INDEX

INTRODUCTION

A correct prediction of stocks can lead to huge profits for the seller and the broker. Frequently, it is brought out that prediction is chaotic rather than random, which means it can be predicted by carefully analyzing the history of respective stock market. Stock markets provide opportunities along with associated risks for investors to make profits. This research paper is about analyzing the NVIDIA (NVDA) stock market dataset from 02 Jan 2019 upto 18 March 2022. In this paper to analyze the NVDA data we will use two techniques Time Series Analysis and Machine Learning Linear Regression.

Statsmodels is a Python package that provides a complement to SciPy for statistical computations including descriptive statistics and estimation of statistical models. Beside the initial models, linear regression, robust linear models, generalized linear models, the statsmodels.tsa includes some basic tools and models for time series analysis. This includes statistical tests and several linear model classes: autocorrelation (ACF) , partial autocorrelation (PACF), autoregressive integrated moving average(ARIMA), seasonal decompose , simple smoothening. In this project we would like to introduce and provide an overview of the time series analysis features of statsmodels and pmdarima. In the outlook at the end we try to use simple forecast and Holt's smoothening method to complete the time series forecasting of our data.

Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something . Machine learning has significant applications in the stock price prediction. Various machine learning methods have been used for stock market prediction. The machine learning approaches are mainly categorized into supervised and unsupervised approaches. In the supervised learning approach, named input data and the desired output are given to the learning algorithms.

Linear Regression is the analysis of two separate variables to define a single relationship and it is a useful measure for technical and quantitative analysis in financial markets. Plotting stock prices along a normal distribution (bell curve) can allow traders to see when a stock is overbought or oversold. Using linear regression , a trader can identify key price points-entry price, stop-loss price and exit price. In this paper using linear regression, we try to predict the values and it compare with actual values. Along with this also try to predict future thirty days or one month values.

LITERATURE REVIEW

Kaleb Phipps , Martin Ratz , Dirk Muller , Veit Hagenmeyer and Ralf Mikut in their paper titled "***REVIEW OF AUTOMATED TIME SERIES PIPELINE***" (**Feb , 2022**) created a model using complex design which mainly included the sections data pre-processing , feature engineering , hyperparameter optimization , forecasting method selection and forecast ensembling organized in a pipeline structure . They used both Automated Machine Learning (AutoML) and automated statistical forecasting methods in a single pipeline for which they compared the automation methods for pipeline section and then analyzed automation methods regarding combination , interaction and coverage of the pipeline section. Finally they concluded that automation forecasting pipeline could be a large-scale process of time series forecasting and in future automation could be its one of the prime solution.

Julien Siebert , Janek Grob and Christof Schroth in the paper titled "***A Systematic Review Of Python Packages for Time Series Analysis***" (**June , 2021**) reviewed the packages that were used to analyse , classified the packages according to analysis tasks implemented and the means for evaluating the result. At last they their result concluded that forecasting is by far the most frequently implemented task and that half of the packages provide in gaining knowledge about data or generate synthetic data.

Wes McKinney , Josef Perktold and Skipper Seabold in the paper titled "**Time Series Analysis in Python with Statsmodels**" (**2011**) used time series analysis feature of scikits.statsmodels to show linear model classes , autoregressive , moving average , ARMA and Vector autoregressive models VAR. The process followed by setting up the ols parameters , then stationary check of unit roots , ARMA implementation and modelling and then finally using the VAR model to generate the forecast.

OBJECTIVES

A. Linear Regression

1) We will import the dataset .
2) Check is their null value is present in data.
3) Describe the data by visualization
4) Splitting the dataset for test and train dataset.
5) Showing a comparison between actual and predicted values.
6) Now creating a variable(dimension) for future prediction , i.e. for thirty days(1 month)
7) Plotting the final result – Actual, train - test dataset and Predicted values

B. Time Series

1) We will import the dataset.
2) Then followed by a stationary test on the dataset , more specifically on the closing price.
3) If not stationary , we try to make it stationary by removing trend and seasonality.
4) Now we prepare for model fitting. For stationary model we can use ARMA model but for non-stationary model we will have to use ARIMA.
5) We will use the model for forecasting using two different methods
6) Once model has been fitted we will check the performance metrics of the model.

PROJECT DESCRIPTION

A. Linear Regression

Import data →Data Preprocessing →Graphical visuals of the features in the dataset → Fitting the Linear model → Presenting the comparison between actual and predicted values → Going for thirty days prediction → Plotting the result.

B. Time Series

Graphical visuals of the features in the dataset → Stationary Check of the model →
Finding out trend and seasonality → Performing seasonal decompose →Followed by ARIMA model fiiting → Use the model for forecasting →Lastly measuring the performance of the model.

SOFTWARE REQUIREMENTS

- Device Specifications → Windows 10 , Intel ICore i7 10^{th} Gen , 64bit OS .
- Software / Tools →*Python , Statistical Software and Microsoft Power BI.*
- Libraries → *pandas , numpy , matplotlib , seaborn , scipy , statsmodels.tsa.stattools , sklearn.metrics , pmdarima, statsmodels.tsa.api , statsmodels.api , sklearn.linear_model , sklearn.model_selection , sklearn.metrics , statsmodels.stats.outliers_influence*

DATA

For our application of time series and simple linear regression we will be using the stock market dataset which has been downloaded from the yahoo website , finance.yahoo.com .
We will be using the Nvidia stock market dataset which is from 02 January , 2019 to 18 March , 2022.

Date	Open	High	Low	Close	Adj Close	Volume
Wednesday, January 2, 2019	32.66	34.619999	32.512501	34.055	33.831318	50875200
Thursday, January 3, 2019	33.447498	33.790001	31.922501	31.997499	31.787331	70555200
Friday, January 4, 2019	32.735001	34.432499	32.424999	34.047501	33.82386	58562000
Monday, January 7, 2019	34.625	36.2225	34.107498	35.849998	35.614529	70916000
Tuesday, January 8, 2019	36.672501	36.695	34.224998	34.9575	34.727886	78601600
Wednesday, January 9, 2019	35.474998	36.122501	34.965	35.645	35.410873	61726000
Thursday, January 10, 2019	35.450001	36.395	34.84	36.307499	36.069023	52315600
Friday, January 11, 2019	36.0825	37.4375	35.802502	37.2075	36.963104	87476400
Monday, January 14, 2019	36.68	37.865002	36.442501	37.610001	37.362961	73016800
Tuesday, January 15, 2019	37.939999	38.337502	37.282501	37.467499	37.221397	61701200
Wednesday, January 16, 2019	37.7425	38.075001	37.154999	37.209999	36.96558	47010400
Thursday, January 17, 2019	36.877499	38.3325	36.602501	37.93	37.680862	49343600
Friday, January 18, 2019	38.432499	39.494999	37.912498	39.232498	38.974808	65133600
Tuesday, January 22, 2019	38.927502	39.044998	36.887501	37.192501	36.948208	66155200
Wednesday, January 23, 2019	37.75	38.645	37.0075	37.322498	37.077358	59102000
Thursday, January 24, 2019	38.174999	39.637501	38.127499	39.459999	39.200821	70897200
Friday, January 25, 2019	38.860001	40.220001	37.825001	40.037498	39.774521	115457200
Monday, January 28, 2019	34.137501	35.41	32.75	34.502499	34.275879	251152800
Tuesday, January 29, 2019	34.037498	34.525002	32.752499	32.900002	32.683918	115393200
Wednesday, January 30, 2019	33.6675	34.4925	32.865002	34.3475	34.121895	97422800
Thursday, January 31, 2019	34.314999	36.297501	34.095001	35.9375	35.701454	84285200
Friday, February 1, 2019	36.125	36.697498	35.645	36.182499	35.944851	62504800
Monday, February 4, 2019	36.342499	37.669998	36.119999	37.294998	37.050041	52859200
Tuesday, February 5, 2019	37.415001	37.857498	37.075001	37.487499	37.241276	54242400
Wednesday, February 6, 2019	37.822498	38.900002	37.767502	38.25	37.998764	70246400
Thursday, February 7, 2019	37.782501	37.805	36.422501	36.855	36.61293	63712000
Friday, February 8, 2019	36.177502	37.150002	36.032501	37.0425	36.799198	46160400

NVIDIA Corp.(NVDA) engages in the design and manufacture of computer graphics processors, chipsets, and related multimedia software. It operates through the Graphics Processing Unit (GPU) and Tegra Processor segments. The GPU segment comprises of product brands, which aims specialized markets including GeForce for gamers; Quadro for designers; Tesla and DGX for AI data scientists and big data researchers; and GRID for cloud-based visual computing users. The Tegra Processor segment integrates an entire computer onto a single chip, and incorporates GPUs and multi-core CPUs to drive supercomputing for autonomous robots, drones, and cars, as well as for consoles and mobile gaming and entertainment devices. The company was founded by Jen-Hsun Huang, Chris A. Malachowsky, and Curtis R. Priem in January 1993 and is headquartered in Santa Clara, CA.

Later on NVIDIA launched its Deep Learning courses (2009) for AI enhancement.
The NVIDIA Deep Learning Institute offers resources for diverse learning needs—from learning materials to self-paced and live training to educator programs—giving individuals, teams, organizations, educators, and students what they need to advance their knowledge in AI, accelerated computing, accelerated data science, graphics and simulation, and more.

DATASET INFORMATION

DATE→The stock price of a particular stock in the market at that particular date.
HIGH→ It show the high price a stock attained for a particular period of time.
LOW→ It show the low price a stock attained for a particular period of time.
CLOSE→ It show the closing prices of the stock for the same period.
OPEN→ It show the opening prices of the stock for the same period.
VOLUME→ The total amount of a security that changes hands over a given period of time.
ADJ-CLOSE→It amends a stock's closing price to reflect that stock's value after accounting for any corporate actions and often used when examining historical returns or doing a detailed analysis of past performance.

Closing price *generally refers to the last price at which a stock trades during a regular trading session. A stock's closing price determines how a share performs during the day.The close price is considered the reference point for any time frame. It's the price traders agreed on after all the action throughout the day. When researching historical stock price data, one uses the closing price as the standard measure of the stock's value as of a specific date.*
We will be using this closing price for all of our analysis , forecasting and prediction.

METHODOLOGY

Two methods will be applied one Machine Learning Time Series Analysis and Forecasting and Statistical Linear Regression Model.

1.0 MACHINE LEARNING MODEL

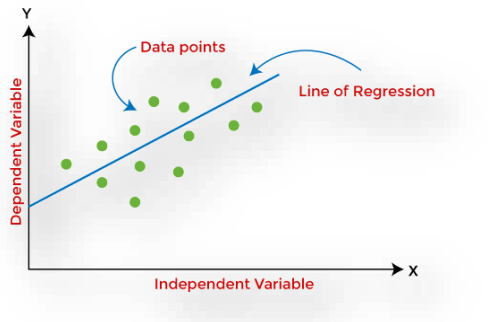
A machine learning model is defined as a mathematical representation of the output of the training process. Basically it is a file that has been trained to recognize certain types of patterns. We train a model over a set of data, providing it an algorithm that it can use to reason over and learn from those data. Once we have trained the model, we can use it to reason over data that it hasn't seen before, and make predictions about those data.

For example, let's say we want to build an application that can recognize a user's emotions based on their facial expressions. We can train a model by providing it with images of faces that are each tagged with a certain emotion, and then we can use that model in an application that can recognize any user's emotion.

1.1 SIMPLE LINEAR REGRESSION

Linear regression is the simplest machine learning model in which we try to predict one output variable using one or more input variables. The representation of linear regression is a linear equation, which combines a set of input values(x) and predicted output(y) for the set of those input values. It is represented in the form of a line:

$$Y = bx + c$$



[**Source : https://www.javatpoint.com/linear-regression-in-machine-learning**]

The main aim of the linear regression model is to find the best fit line that best fits the data points.

TRAIN AND TEST MODEL

Train and test model are two important concepts in machine learning. It is a method to measure the accuracy of model. We train the model using the training set and test the model using the testing set. Train the model means create the model and test the model means test the accuracy of the model.

MODEL PREDICTION

Prediction refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

2.0 TIME SERIES

2.1 SIMPLE DEFINITION OF TIME SERIES

A Time series is a set of observations, each one being recorded at a specific time. It decomposes the past historical data to depict the trend, seasonality, and noise to derive the future trends from it. It's a type of predictive analysis that forecasts the value of a variable in future occurrences based on history. The predicted values can be influenced by certain external factors which are known as independent variables like in the case of temperature on a particular day is influenced by the humidity or wind speed.

A discrete time series is one in which the set of time points at which observations are made is a discrete set.

Continuous time series are obtained when observations are made continuously over some time intervals.

2.2 COMPONENTS OF TIME SERIES

- **Trend** → Steady upward or downward movement with little fluctuations over a period of time.
  Eg. Increase / Decrease in profit generation .
- **Seasonal Variations**→Short term fluctuations in a time series which occur periodically within a year.
  Eg. Selling of Umbrellas in the monsoon season.
- **Cyclical Variations** →Recurrent upward / downward movement in a time series but the period of the cycle is greater than a year(not for a fixed period).
  Eg. Business Cycle
- **Irregular Variations** →Fluctuations of short term , erratic in nature , regularity (random) in occurance.
  Eg. Natural Calamities , Sudden Poverty.

DATA ANALYSIS

MACHINE LEARNING MODEL

1.2 DATA PREPROCESSING

Once the data is available, it needs some pre-processing so that it can be fed to a machine learning model. The significance of the output depends on the pre-processing of the data.

- **Importing the Libraries**

```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
plt.style.use('seaborn')
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from matplotlib.pyplot import figure
from sklearn.metrics import r2_score
```

- **Reading the Data**

	Date	Open	High	Low	Close	Adj Close	Volume
0	02-01-2019	32.660000	34.619999	32.512501	34.055000	33.831314	50875200
1	03-01-2019	33.447498	33.790001	31.922501	31.997499	31.787331	70555200
2	04-01-2019	32.735001	34.432499	32.424999	34.047501	33.823875	58562000
3	07-01-2019	34.625000	36.222500	34.107498	35.849998	35.614521	70916000
4	08-01-2019	36.672501	36.695000	34.224998	34.957500	34.727894	78601600
5	09-01-2019	35.474998	36.122501	34.965000	35.645000	35.410873	61726000
6	10-01-2019	35.450001	36.395000	34.840000	36.307499	36.069019	52315600
7	11-01-2019	36.082500	37.437500	35.802502	37.207500	36.963108	87476400
8	14-01-2019	36.680000	37.865002	36.442501	37.610001	37.362968	73016800
9	15-01-2019	37.939999	38.337502	37.282501	37.467499	37.221405	61701200
10	16-01-2019	37.742500	38.075001	37.154999	37.209999	36.965591	47010400
11	17-01-2019	36.877499	38.332500	36.602501	37.930000	37.680874	49343600
12	18-01-2019	38.432499	39.494999	37.912498	39.232498	38.974804	65133600
13	22-01-2019	38.927502	39.044998	36.887501	37.192501	36.948212	66155200
14	23-01-2019	37.750000	38.645000	37.007500	37.322498	37.077354	59102000

There are five columns. The Open column tells the price at which a stock started trading when the market opened on a particular day. The Close column refers to the price of an individual stock when the stock exchange closed the market for the day. The High column depicts the highest price at which a stock traded during a period. The Low column tells the lowest price of the period. Volume is the total amount of trading activity during a period of time.

- **Checking the dataset information**

In our data set , we observe all column values are in float data type , except the Volume which is an integer type.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 811 entries, 0 to 810
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       811 non-null    object
 1   Open       811 non-null    float64
 2   High       811 non-null    float64
 3   Low        811 non-null    float64
 4   Close      811 non-null    float64
 5   Adj Close  811 non-null    float64
 6   Volume     811 non-null    int64
dtypes: float64(5), int64(1), object(1)
memory usage: 44.5+ KB
```

Before we proceed we convert the Volume into a float data type.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 811 entries, 0 to 810
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       811 non-null    object
 1   Open       811 non-null    float64
 2   High       811 non-null    float64
 3   Low        811 non-null    float64
 4   Close      811 non-null    float64
 5   Adj Close  811 non-null    float64
 6   Volume     811 non-null    float64
dtypes: float64(6), object(1)
memory usage: 44.5+ KB
```

- **Checking whether null/NaN values are present or not**

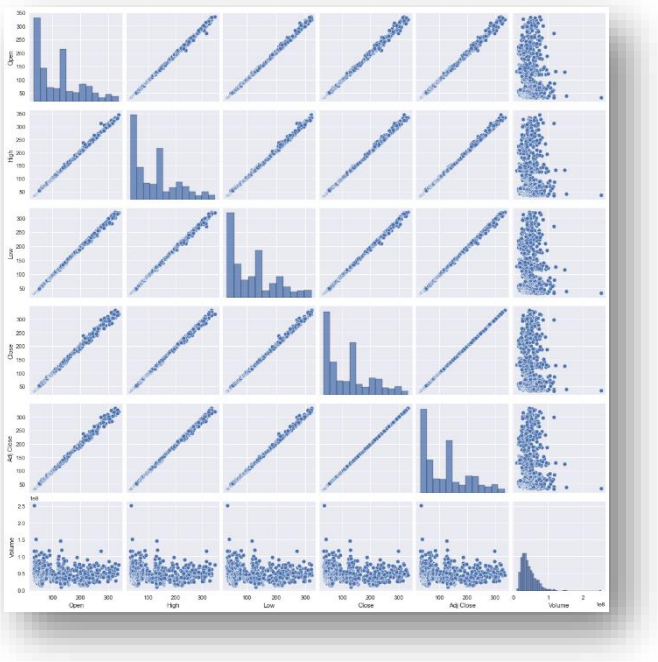
In are data set there is no null/NaN values are present

	Date	Open	High	Low	Close	Adj Close	Volume
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
806	False	False	False	False	False	False	False
807	False	False	False	False	False	False	False
808	False	False	False	False	False	False	False
809	False	False	False	False	False	False	False
810	False	False	False	False	False	False	False

811 rows × 7 columns

- **Data Visualization**

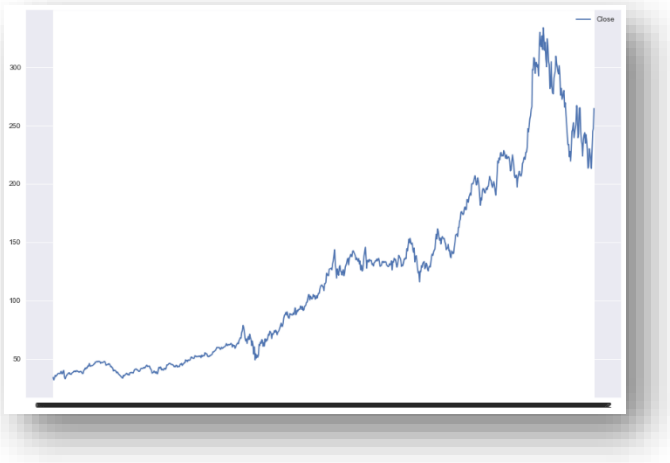
Pair plot helps to visualize the data and describes the relationship between two variables i.e. they are continuous or categorical. Pairplot is a module of seaborn library which provides a high-level interface for representing attractive and informative statistical graphics.
From the picture below, we can observe the relationship in each plot.



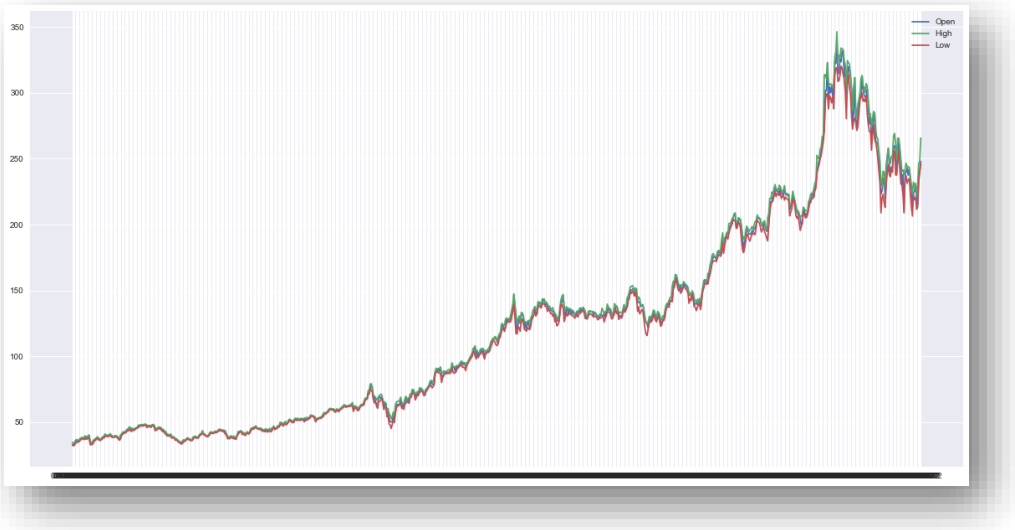
Here you can see data is continuous.

The plots are in matrix format where the row name x axis represents and column name represents the y axis. The main-diagonal subplots are the univariate histograms (distributions) for each attribute. The linear straight line between the attributes clearly represent that there is positive correlation between variables.

- This is a graphical representation of closed values based on dates.



- Trend of open, low and high price with respective of dates.

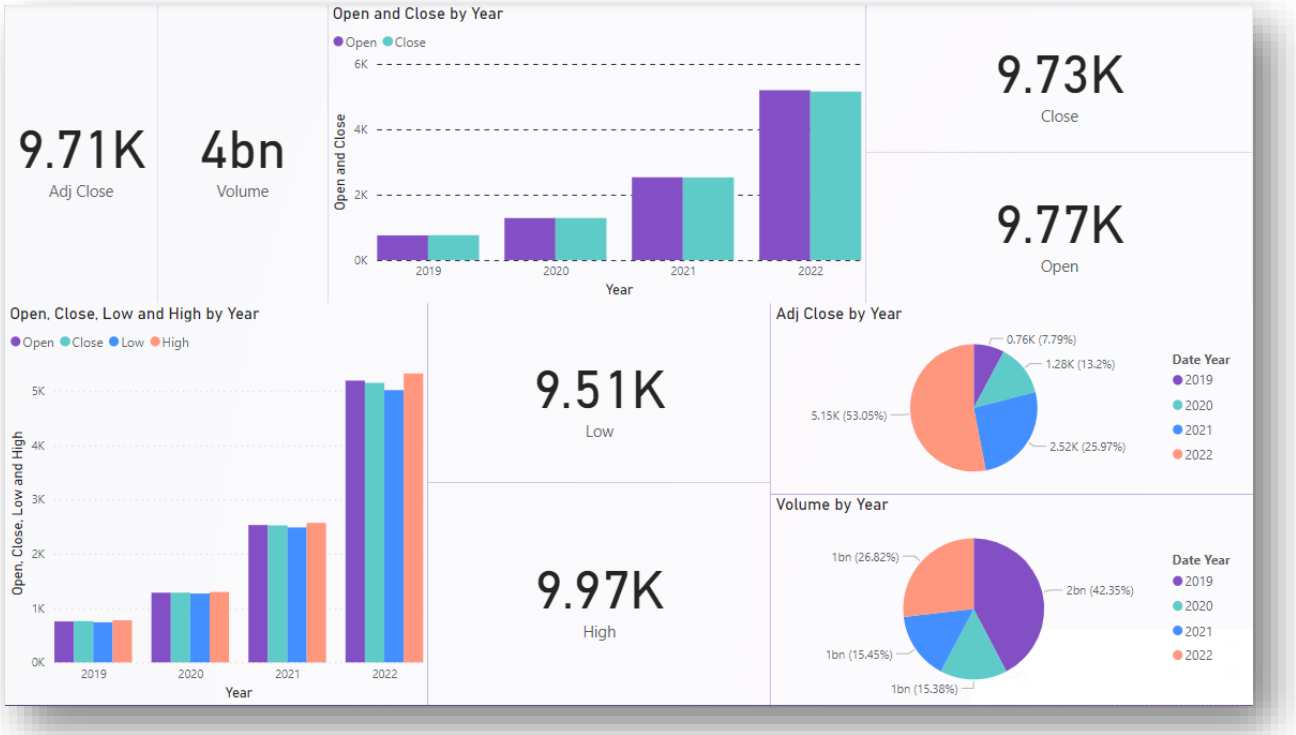


- **<u>Model Fitting</u>**

Firstly we consider ,
x=['Open','High','Low','Volume']
y=['Close']
Then using train_test_split we split the data with test size 0.2 to create linear model for prediction of close price. After that we create a data frame of actual and predicted values and compare it with each other.

TIME SERIES

2.3 TIME SERIES : GRAPHICAL VISUALIZATION



A complete Power BI dashboard showing the annual increase or decrease almost till beginning of 2022 (i.e. till March 2022).

2.4 TIME SERIES : STATIONARY MODEL

The Time series data model works on stationary data. The stationarity of data is described by the following three criteria:-

1) It should have a constant mean (the average value of all the data)
2) It should have a constant variance (difference of each point value from the mean)
3) Auto covariance(a relationship between any two values at a certain amount of time) does not depend on the time .

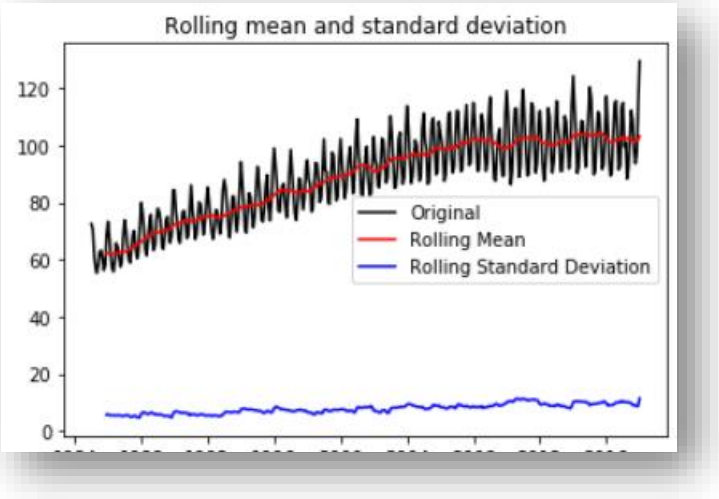
There are two methods in python to check data stationarity:-

1) Rolling statistics

This method gave a visual representation of the data to define its stationarity. A Moving variance or moving average graph is plot and then it is observed whether it varies with time or not. In this method, a moving window of time is taken (based on our needs, for eg-10, 12, etc.) and then the mean of that time period is calculated as the current value.

There are various ways in which the rolling average can be calculated, but one such way is to take a fixed subset from a complete series of numbers. The first moving average is calculated by

averaging the first fixed subset of numbers, and then the subset is changed by **moving forward** to the next fixed subset (including the future value in the subgroup while excluding the previous number from the series).

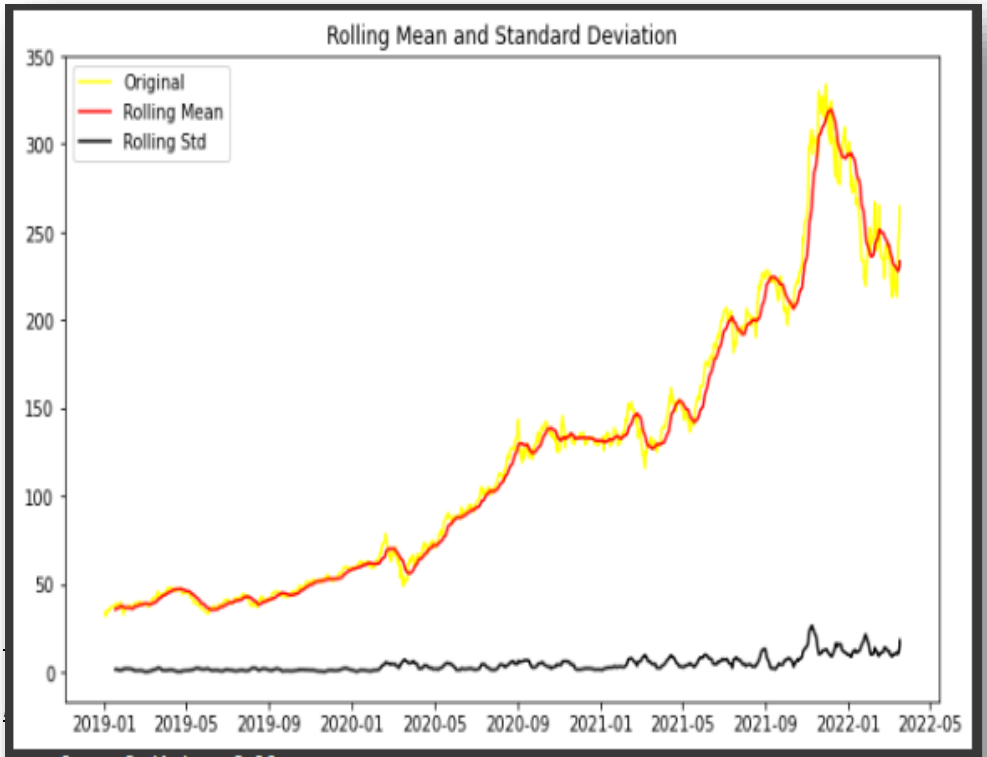


2) Augmented Dickey- fuller Test (ADCF)

In this method, we take a null hypothesis that the data is non-stationary. After executing this test, it will give some results comprised of test statistics and some other critical values that help to define the stationarity. If the test statistic is less than the critical value then we can reject the null hypothesis and say that the series is stationary.
So it help us understand if the series is stationary or not.

The null and alternate hypothesis of this test is:
(*)Null Hypothesis: The series has a unit root (value of a =1)
(*)Alternate Hypothesis: The series has no unit root.

If we fail to reject the null hypothesis, we can say that the series is nonstationary. This means that the series can be linear or difference stationary.
If both mean and standard deviation are flat lines(constant mean and constant variance), then series becomes stationary.




```
Results of dickey fuller test
Test Statistics                 -0.195654
p-value                          0.939038
No. of lags used                 6.000000
Number of observations used    804.000000
critical value (1%)             -3.438510
critical value (5%)             -2.865142
critical value (10%)            -2.568688
dtype: float64
```

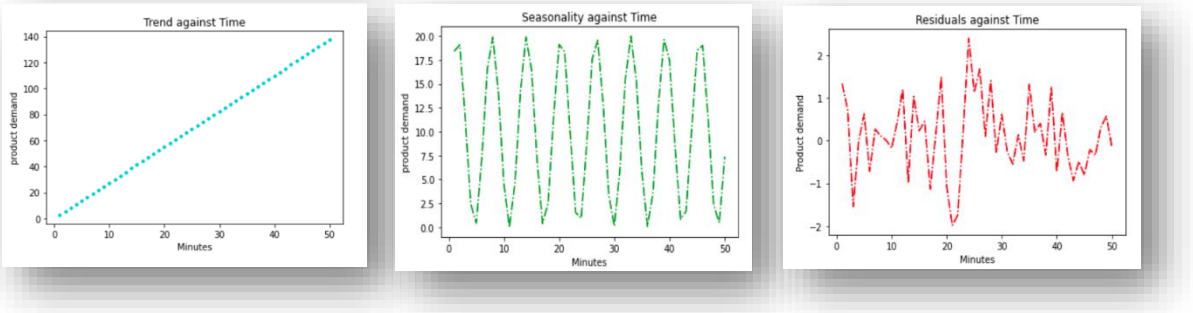
Through the above graph, we can see the increasing mean and standard deviation and hence our series is not stationary.
We see that the p-value is greater than 0.05 so we cannot reject the Null hypothesis. Also, the test statistics is greater than the critical values. so the data is non-stationary.
In order to perform a time series analysis, we may need to separate seasonality and trend from our series. The resultant series will become stationary through this process.

2.5 TIME SERIES : SEPARATION OF TREND AND SEASONALITY FROM MODEL

As we know there are few components of time series , mainly

1. **Seasonality**: Describes the periodic signal in your time series.
2. **Trend**: Describes whether the time series is decreasing, constant, or increasing over time.

3. **Noise / Residual** : Describes what remains behind the separation of seasonality and trend from the time series.



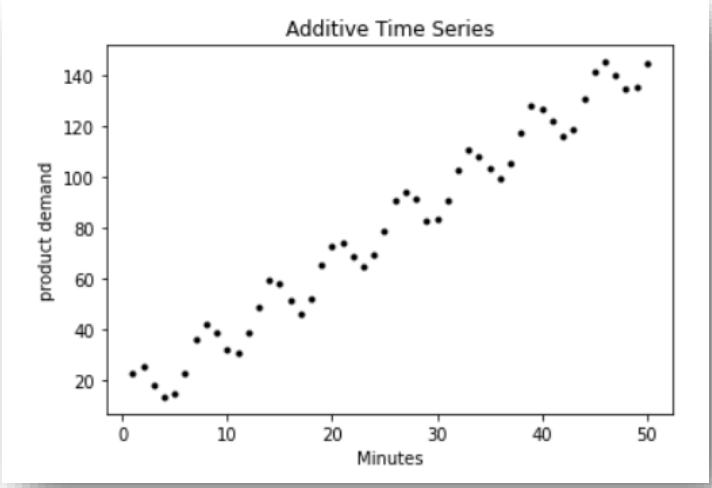
[Source : https://www.section.io/engineering-education]

Time series decomposition is a technique that splits a time series into several components, each representing an underlying pattern category, trend, seasonality, and noise. Decomposition provides a useful abstract model for thinking about time series generally and for better understanding problems during time series analysis and forecasting.

An additive model suggests that the components are added together as follows :

Level + Trend + Seasonality + Noise

An additive model is linear where changes over time are consistently made by the same amount.

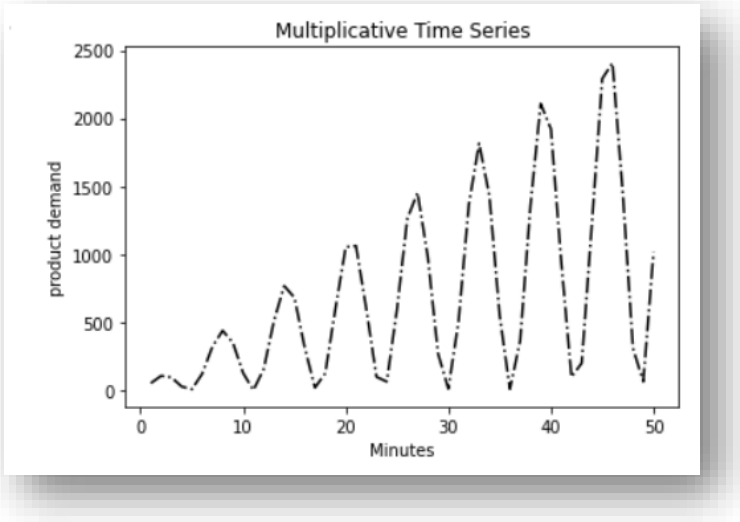


[**Source : https://www.section.io/engineering-education**]

A multiplicative model suggests that the components are multiplied together as follows:

Level * Trend * Seasonality * Noise

A multiplicative model is nonlinear, such as quadratic or exponential. Changes increase or decrease over time.



[**Source : https://www.section.io/engineering-education**]

2.6 TIME SERIES : SEASONAL DECOMPOSE

Time series decomposition is about breaking up a time series into components, most notably: a trend component, a seasonal component, and a residual component. There are many methods to decompose a time series with a single seasonal component implemented in Python.

But why would we decompose a time series?
Time series decomposition is used in time series analysis for tasks such as:

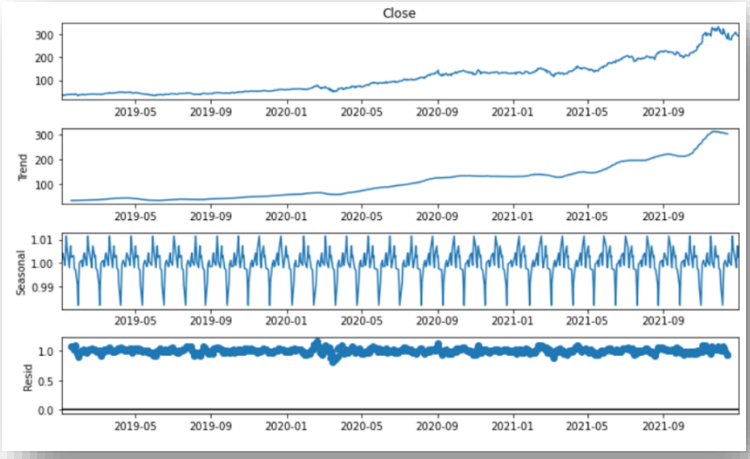
- exploratory data analysis (e.g., has unemployment increased this quarter after adjusting for seasonality?);
- pre-processing time series to identify and impute outliers and missing data;
- extracting features from time series for later use in classification, regression, and forecasting tasks;
- building forecasts (e.g., the components can be forecasted separately and then aggregated to produce the final forecast).

Some commonly used methods for time series decomposition include:

- A naive method --- where we apply a moving average to extract a trend and take averages of a seasonal index (e.g., month) to extract seasonality
- Seasonal-Trend decomposition .
- X-13-ARIMA-SEATS .

The below graphs shows the seasonal decompose where actual , trend , seasonality and residual has been decomposed.
The first graph shows the plot with additive trend while the second one shows the multiplicative trend.

The graphs shows **additive trend** where frequency or period equals twelve.



The graphs shows **multiplicative trend** where frequency or period equals twelve.

From the two graphs we can conclude that there is **additive effect** on the dataset. So for our future reference model we will be using additive model.

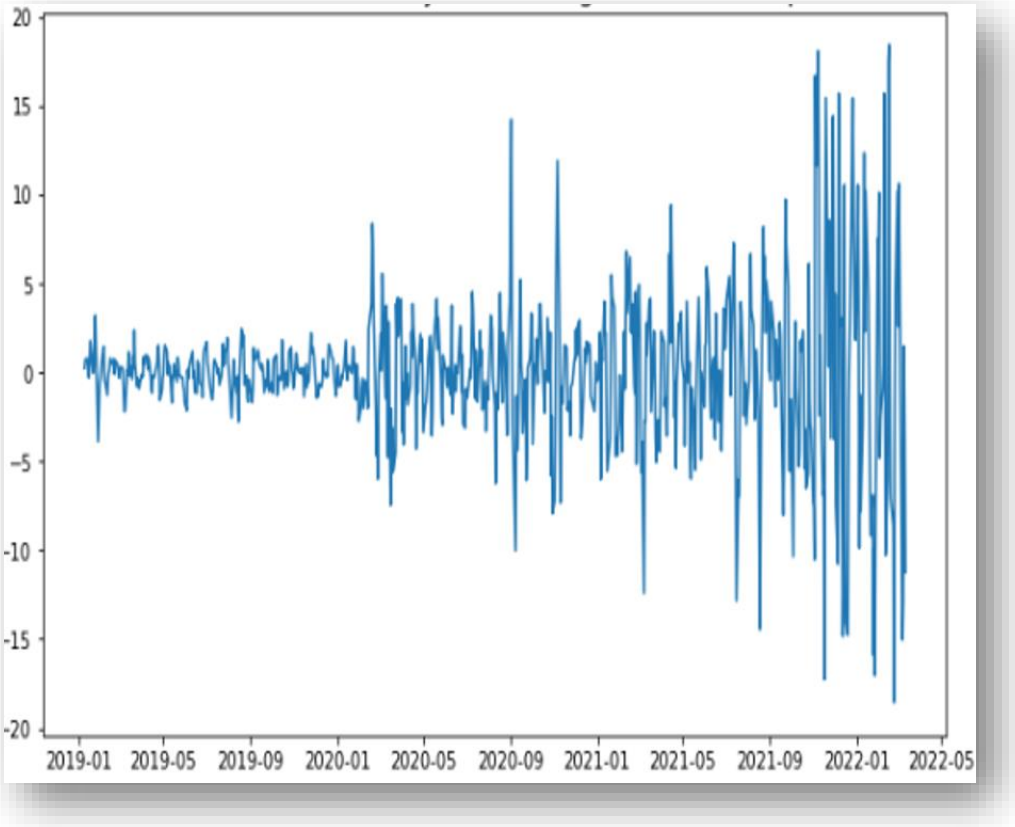
2.7 TIME SERIES : DETREND AND DESEASONALIZE

To make a model stationary we have to perform two main operations :

1) Detrend time series model – removing trend component.
2) Deseasonalize time series model – removing varying seasonal effect.

Detrending a time series means to remove the trend component from the time series.
There are multiple approaches of doing this as listed below:

1) We subtract the line of best fit from the time series.
2) For more complex trends, we may want to use quadratic terms (x^2) in the model.
3) We subtract the trend component obtained from time series decomposition.
4) We subtract the mean.
5) Apply a filter like Baxter-King filteror the Hodrick-Prescott Filter to remove the moving average trend lines or the cyclical components.

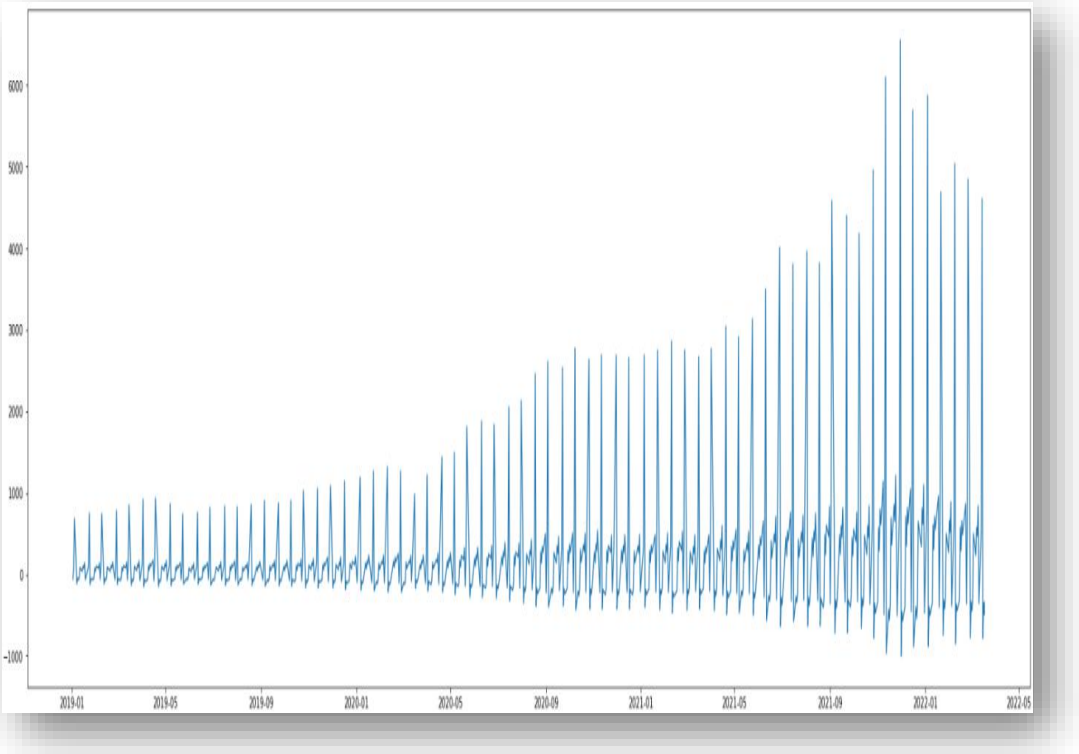


In many cases, seasonal patterns are removed from time-series data when they are released on public databases. Data that has been stripped of its seasonal patterns is referred to as seasonally adjusted or **Deseasonalized** data.

There are multiple approaches to **deseasonalize** a time series.

These approaches are listed below:

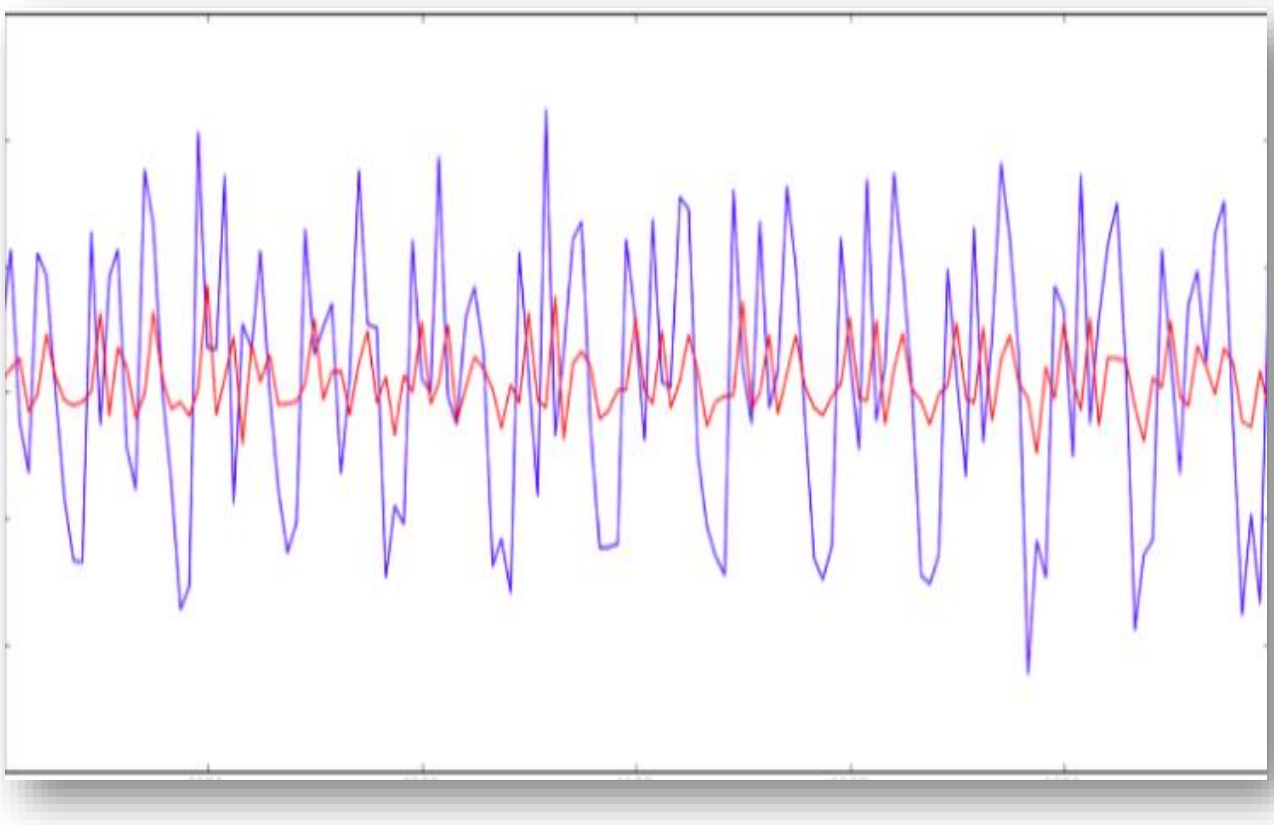
1) We take a moving average with length as the seasonal window. This will smoothen in series in the process.
2) Seasonal difference the series (i.e. to subtract the value of previous season from the current value).
3) Then to divide the series by the seasonal index obtained from decomposition.
4) If dividing by the seasonal index does not work well, we will take a log of the series and then do the deseasonalizing.
5) We can later restore to the original scale by taking an exponential.



2.8 TIME SERIES : AR , MA , ARMA AND ARIMA MODELS

- **A. Autoregressive Model: AR(p)**

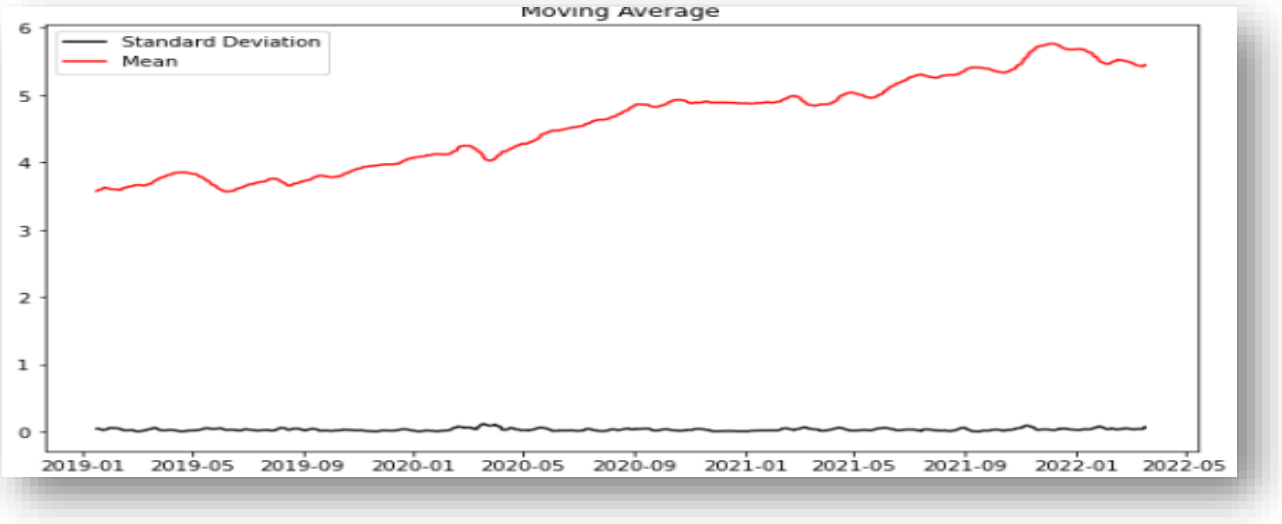
The autoregressive model uses observations from preivous time steps as input to a regression equations to predict the value at the next step. The AR model takes in one argument, p, which determines how many previous time steps will be inputted.
The order, p, of the autoregressive model can be deterimined by looking at the partial autocorrelation function (PACF). The PACF gives the partial correlation of a stationary time series with its own lagged values, regressed of the time series at all shorter lags.



[**Source : https://medium.com/@stallonejacob**]

- **B. Moving Average Model: MA(q)**

A moving average, also called a rolling or running average, is used to analyze the time-series data by calculating averages of different subsets of the complete dataset. Since it involves taking the average of the dataset over time, it is also called a moving mean (MM) or rolling mean.
The moving average model is a time series model that accounts for very short-run autocorrelation. It basically states that the next observation is the mean of every past observation. The order of the moving average model, q, can usually be estimated by looking at the ACF plot of the time series.



We start by taking a log of the series to reduce the magnitude of the values and reduce the rising trend in the series. Then after getting the log of the series, we find the rolling average of the series. A rolling average is calculated by taking input for the past 12 months and giving a mean consumption value at every point further ahead in series.

- **C. Autoregressive Moving Average Model: ARMA(p,q)**

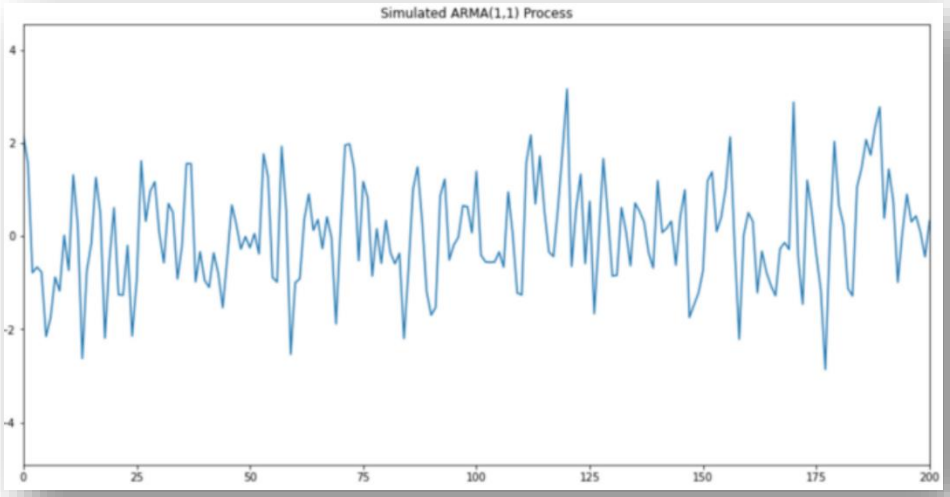
Autoregressive moving average models are simply a combination of an AR model and an MA model.

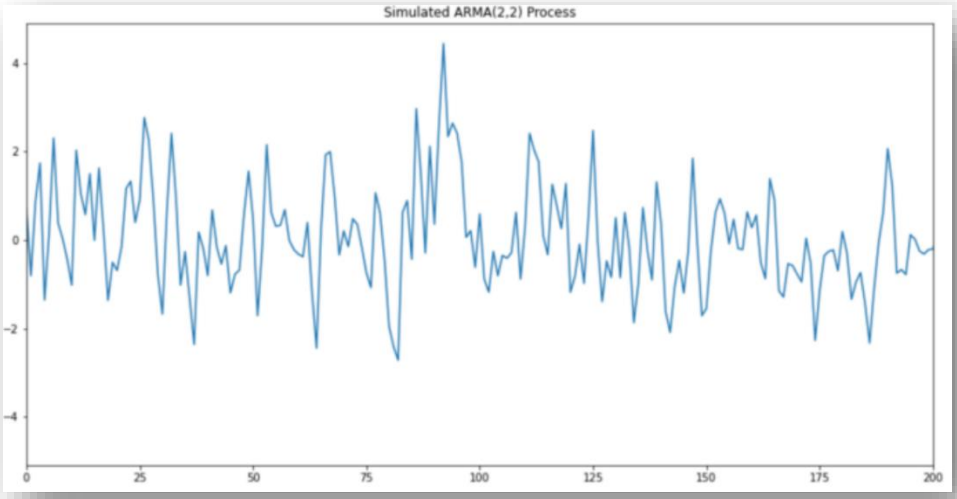
The process $\{X_t , t \in T\}$ is called an ARMA (p,q) process if $\{X_t\}$ is stationary and if for every t.

$$X_t - \emptyset_1 X_{t-1} - \emptyset_2 X_{t-2} - \cdots - \emptyset_P X_{t-P} = Z_t - Z_1\theta_{t-1} - Z_2\theta_{t-2} - \cdots - Z_P\theta_{t-P}$$

Where $Z_t \sim WN\ (0 , \sigma^2)$.

ARMA is basically a combination of both AR and MA.





[**Source** : **https://towardsdatascience.com/advanced-time-series-analysis-with-arma-and-arima-a7d9b589ed6d**]

The following above graph shows specific ARMA models when p=1 , q=1 , i.e. ARMA(1,1) process and when p=2 , q=2 , i.e. ARMA(2,2) process.

- **D. Autoregressive Integrated Moving Average Model: ARIMA(p,d,q)**

ARIMA is an acronym that stands for Auto-Regressive Integrated Moving Average. It is a class of model that captures a suite of different standard temporal structures in time series data.

An ARIMA model is a class of statistical models for analyzing and forecasting time series data. It is really simplified in terms of using it, yet this model is really powerful.

ARIMA used only when the data is non-stationary as in our case while ARMA follows the stationarity rule.

The parameters of the ARIMA model are defined as follows:

- p: The number of lag observations included in the model, also called the lag order.
- d: The number of times that the raw observations are differenced, also called the degree of difference.
- q: The size of the moving average window, also called the order of moving average.

The forecasting equation is constructed as follows. First, let y denote the d^{th} difference of Y, which means:
If d=0: $y_t = \boldsymbol{Y_t}$.
If d=1: $y_t = \boldsymbol{Y_t - Y_{t-1}}$.
If d=2: $y_t = \boldsymbol{Y_t - Y_{t-1} - Y_{t-1} - Y_{t-2} = Y_t - 2Y_{t-1} + Y_{t-2}}$.

For d=0 , ARIMA model gets converted into ARMA model , i.e. from non-stationary to stationary model.

In terms of y, the general forecasting equation is:
$\widehat{y_t} = \mu + \varphi_1 y_{t-1} + \cdots + \varphi_p y_{t-p} - \theta_1 e_{t-1} - \cdots - \theta_q e_{t-q}$.

Now we are going to create an ARIMA model and will train it with the closing price of the stock on the train data. So let us split the data into training and test set and visualize it.

Its time to choose parameters p,q,d for ARIMA model. Now we are going to use Auto ARIMA to get the best parameters .

```
                            ARIMA Model Results
==============================================================================
Dep. Variable:                D.Close   No. Observations:                  725
Model:                 ARIMA(2, 1, 0)   Log Likelihood                1526.237
Method:                       css-mle   S.D. of innovations              0.029
Date:                Sat, 16 Apr 2022   AIC                          -3044.474
Time:                        13:56:46   BIC                          -3026.130
Sample:                             1   HQIC                         -3037.395

==================================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
const              0.0031      0.001      3.043      0.002       0.001       0.005
ar.L1.D.Close     -0.1546      0.037     -4.165      0.000      -0.227      -0.082
ar.L2.D.Close      0.0688      0.037      1.847      0.065      -0.004       0.142
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -2.8506           +0.0000j            2.8506            0.5000
AR.2            5.0960           +0.0000j            5.0960            0.0000
-----------------------------------------------------------------------------
```

Auto ARIMA: Automatically discover the optimal order for an ARIMA model.
The auto_arima function seeks to identify the most optimal parameters for an ARIMA model, and return a fitted ARIMA model.
The auro_arima function works by conducting differencing tests (i.e., Kwiatkowski–Phillips–Schmidt–Shin, Augmented Dickey-Fuller or Phillips–Perron) to determine the order of differencing, d, and then fitting models within ranges of defined start_p, max_p, start_q, max_q ranges. If the seasonal optional is enabled, auto_arima also seeks to identify the optimal P and Q hyper- parameters after conducting the Canova-Hansen to determine the optimal order of seasonal differencing, D.

```
Performing stepwise search to minimize aic
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=-3024.777, Time=0.18 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=-3043.070, Time=0.11 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=-3040.385, Time=0.29 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=-3019.250, Time=0.09 sec
 ARIMA(2,1,0)(0,0,0)[0] intercept   : AIC=-3044.474, Time=0.16 sec
 ARIMA(3,1,0)(0,0,0)[0] intercept   : AIC=-3042.734, Time=0.96 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=-3042.409, Time=0.31 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=-3044.058, Time=0.61 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=-3040.803, Time=0.40 sec
 ARIMA(2,1,0)(0,0,0)[0]             : AIC=-3037.488, Time=0.17 sec

Best model:  ARIMA(2,1,0)(0,0,0)[0] intercept
Total fit time: 3.302 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                           y   No. Observations:                  726
Model:                 SARIMAX(2, 1, 0)   Log Likelihood                1526.237
Date:                  Fri, 25 Mar 2022   AIC                          -3044.474
Time:                          11:13:26   BIC                          -3026.129
Sample:                               0   HQIC                         -3037.394
                                  - 726
Covariance Type:                    opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0033      0.001      2.981      0.003       0.001       0.006
ar.L1         -0.1540      0.026     -5.849      0.000      -0.206      -0.102
ar.L2          0.0689      0.026      2.656      0.008       0.018       0.120
sigma2         0.0009   2.51e-05     34.554      0.000       0.001       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):               780.58
Prob(Q):                              0.98   Prob(JB):                         0.00
Heteroskedasticity (H):               1.00   Skew:                            -0.48
Prob(H) (two-sided):                  1.00   Kurtosis:                         7.99
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

RESULTS AND INTERPRETATIONS

1.3 LINEAR REGRESSION PREDICTION

- **Prediction**

Here we try to predict the close value .

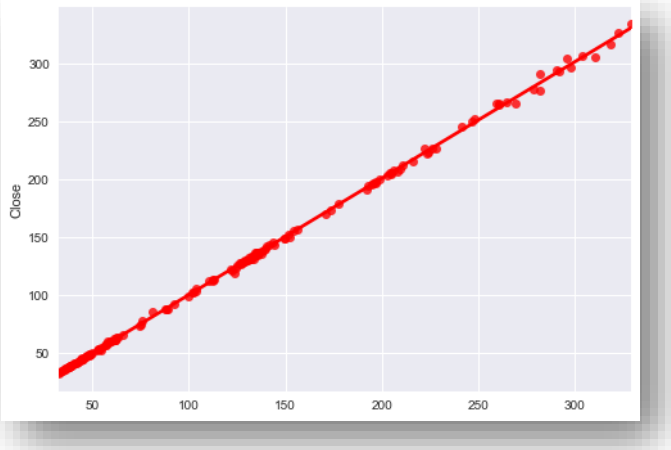
```
Date
2020-01-09    138.210007
2021-03-19    128.457504
2022-01-28    228.399994
2021-07-07    203.717499
2021-02-24    144.990005
                 ...
2022-11-01    278.170013
2019-07-10     46.082500
2021-01-07    202.119995
2021-03-23    130.707504
2021-09-20    211.130005
Name: Close, Length: 608, dtype: float64
```

We observe that there is not much difference in actual and predicted values.

We check the accuracy of the prediction by using R^2 score.

r2_score(predicted,y_test)=0.9994739616773841

This is also we show that by using graph of Comparison of Actual Value v/s Predicted Values.



The graph clearly represents that there is not a lot of dispersion between actual value and the predicted value. It signifies good accuracy of the predicted values and we can move further for the forecasting method.

- **Future One month or Thirty Days Prediction**

From the past values of close price we try to predict the future thirty days or one month values of Close price.

	Predicted_Outcome1
0	297.564676
1	310.170213
2	304.152395
3	301.051735
4	297.043059
5	295.352664
6	302.210845
7	294.183885
8	277.898147
9	283.442635
10	274.449733
11	275.927621
12	279.955604
13	281.713593
14	267.958614

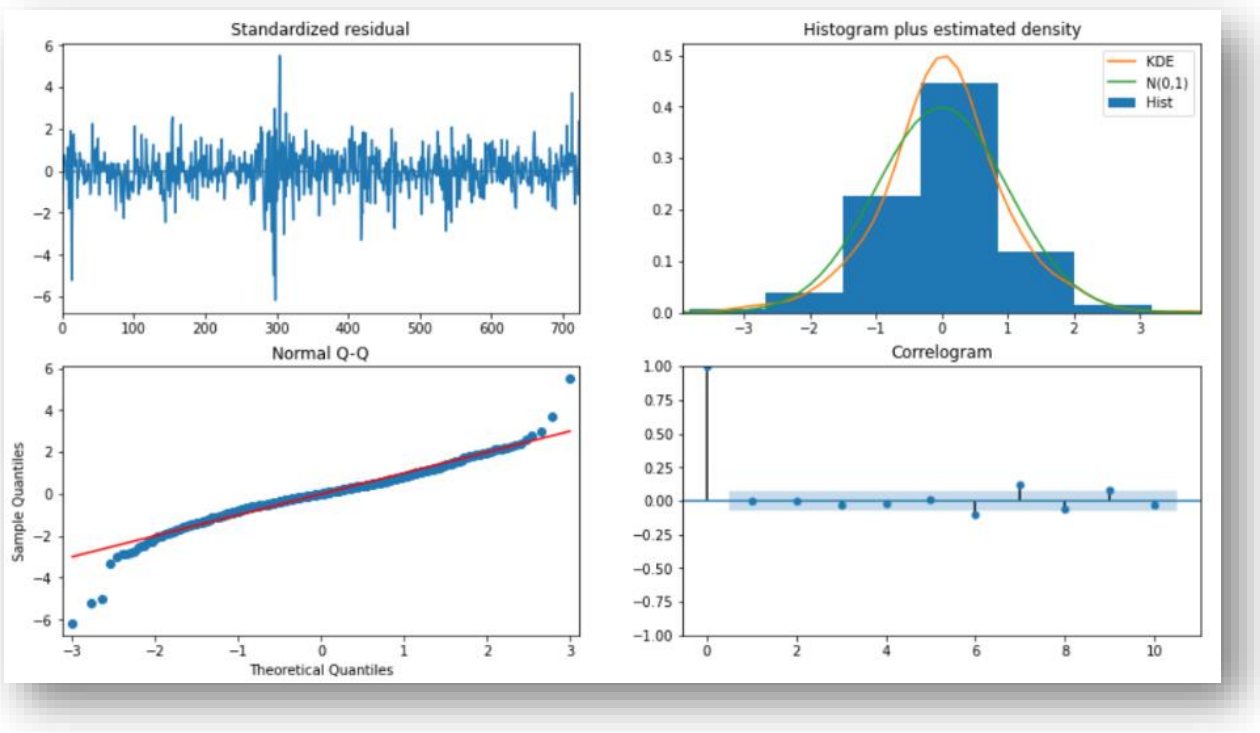
```
[297.56467567 310.17021282 304.15239468 301.05173529 297.04305927
 295.3526639  302.21084519 294.18388493 277.89814745 283.44263461
 274.44973317 275.92762073 279.95560397 281.71359293 267.95861398
 271.50362711 261.46749479 253.39223365 244.53456384 237.03887263
 237.01954996 226.89650041 231.22390869 223.2259247  231.88074127
 247.78012392 249.24835691 255.08262902 242.58336075 246.16700473]
```

2.9 TIME SERIES MODELLING AND FORECASTING

This is section is divided into three parts :-

1) Auto ARIMA plot diagnostics.
2) Forecast Measures.
3) Performance Metrics.

2.9.A AUTO ARIMA PLOT DIAGNOSTICS



Top left: The residual errors seem to fluctuate around a mean of zero and have a uniform variance.

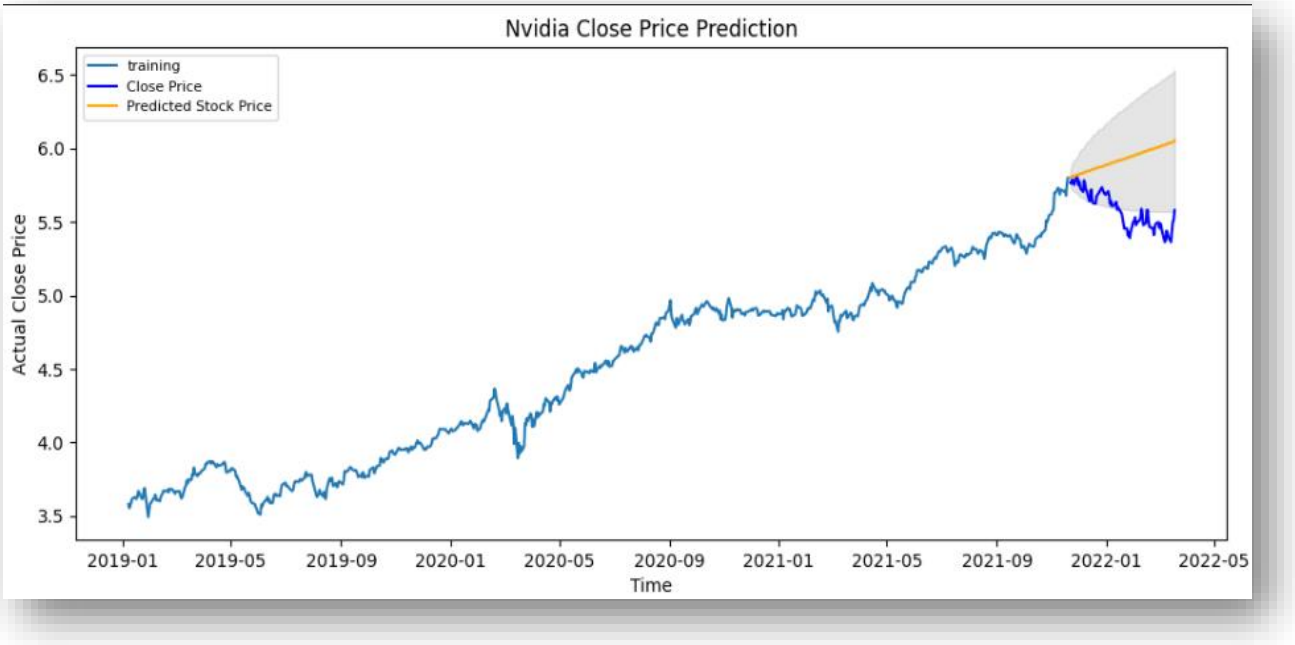
Top Right: The density plot suggest normal distribution with mean zero.

Bottom left: All the dots should fall perfectly in line with the red line. Any significant deviations would imply the distribution is skewed.

Bottom Right: The Correlogram, ACF plot shows the residual errors are not autocorrelated. Any autocorrelation would imply that there is some pattern in the residual errors which are not explained in the model.

2.9.B FORECAST MEASURES

1) Simple Forecast

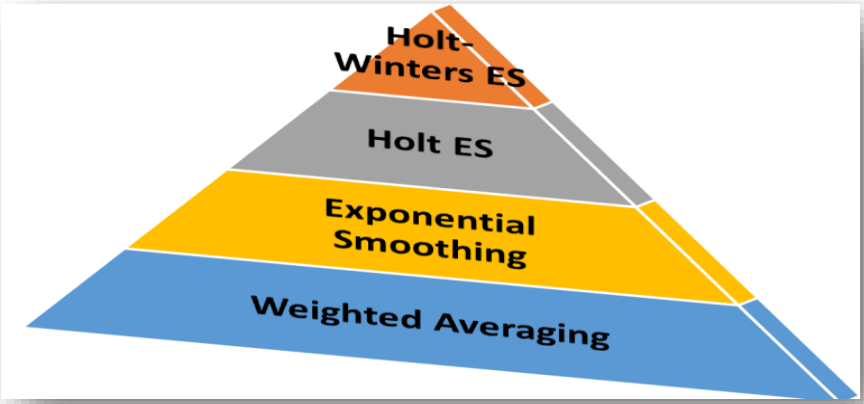


This a simple forecasting model using the auto_arima model shown above.
The orange line under the shaded region gives us a glimpse of what the price might look in the future and here it suggests that closing price will show a increasing upward trend for the coming periods.

2) Holt Winters Exponential Smoothening

A weighted average is simply an average of n numbers where each number is given a certain weight and the denominator is the sum of those n weights.
Simple Exponential Smoothing as the name suggest is used for forecasting when the data set has no trend or seasonality .
Holt Exponential Smoothing can be used to forecast time series data that has trend but fails in the presence of seasonal variations in time series data.
The **Holt-Winters Exponential Smoothening** modifies the **Holt Exponential Smoothening** technique so that it can be used in the presence of ***both trend*** and ***seasonality*.**

Holt Winter's time series model is a very powerful prediction algorithm despite being one of the simplest models. It is used in various business problems mainly because of two reasons one of which is the simplest **implementation** and the model will **evolve** as business requirement change.

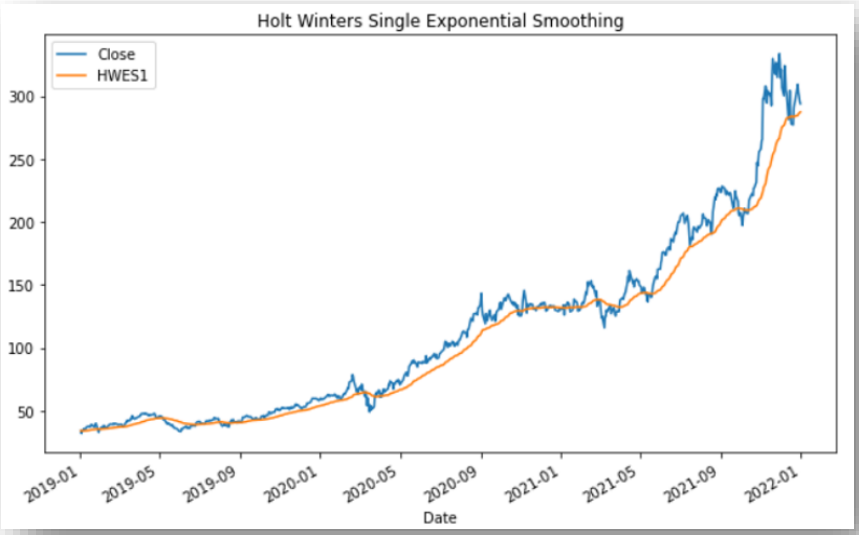


[**Source : https://timeseriesreasoning.com/contents/holt-winters-exponential-smoothing/**]

Types of Exponential Smoothening (ES) :-

1) *Single ES – Simple ES*
2) *Double ES – Holt ES (additive and multiplicative)*
3) *Triple ES – Seasonality + Trend , i.e. HWES*

A. Simple Exponential Smoothening / Single Exponential Smoothening



Simple or Single Exponential Smoothening as the name suggest s is used for forecasting when the dataset has no trend and seasonality , i.e. free of both trend and seasonal components.

B. Holt Exponential Smoothening / Double Exponential Smoothening

Holt / Double Exponential Smoothening is a technique that overcomes one of the two above defined shortcomings. It can be used to forecast the data when trend component is present in the data but fails in the presence of seasonal component.

Here the **trend** is taken as **additive** component.

C. Holt Winter Exponential Smoothening / Triple Exponential Smoothening

Holt Winter / Triple Exponential Smoothening modifies the Holt Exponential Smoothening technique so that it can be used to forecast when both trend and seasonality is present in the data.

Here the parameters of **trend** and **seasonality** has been taken as **additive**.

Although the graph may look similar to that of Double Exponential Smoothening but difference do exist (towards the end of 2022) which might not be visible unless focused on a particular month of year .

D. Holt Winter Forecasting

As we already know that there are two variations in the seasonal component and they can be defined as the following under smoothening effect :-

- **Additive Model** :- An additive model is linear where changes over time are consistently made by the same amount. It can be shown as the form **Level+Trend+Seasonality+Noise.**

- **Multiplicative Model** :- An multiplicative model is non-linear such as quadratic or exponential where changes increases or decreases over time . It can be shown as the form **Level*Trend*Seasonality*Noise .**

Now we will derive the forecasting equation from the following equations :-

Trend Estimation

$B = \beta * (L) + (1-\beta) * \beta_1$ ----------(I)

where (L)=estimated rate of change and β_1=estimated trend.

Level Estimation

$\alpha = \alpha * \frac{T}{S} + (1-\alpha) * (L+B)$ ----------(II)

where T=Time Series value , S=Seasonal Component and (L+B)=estimated level.

Seasonal Component Estimation

$S = \varphi * \frac{T}{L} + (1- \varphi) * S$ ----------(III)

where T=Time Series value , L=Level and S=Seasonal Component.

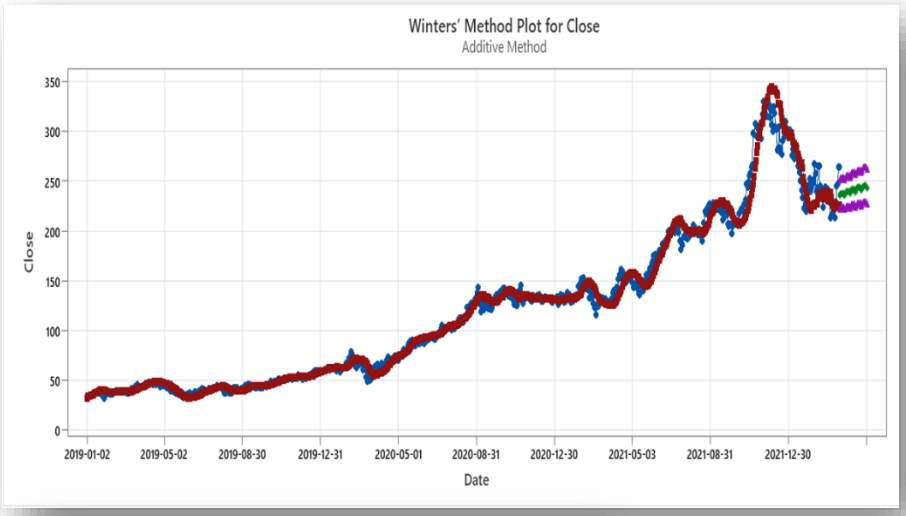
From I , II and III we can derive the forecasting equation :

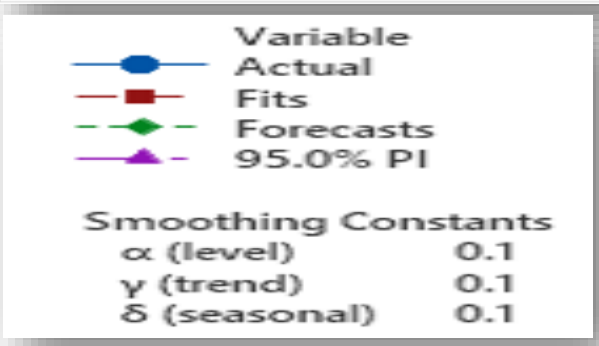
$$F= (L+K*B) * S$$

Now we try to forecast the upper , lower and forecasted value of Close values of the Nvidia Stock Market.

Forecasts

Period	Forecast	Lower	Upper
812	235.695	220.927	250.464
813	237.921	223.095	252.746
814	237.818	222.932	252.704
815	237.953	223.004	252.903
816	237.656	222.640	252.673
817	235.568	220.480	250.655
818	237.699	222.538	252.860
819	239.924	224.686	255.163
820	239.822	224.503	255.141
821	239.957	224.554	255.360
822	239.660	224.171	255.150
823	237.572	221.992	253.151
824	239.703	224.030	255.376
825	241.928	226.160	257.697
826	241.826	225.958	257.693
827	241.961	225.991	257.931
828	241.664	225.590	257.738
829	239.575	223.394	255.757
830	241.707	225.415	257.999
831	243.932	227.527	260.337
832	243.830	227.309	260.350
833	243.965	227.326	260.603
834	243.668	226.909	260.427
835	241.579	224.697	258.461
836	243.711	226.703	260.718
837	245.936	228.801	263.071
838	245.834	228.568	263.099
839	245.969	228.571	263.366
840	245.672	228.140	263.204
841	243.583	225.914	261.252





For the above graph we have taken ***0.1***(as constant value) for ***level*** , ***trend*** and ***seasonality***.

Over here we can see the actual stock price has shown a increasing growth with increasing rate over a constant period of time which is then accompanied by a little dip at the later stage and then again started increasing with an increasing rate. The slight dip might due to certain economic conditions that have affected the stock market such as fall of dollar rate or low revenue generated due to low sales , etc.

From the simple forecast (shown above) and triple exponential smoothening closing price shows us a steep growth over a later period of time . So for the time being we can say that both methods of forecast shows the exact result and exact increasing upward trend.

2.9.C PERFORMANCE METRICS

As we can see our model did quite well , so we can conclude our model fitting and forecast was quite right.
Let us also check the commonly used accuracy metrics to judge forecast results.

MAPE (Mean Absolute Percentage Error) is an error metric used to measure the performance of regression machine learning models. It is a popular metric to use amongst data scientists as it returns the error as a percentage, making it both easy for end users to understand and simpler to compare model accuracy across use cases and datasets.

MAPE returns error as a percentage, making it refreshingly easy to understand the 'goodness' of the error value. The lower the percentage, the more accurate the model.

```
MAPE: 6.198012417301349
```

Around **6.19% MAPE**(Mean Absolute Percentage Error) implies the model is about 93.81% accurate in predicting the test set observations.

CONCLUSION

The stock market plays a remarkable role in our daily lives. It is a significant factor in a country's GDP growth.In this paper we use Machine Learning model. By using machine learning we understand that in stock price prediction is used to discover patterns in data . Usually, a tremendous amount of structured and unstructured data is generated from stock markets. Using machine learning algorithms, it is possible to quickly analyze more complex data and generate more accurate results.

While performing the stationary check on the dataset we examined that the dataset is a non stationary one and that it has both trend and stationary component. We perform the detrend and deseasonalize techniques to remove both the component from the dataset. As we were well aware that for a stationary model we can use AR , MA , ARMA model and that for non stationary one we have to use ARIMA OR SARIMA model.

Our model being non stationary one we use the ARIMA (p,d,q) model. Now there are two ways of using ARIMA model :

- By choosing the optimal values of p,d,q manually , which might not be so effective , and
- Then can use the auto function to automatically detect the best parameters of the ARIMA model.

Once our model fitting has been done we then moved on to forecasting section were we have used Simple forecasting method for the forecast of Nvidia Stock closing price over the period of thirty days and then did the same using Holt Winter's Exponential Smoothening preferably known as Triple Exponential Smoothening which also gives us the same result like the previous one , i.e. an steep increasing trend over the period of time.

To check the result we used the performance metrics and from there we can conclude that our model did quite well and the accuracy rate was far above ninety percent.

FUTURE STUDY

In financial markets, a machine learning (ML) has become a powerful analytical tool used to help and

manage investment efficiently. ML has been widely used in the financial sector to provide a new mechanism that can help investors make better decisions in both investment and management to achieve better performance of their securities investment.

We the evolving nature of ML and the growth of Deep Learning (DL) , preferably Neural Networks one can apply them and obtain a more better and good quality model.

Besides Time Series and Linear Regression we can used models like SVR, KNN, XGBoost, LightGBM from ML and using the DL one can apply LSTM on Time Series to obtain high efficient models.

Such ML models can later be deployed on clouds such as Heroku , GCP , Azure , AWS EC2 , etc to make a more production and user interactive based model.

REFERENCES

- Central European Economic Journal.
- ***Linear regression of Machine learning*** by JasonBrowniee.
- Stock Market Prediction Using Machine Learning Techniques : ***A Decade Survey on Methodologies, Recent Developments, and Future Directions(MDPI)***.
- Kaleb Phipps , Martin Ratz , Dirk Muller , Veit Hagenmeyer , Ralf Mikut | **Feb , 2022** | ***REVIEW OF AUTOMATED TIME SERIES PIPELINE.***
- Julien Siebert , Janek Grob , Christof Schroth | **June , 2021** | ***A Systematic Review Of Python Packages for Time Series Analysis.***
- Wes McKinney , Josef Perktold , Skipper Seabold | **2011** | ***Time Series Analysis in Python with Statsmodels.***