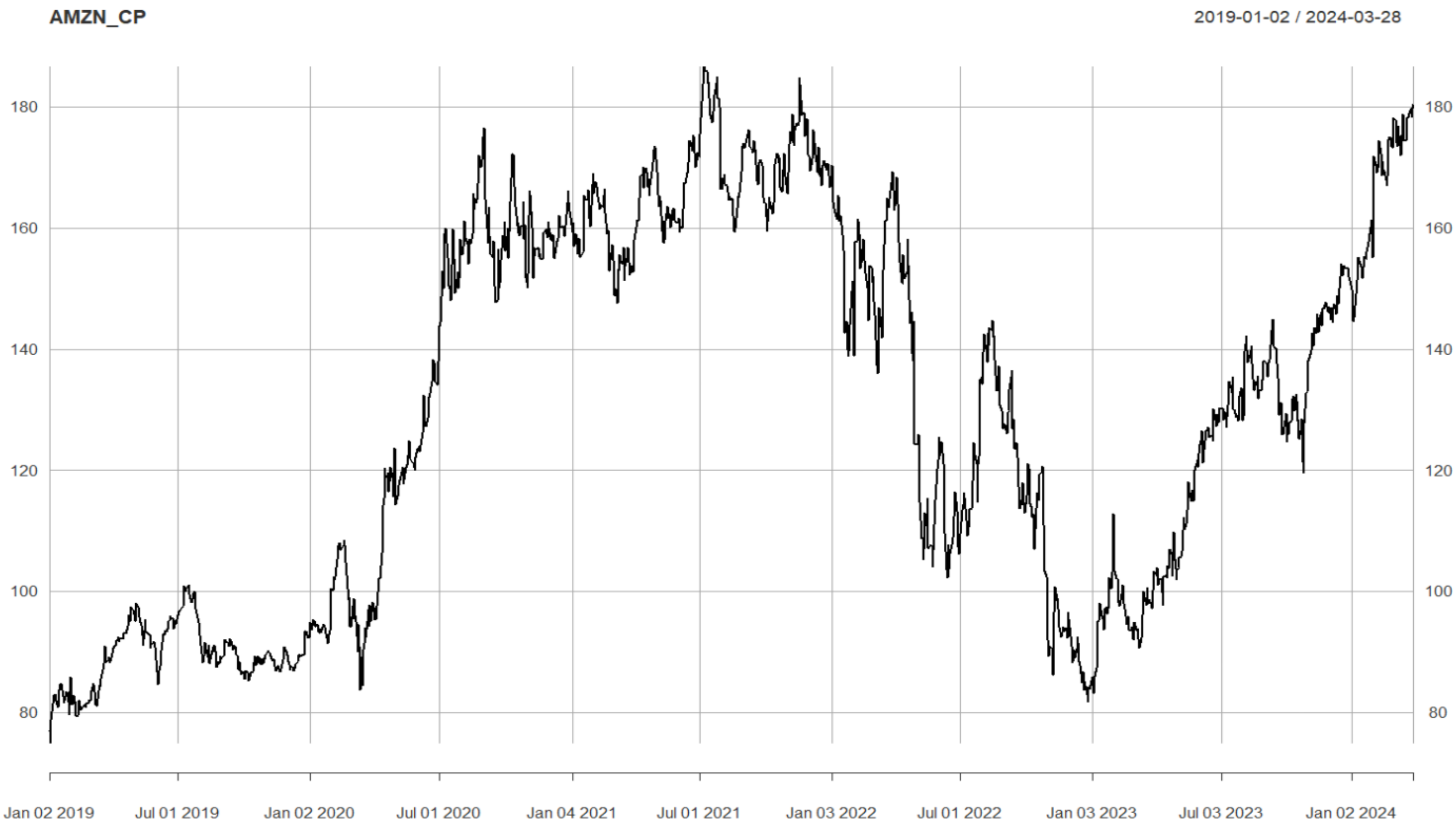
```
> tail(AMZN)
           AMZN.Open AMZN.High AMZN.Low AMZN.Close AMZN.Volume AMZN.Adjusted
2024-03-21    179.99    181.42   178.15     178.15    32824300        178.15
2024-03-22    177.75    179.26   176.75     178.87    27964100        178.87
2024-03-25    178.01    180.99   177.24     179.71    29815500        179.71
2024-03-26    180.15    180.45   177.95     178.30    29659000        178.30
2024-03-27    179.88    180.00   177.31     179.83    33272600        179.83
2024-03-28    180.17    181.70   179.26     180.38    38051600        180.38
```

- Closing price of each day would be used for model training and prediction

# Plotting graph of Amazon Stock Prices to observe the trend

```
library(quantmod)
library(forecast)
library(tseries)
library(timeSeries)
library(dplyr)
library(fGarch)
library(xts)
library(readr)
library(moments)
library(tsfknn)

# Extracting stock data for Amazon
getSymbols("AMZN", from= "2019-01-01", to = "2024-04-01")

head(AMZN)

tail(AMZN)

# Separating Closing Prices of stocks from data
AMZN_CP = AMZN[,4]

# Plotting graph of Amazon Stock Prices to observe the trend
plot(AMZN_CP)
```


```
# Plotting graph of Amazon Stock Prices to observe the trend
plot(AMZN_CP)
```

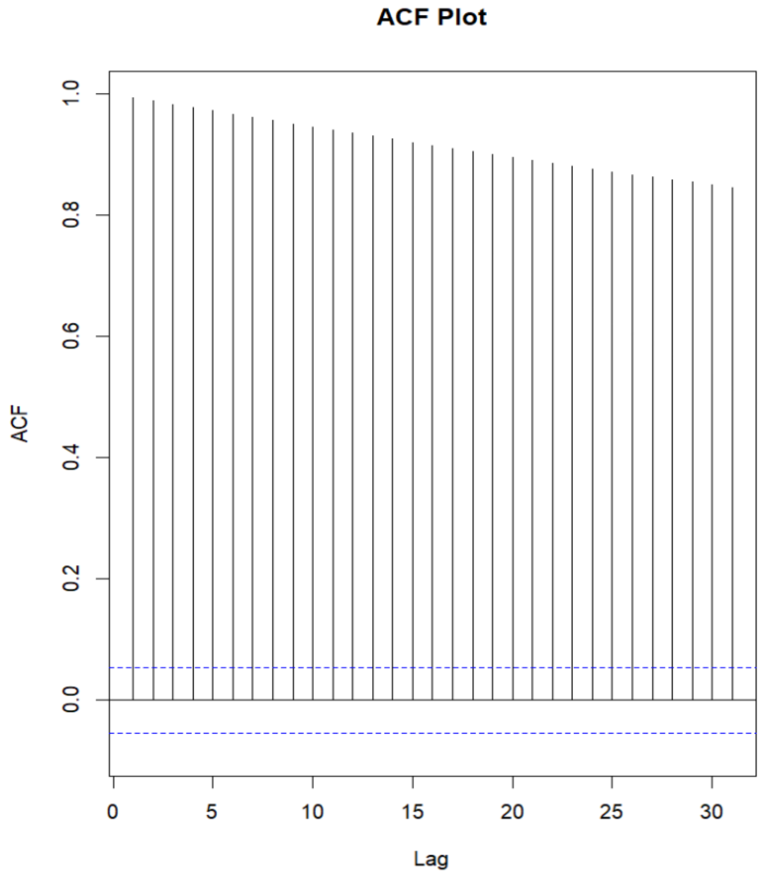



The above graph shows Amazon’s stock price trend

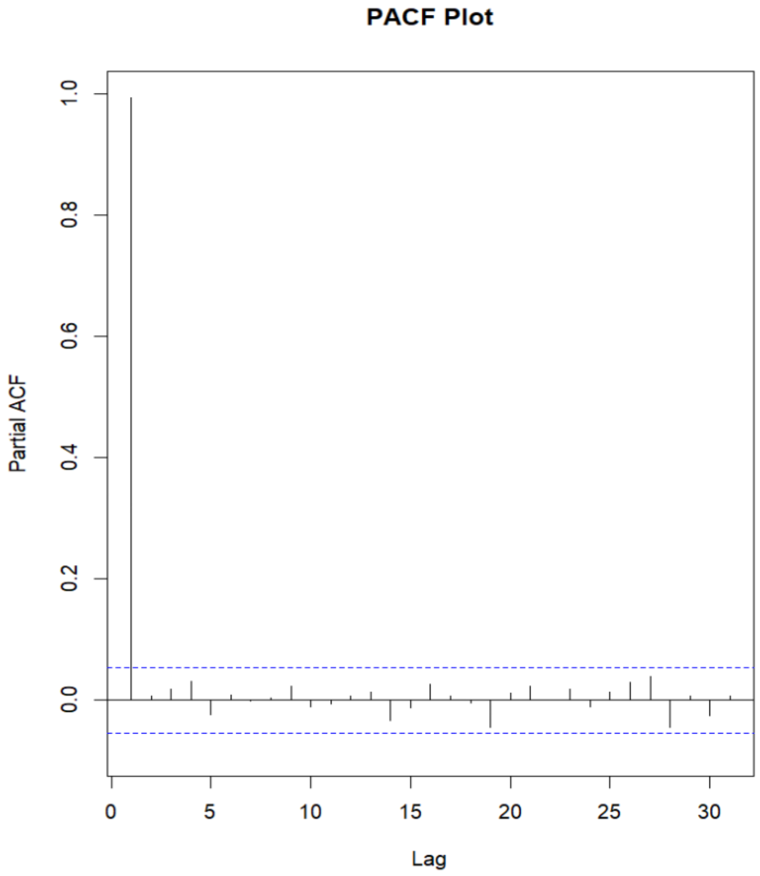
# Plotting the ACF and PACF plot of data

```
# Separating Closing Prices of stocks from data
AMZN_CP = AMZN[,4]

# Plotting graph of Amazon Stock Prices to observe the trend
plot(AMZN_CP)

# Plotting the ACF and PACF plot of data
par(mfrow=c(1,2))
Acf(AMZN_CP, main = 'ACF Plot')
Pacf(AMZN_CP, main = 'PACF Plot')
```


```
# Plotting the ACF and PACF plot of data
par(mfrow=c(1,2))
Acf(AMZN_CP, main = 'ACF Plot')
Pacf(AMZN_CP, main = 'PACF Plot')
```




**ACF Plot**

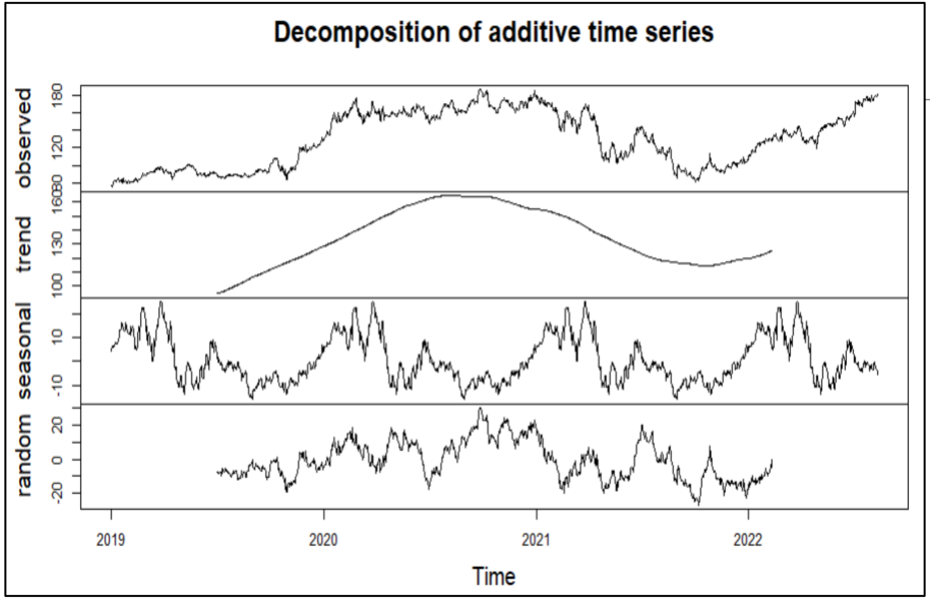



**PACF Plot**

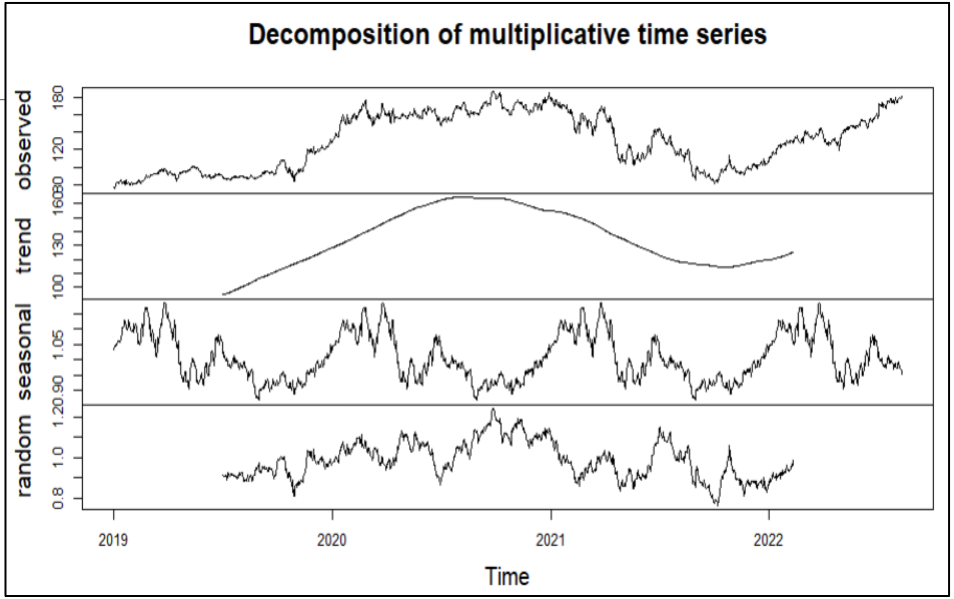
# Plotting Additive and Multiplicative Decomposition

```r
# Separating Closing Prices of stocks from data
AMZN_CP = AMZN[,4]

# Plotting graph of Amazon Stock Prices to observe the trend
plot(AMZN_CP)

# Plotting the ACF and PACF plot of data
par(mfrow=c(1,2))
Acf(AMZN_CP, main = 'ACF Plot')
Pacf(AMZN_CP, main = 'PACF Plot')

# Plotting Additive and Multiplicative Decomposition
AMZN.ts <- ts(AMZN_CP, start=c(2019,1,1), frequency = 365.25)
AMZN.add  <- decompose(AMZN.ts,type = "additive")
plot(AMZN.add)
AMZN.mult <- decompose(AMZN.ts,type = "multiplicative")
plot(AMZN.mult)
```


```r
# Plotting Additive and Multiplicative Decomposition
AMZN.ts <- ts(AMZN_CP, start=c(2019,1,1), frequency = 365.25)
AMZN.add  <- decompose(AMZN.ts,type = "additive")
plot(AMZN.add)
AMZN.mult <- decompose(AMZN.ts,type = "multiplicative")
plot(AMZN.mult)
```




Additive Series Decomposition




Multiplicative Series Decomposition

# Augmented Dickey Fuller Test

```r
# ADF test on Closing Prices
print(adf.test(AMZN_CP))

# Splitting into test and train data
N = length(AMZN_CP)
n = 0.7*N
train = AMZN_CP[1:n, ]
test  = AMZN_CP[(n+1):N,  ]
predlen=length(test)

# Taking log of dataset
logs=diff(log(AMZN_CP), lag =1)
logs = logs[!is.na(logs)]

# Log returns plot
plot(logs, type='l', main= 'Log Returns Plot')

# ADF test on log of Closing Prices
print(adf.test(logs))
```

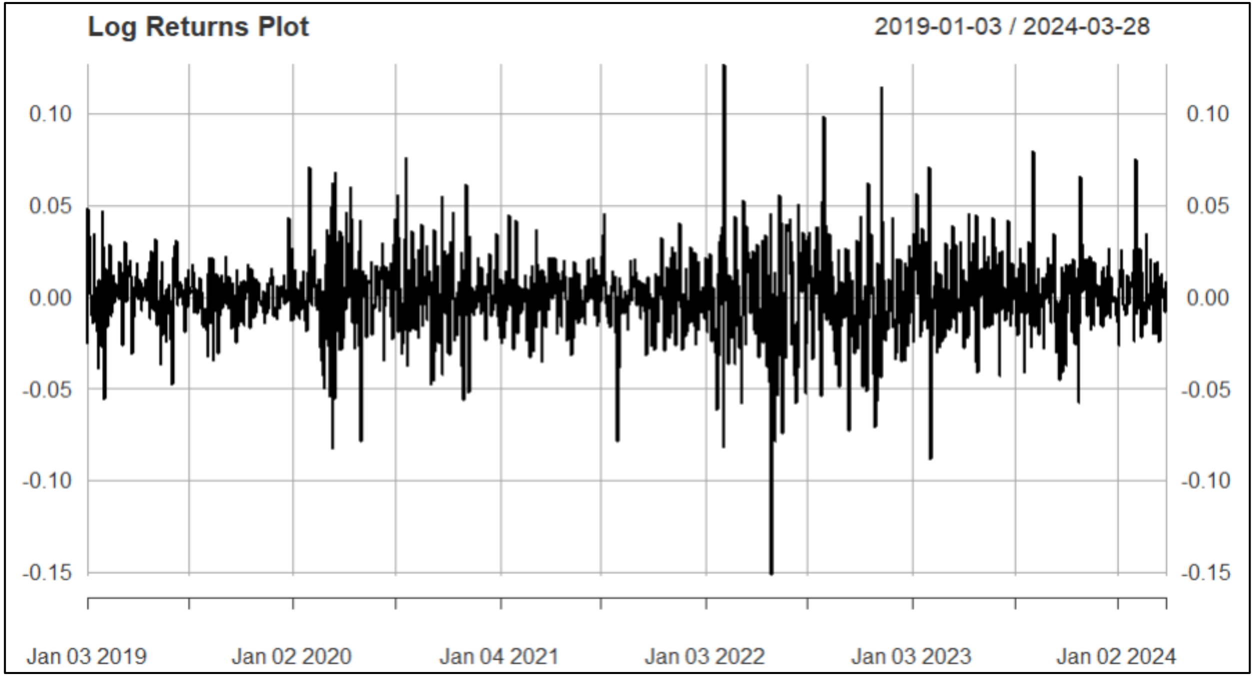
# Augmented Dickey Fuller Test

```
> print(adf.test(AMZN_CP))

        Augmented Dickey-Fuller Test

data:  AMZN_CP
Dickey-Fuller = -1.5671, Lag order = 10, p-value = 0.7616
alternative hypothesis: stationary
```

```
> print(adf.test(logs))

        Augmented Dickey-Fuller Test

data:  logs
Dickey-Fuller = -11.335, Lag order = 10, p-value = 0.01
alternative hypothesis: stationary
```

# Log Returns Plot of Amazon Closing Price

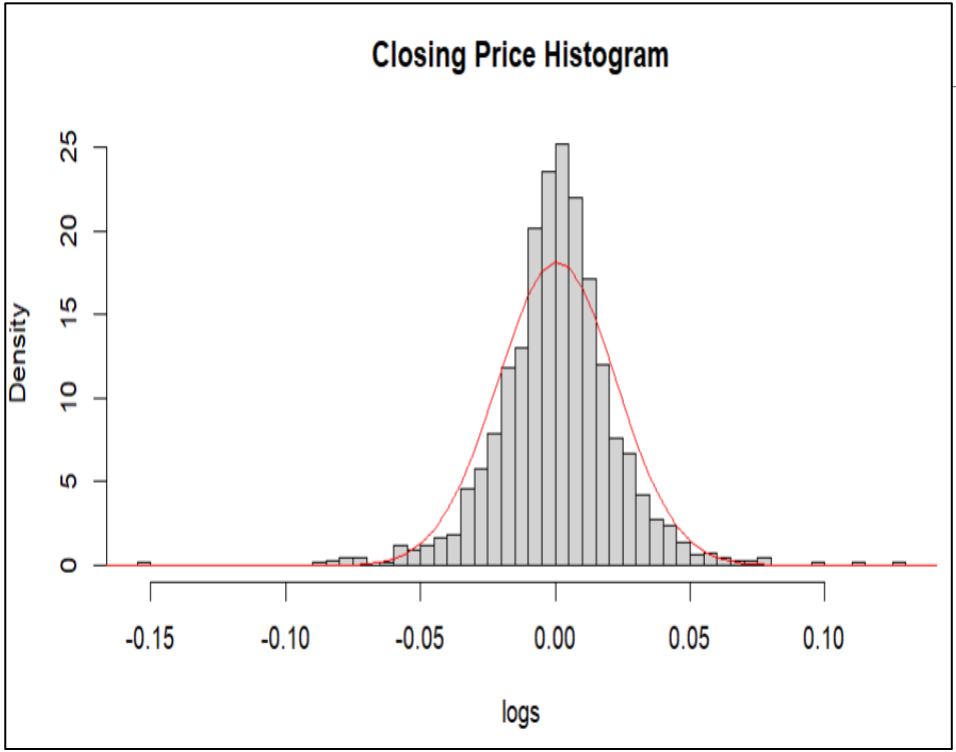
```
# Log returns plot
plot(logs, type='l', main= 'Log Returns Plot')
```

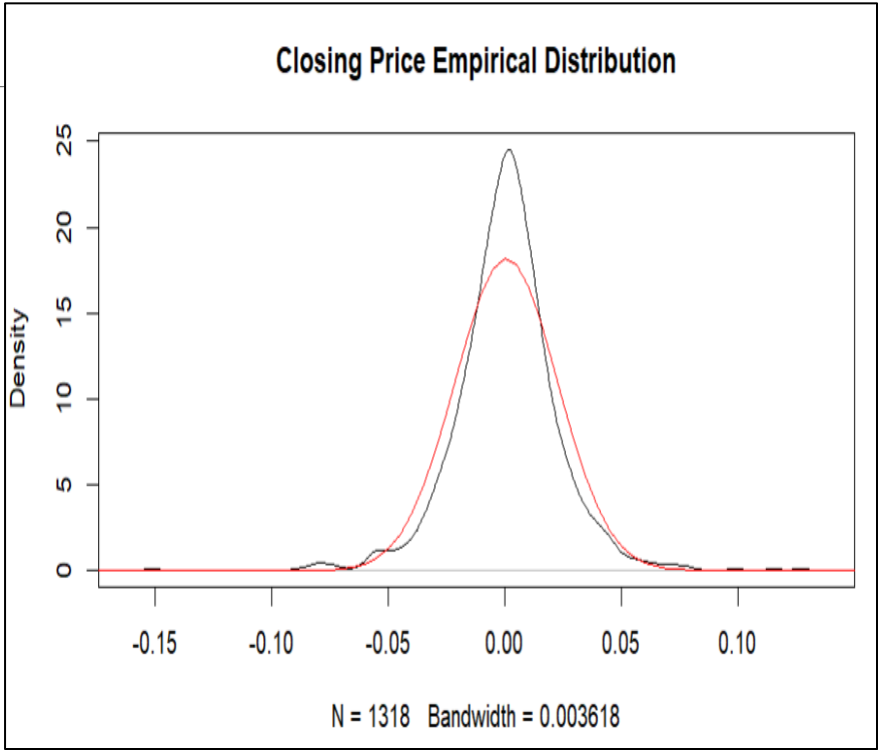


# Histogram and Empirical Distribution

```
# Histogram and Emperical Distribution
m=mean(logs);
s=sd(logs);
hist(logs, nclass=40, freq=FALSE, main='Closing Price Histogram');
curve(dnorm(x, mean=m,sd=s), from = -0.3, to = 0.2, add=TRUE, col="red")
plot(density(logs), main='Closing Price Empirical Distribution');
curve(dnorm(x, mean=m,sd=s), from = -0.3, to = 0.2, add=TRUE, col="red")
```

# Histogram and Empirical Distribution



# 1. ARIMA Modeling

- ARIMA stands for **Auto-Regressive Integrated Moving Average**.
- Used for forecasting a time series which can be made to be "**stationary**" by differencing.
- ARIMA predictor for linear equation consist of *lags of the dependent variable* and/or *lags of the forecast errors*.
- Parameters of ARIMA Model – **AR, I and MA (p, d, q**) where:
    - ❖ **p** is the number of autoregressive terms.
    - ❖ **d** is the number of non-seasonal differences needed for stationarity
    - ❖ **q** is the number of lagged forecast errors in the prediction equation

# ARIMA Fitting

```
# Fitting the ARIMA model
# Auto ARIMA with seasonal = FALSE
fit1<-auto.arima(AMZN_CP, seasonal=FALSE)
tsdisplay(residuals(fit1), lag.max = 40, main='(1,1,1) Model Residuals')
fcast1<-forecast(fit1, h=30)
plot(fcast1)
accuracy(fcast1)

# Auto ARIMA with lambda = "auto"
fit2<-auto.arima(AMZN_CP, lambda = "auto")
tsdisplay(residuals(fit2), lag.max = 40, main='(2,1,2) Model Residuals')
fcast2<-forecast(fit2, h=30)
plot(fcast2)
accuracy(fcast2)
```

# ARIMA Fitting

```
> # Auto ARIMA with seasonal = FALSE
> fit1<-auto.arima(AMZN_CP, seasonal=FALSE)
> tsdisplay(residuals(fit1), lag.max = 40, main='(1,1,1) Model Residuals')
> fcast1<-forecast(fit1, h=30)
> plot(fcast1)
> accuracy(fcast1)
                     ME     RMSE      MAE        MPE     MAPE     MASE        ACF1
Training set 0.07846889 2.824627 1.997025 0.04056302 1.564788 0.999271 -0.02146464
```

**ARIMA with seasonal = FALSE**
**MAPE = 1.564788**

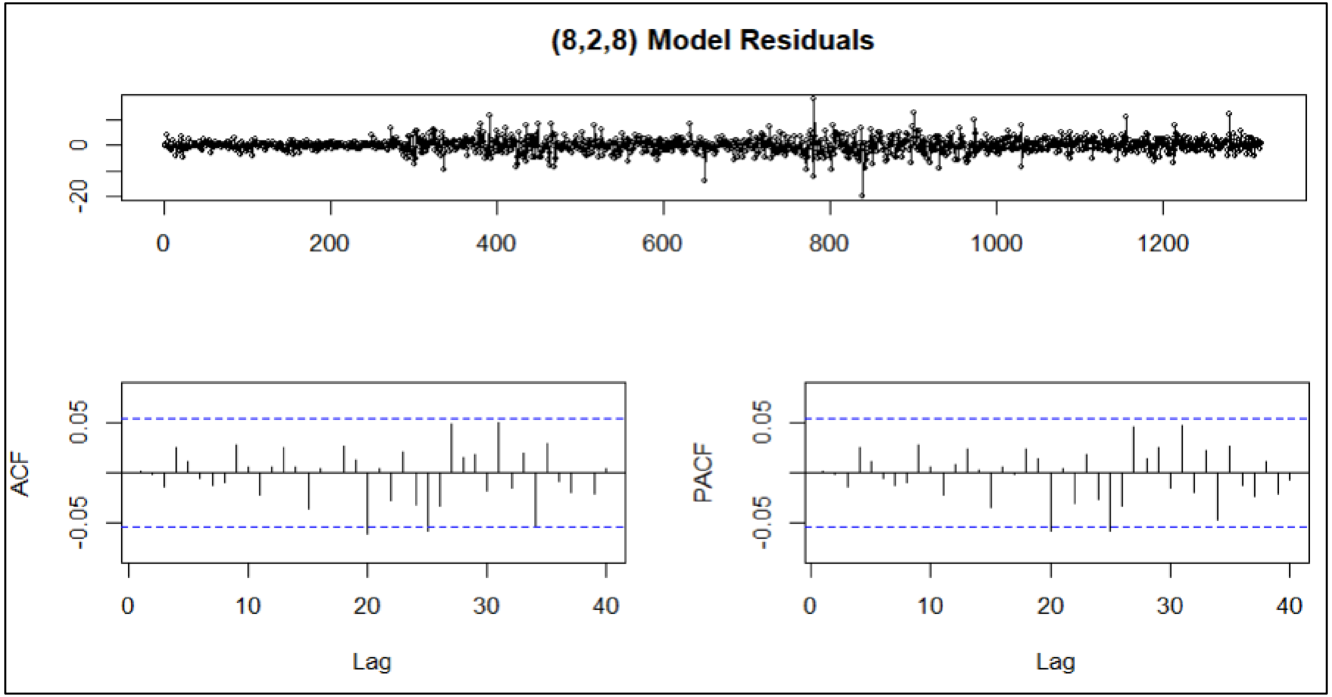
```
> # Auto ARIMA with lambda = "auto"
> fit2<-auto.arima(AMZN_CP, lambda = "auto")
> tsdisplay(residuals(fit2), lag.max = 40, main='(2,1,2) Model Residuals')
> fcast2<-forecast(fit2, h=30)
> plot(fcast2)
> accuracy(fcast2)
                     ME     RMSE      MAE        MPE     MAPE      MASE        ACF1
Training set 0.07859248 2.824634 1.997148 0.04072362 1.564948 0.9993329 -0.02149591
```

**ARIMA with lambda = auto**
**MAPE = 1.564948**

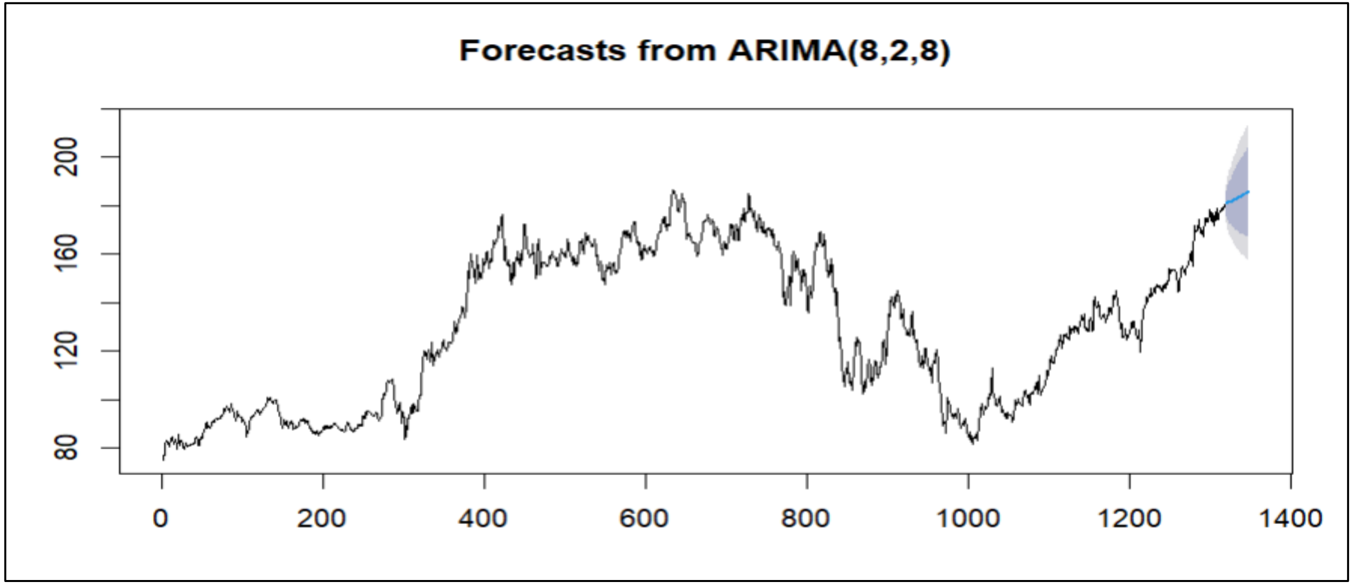
# Optimized ARIMA Model
# Order = (8, 2, 8)

```r
# ARIMA model with optimized p,d and q
fit3<-arima(AMZN_CP, order=c(8,2,8))
tsdisplay(residuals(fit3), lag.max = 40, main='(8,2,8) Model Residuals')
fcast3<-forecast(fit3, h=30)
plot(fcast3)
accuracy(fcast3)
```

# Optimized ARIMA Model
# Order = (8, 2, 8)



# Optimized ARIMA Model
# Order = (8, 2, 8)

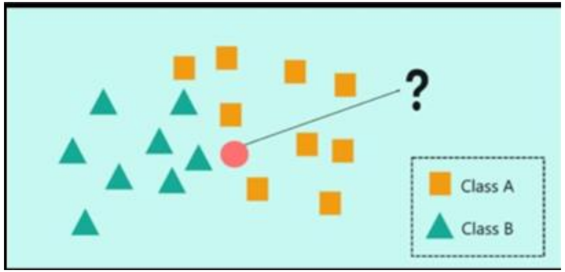



```
> accuracy(fcast3)
                      ME    RMSE      MAE         MPE     MAPE      MASE        ACF1
Training set 0.005884535 2.79802 1.973335 -0.01408278 1.548573 0.9874173 0.001869132
```

**Order = (8, 2, 8)**
**MAPE – 1.548573**

# 2. KNN Modeling

- Popular algorithm used in classification and regression problems.
- A collection of samples, each consisting a vector of features and its associated class or numeric value is stored.
- KNN finds its k most similar examples (known as nearest neighbors) according to a distance metric and predicts its class according to the majority class of neighbors.
- In regression, an aggregation of target values associated with its nearest neighbors is predicted.
- R package for KNN – tsfknn – used for univariate the time series forecasting

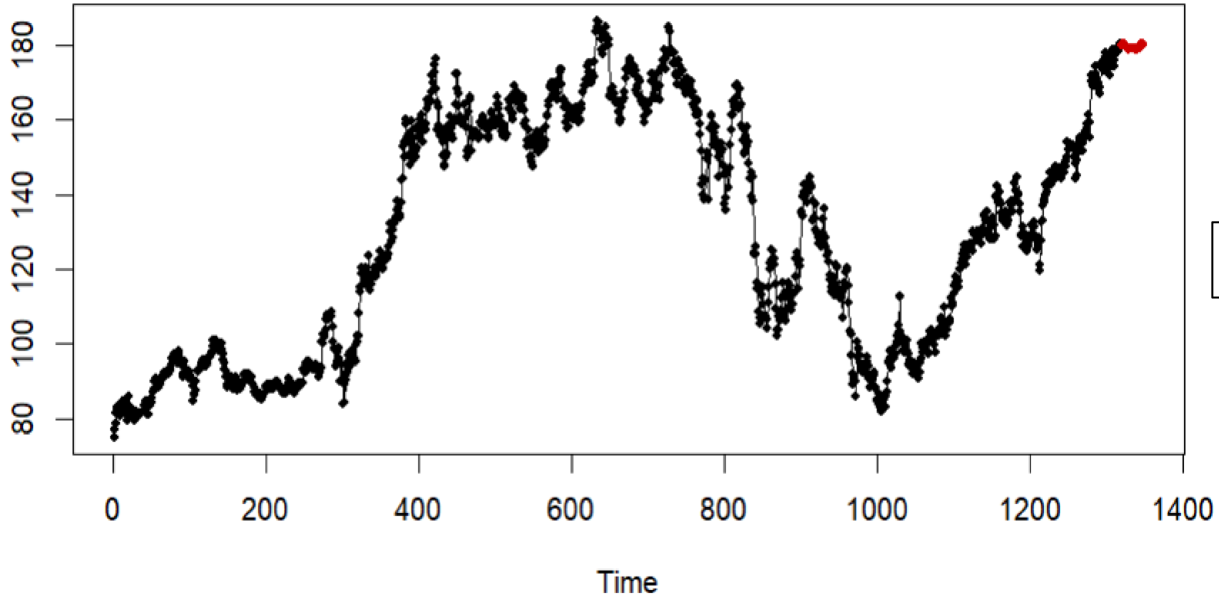



KNN Model

# 2. KNN Modeling

```r
# K Nearest Neighbours
df <- data.frame(ds = index(AMZN),
                 y = as.numeric(AMZN[,'AMZN.Close']))

predknn <- knn_forecasting(df$y, h = 30, lags = 1:30, k = 40, msas = "MIMO")
ro <- rolling_origin(predknn)
print(ro$global_accu)
plot(predknn, type="c")
```

# 2. KNN Modeling




KNN Forecast Plot

| RMSE | MAE | MAPE |
|---|---|---|
| 5.255123 | 4.335268 | 2.442462 |

**MAPE = 2.4425**

# 3. PROPHET MODELING

- Helps in shaping business decisions by following statistical approach.
- Developed by Facebook's Core Data Science team for business forecasting.
- Idea behind : By fitting the trend component very flexibly, we more accurately model seasonality.
- We use a very flexible regression model (like curve-fitting) instead traditional time series - gives us modeling flexibility, easier model fitting, gracefully handle missing data.

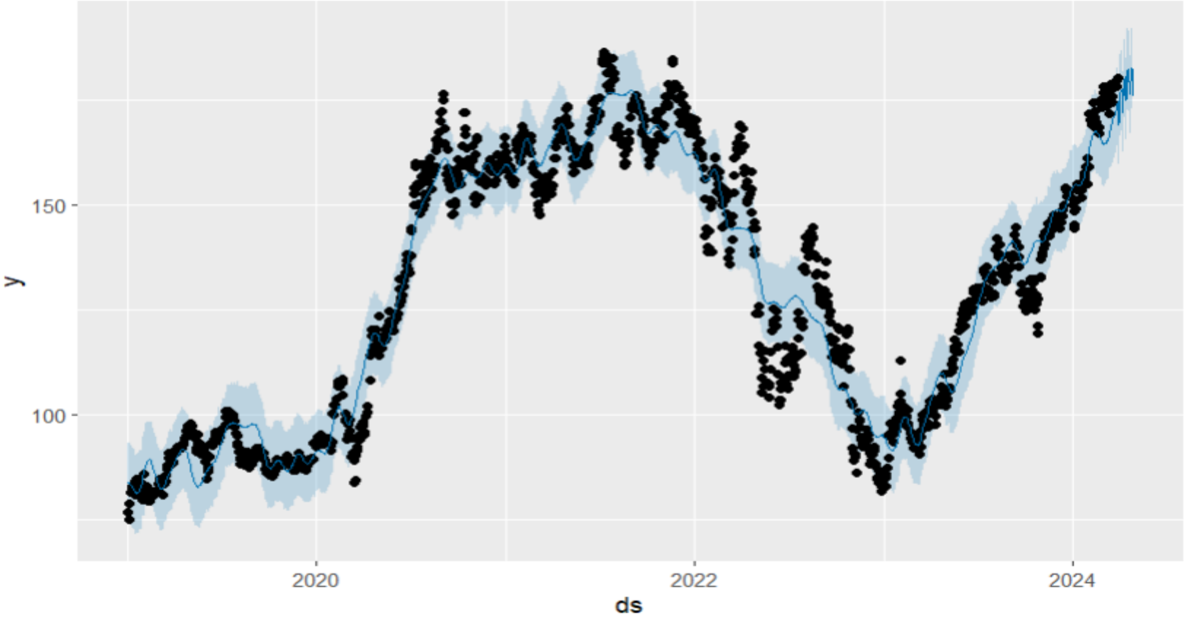
# 3. PROPHET Modeling

```
# Prophet
df <- data.frame(ds = index(AMZN),
                 y = as.numeric(AMZN[,'AMZN.Close']))
prophet_model <- prophet(df)
future <- make_future_dataframe(prophet_model, periods = 30)
forecast_prophet <- predict(prophet_model, future)

# Calculating MAPE
# Extracting the last 30 days of actual closing prices
actual <- as.numeric(AMZN_CP[(length(AMZN_CP) - 29):length(AMZN_CP)])

# Extracting the last 30 days of predicted closing prices from the Prophet forecast
predicted <- forecast_prophet$yhat[(nrow(forecast_prophet) - 29):nrow(forecast_prophet)]

mape <- mean(abs((actual - predicted) / actual)) * 100
cat("MAPE for Prophet model:", mape, "%\n")

print(mape)
```

# 3. PROPHET Modeling

```
#Plotting
plot(
  prophetpred,
  forecastprophet,
  uncertainty = TRUE,
  plot_cap = TRUE,
  xlabel = "ds",
  ylabel = "y"
)
dataprediction <- data.frame(forecastprophet$ds,forecastprophet$yhat)
trainlen <- length(AMZN_CP)
dataprediction <- dataprediction[c(1:trainlen),]
prophet_plot_components(prophetpred,forecastprophet)
```

# 3. PROPHET MODELING




**PROPHET Forecast Plot**

MAPE for Prophet model: 2.100888 %

**MAPE = 2.1009**

# RESULTS

| MODEL | MAPE | ACCURACY |
| --- | --- | --- |
| **ARIMA (Best)** | **1.5486** | **98.4514** |
| KNN | 2.4425 | 97.5575 |
| PROPHET | 2.1009 | 97.8991 |

# THANK YOU! 😇