Stock Market Analysis and Prediction using Time Series Analysis

A PROJECT REPORT

Submitted by

Rajdeep Singh (16BCE1358**)**

in partial fulfillment for the award of the degree of

Bachelor of Technology

in

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

Vandalur - Kelambakkam Road, Chennai - 600 127

May, 2020

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

DECLARATION

I hereby declare that the project entitled "**Stock Market Analysis and Prediction using Time Series Analysis**" submitted by me to the **School of Computer Science and Engineering, VIT Chennai, 600127** in partial fulfilment of the requirements of the award of the degree of **B.Tech CSE** is a bona-fide record of the work carried out by me under the supervision of **Prof. Christy Jackson**. I further declare that the work reported in this project, has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma of this institute or of any other institute or University.

Place: Chennai Signature of the Candidate (s)

Date: 25 May, 2020 (Rajdeep Singh)

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computing Science and Engineering

CERTIFICATE

This is to certify that the report entitled "**Stock Market Analysis and Prediction using Time Series Analysis**" is prepared and submitted by **Rajdeep Singh (16BCE1358)**, to VIT Chennai, in partial fulfilment of the requirement for the award of the degree of B.Tech CSE programme is a bona-fide record carried out under my guidance. The project fulfils the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: **Prof. Christy Jackson**

Date:

Signature of the Internal Examiner
Name:
Date:

Signature of the External Examiner
Name:
Date:

Approved by the **B. Tech CSE Program Chair**

Name: **Dr. Justus S**
Date:

(Seal of SCSE)

ABSTRACT

Over the years stock market has been considered a very risky investment by people around the globe. This project aims to understand the historical data of stock market and derive analysis from it to reduce the gap of knowledge between the market behavior and the investor. A stock data comprises of lot of statistical terms which are difficult to understand by a normal person who wants to step into stock market investments, this project aims at reducing the gap of knowledge.

This project aims to tell the market scenario of the future by supporting it with statistical answers. Stock market volatility, Daily returns, cumulative returns, Correlations between different stocks, Sharpe Ratio of the stocks, CAGR value, Simple Moving Average are some important statistical terms to understand the risk of the investment in the stocks. For prediction of the future behavior of stocks I have worked on ARIMA models, Monte Carlo Method and Forecasting using Facebooks prophet library.

Non stationary times series data has three components into which it can be decomposed which are trend, seasonality and Residual. Prophet library will also be used to decompose these components into weekly and yearly trends of the data.

Acknowledgement

It is my pleasure to express with deep sense of gratitude to Prof. Christy Jackson, School of Computer Science and Engineering (SCSE), VIT, Chennai campus, for his/her constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Machine Learning.

I would like to express my gratitude to Chancellor Dr. G. Viswanathan, VP Dr. Sekar Viswanathan, VC Dr. Anand A. Samuel, PRO-VC Dr. V. S. Kanchana Bhaaskaran for providing with an environment to work in and for his inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Justus S, Head of the Department & Co-ordinator, Dr. Kumar R, and Project Coordinator Dr. B V A N S S Prabhakar Rao, B. Tech. Computer Science and Engineering, SCOPE, VIT Chennai Campus, for their valuable support and encouragement to take up and complete the thesis.

Special mention to Dean, Dr. Jagadeesh Kannan R, Associate Dean, Dr. Geetha S, SCOPE, VIT Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

All teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Rajdeep Singh (16BCE1358)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

Abbreviations	Full Form	Description
CAGR	Compounded Annual Growth Rate	Used to Evaluate returns Per year.
ASR	Annual Sharpe Ratio	Risk Adjusted Return from stock over one business year.
SMA	Simple Moving Average	Average Price over a period N.
ADF	Augmented Dickey Fuller Test	Metric used to check stationarity of data.
ACF	Autocorrelation Function	Metric used to Check Correlation of data with itself.
PACF	Partial Autocorrelation Function	Metric used to Check Correlation of data with respect to lag1 of data.
AIC	Akaike Information Criteria	Metric used to check good fit of the model.
MSE	Mean Squared Error	Average of squared values of Errors.
MAE	Mean Absolute Error	Average sum of all absolute errors.
RMSE	Root Mean Squared Error	Root of Average of squared values of Errors.
MAPE	Mean Absolute Percentage Error	Average percentage of all absolute errors.
ARIMA	Auto Regressive Integrated Moving Averages	Model for forecasting.

Chapter 1

Introduction

1.1 EXECUTIVE SUMMARY

This study attempts at understanding the stock market data by analysis using various statistical calculations and visualizations of the historical data of the stocks. This attempt also includes prediction of future stock prices using algorithms which can be beneficial for understanding the time series data. This study also required knowledge of an investor who studies and understands stocks statistical data and various parameters and can tell the reason of stock price to behave in a certain manner, further which helps the investor to invest wisely in the market and gives good returns.

Stock market exact accurate predictions cannot be done solely on the historical data available to us, or in the other means future stock prices are not dependent on the previous day's prices. Predictions in stock market are done based on statistical parameters and calculations which gives us a probabilistic measure and tells whether it is a right stock to invest or not.

This study includes comparison analysis on parameters like daily returns, cumulative returns, Compounded Annual Growth Rate of the share price, understanding the Sharpe Ratio, finding the correlation between different shares, plotting the rolling mean and removing the noise from the plot, component decomposition of time series data into trend, seasonality and residual. For forecasting, this study includes ARIMA model, Monte Carlo Simulation and Prophet procedure by Facebook.

1.2 INTRODUCTION

Time Series is the series of continuous data points index in order of date and time. Data connected through continuous series of time data. Analysis of time series data is done to extract meaningful statistics and other characteristics of data. After statistical calculations of time series data and after data analysis one can understand the behavior of the stock and deduce the amount of risk involved in it before making any investments.

Time series forecasting is a step further in making a value step towards understanding of future behavior. It refers to use of models to predict future values based on previously observed values. Models used in this study are Auto Regressive Integrated Moving Average (ARIMA) model, Augmented Dickey Fuller Test tells about Stationarity of Time Series Data, Monte Carlo Model is used to tell possible future predictions of a stock for a time period, prophet library by Facebook is very robust in processing the time series data and giving future predictions based on daily trend of data, weekly trend of data and yearly trend of data.

1.3 PROBLEM STATEMENT

To understand the time series data of stock market and understand the risk involved in a stock through statistical calculations and use of prediction algorithms to understand the possible future behavior of the stock to draw assumptions for the future investments.

1.4 OBJECTIVES

This project is aimed to understand and predict the stock behavior through statistical calculations and visualizations of historical data analysis.

The objectives are:

- To find change in prices of the stock overtime.
- To find the daily and cumulative return of the stocks.
- To find and plot the moving average of various stocks.
- To find the correlation between different stocks' closing prices and daily returns.
- To find the sharpe ratio of the stock which tells us about the risk adjusted return.
- To find the compounded annual growth rate of the stocks over last 10 years.
- To predict the future stock behavior and the possible stock prices.

1.5 MOTIVATION

Motivation behind this topic of study was the gap of knowledge most people have before they start investing in the stock market. Every year smart investors make good chunk of money out the stock market. Self-made billionaire Warren Buffet is one such example who earned most of his wealth through smart investing. Prediction of future of stock behavior is another motivation for me. This gap of investment knowledge needs to be reduced.

Chapter 2

Background

2.1 BACKGROUND

Stock market data over the years was considered to be very unpredictable and investment in the stock market was not very difficult for newcomers. Moreover, so much data and analytics was also not so easily available to the people. But Data Scientists, statisticians and tried to reduce the gap bit by bit over the years. Now a day's mutual fund applications use AI models, use certain statistic benchmarks to make it easy for a new comer to understand it in some extent.

Moreover, there has been a lot study done by people around the globe to predict the future prices of the stocks but stock market does not solely depend on the historical data. It is also affected by the sentiments of the people, which depend on some future events and we know we cannot predict future events 100% accuracy.

2.2 LITERATURE SURVEY

The author Banarjee D. [1] has done his study on forecasting of National Stock Exchange data using ARIMA model and has done a comparative study on different values of p, d and q on ARIMA and did validation check of forecasted stock price with actual stock price. In his study ARIMA (1,0,1) gave the best fit compared to other models.

The authors Viswam, N., & Reddy, G. S. [2] did study on historical stock market data predicted future prices using ARIMA model and they used MACD model to better analysis of the data.

The authors Angadi, M. C., & Kulkarni, A. P. [3] used ARIMA model for prediction of stock prices, for p, d, q values they used auto ARIMA to get the best fit for the model. Obtained results reveal that ARIMA model has strong tendency to make short time predictions.

The authors Devi, B. U., Sundar, D., & Alli, P. [4] used NIFTY MIDCAP 50 as the index and selected top four MIDCAP companies. They used ARMA and ARIMA models to predict the future stock prices and used AIC and BIC criteria to get the best fit for the model.

The authors Varghese, A., Tarhen, H., Shaikh, A., Banik, P., & Ramadasi, A. [5] used ARMA, Moving average, and ANN model to predict the future prices of the stock and used maximum likelihood estimator and Yule-Walker estimation to check the validation of their models.

The authors Sharma, A., Modak, S., & Sridhar, E. [6] using LSTM model from RNN to predict the random nature of stock prices in future. MSE value of the model came out to be significant and improved.

The authors Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. [7] used NSE listed companies as the stock price data. They used sliding window approach on non-linear model RNN, LSTM and CNN and linear model ARIMA. Non-linear model outperformed the linear model during error calculation.

The authors Mondal, P., Shit, L., & Goswami, S. [8] did a study on effectiveness of ARIMA model in forecasting security values. They used Indian Stock market data from NSE for the analysis. AIC has been used for selecting the best ARIMA model.

Th authors Wang, J., & Wang, J. [9] used principle component analysis and Stochastic time effective neural networks for forecasting the future and used MAE, MAPE, MSE and RMSE to calculate the performance of the model.

Chapter 3

Technical Specification

3.1 SOFTWARE SPECIFICATION

Anaconda Distribution – Anaconda3 2020.02(version)
Anaconda software is used for scientific computing applications using Python and R programming languages. It consists of most of the necessary libraries/packages by default for development, management and deployment in its environment. It is useful for applications like data science, machine learning, predictive analysis and large-scale data processing.

Framework used - Jupyter Notebook 6.0.2(version)
This open-source software is helpful in creating and sharing documents which contains codes, equations, visualizations and narrative texts. Its uses include data cleaning and transformation, statistical modeling, numerical simulation, machine learning, data visualization, and much more.

Language used - Python Programming Language – Python 3.8.1(version)

3.2 PACKAGES INSTALLED

3.2.1 NumPy

NumPy package is used for scientific computing applications. Numpy consists of a powerful object with N-dimensional arrays, includes tools for integrating C/C++ code and Fortran code, broadcasting functions, linear algebra, Fourier transform functions, and various features for random numbers. Besides that, it can hold generic data on its multi-dimensional objects and is helpful in defining arbitrary data-types. With all that NumPy can easily and speedily integrate and work with various type of databases.

3.2.2 Pandas

Pandas library is a flexible, fast, and expressive set of data structures designed to work with structured and time series data. These powerful data structures used for time series operations, statistics and data analysis operations. This makes it useful for doing practical, real world data analysis which makes it fundamentally high-level building block of Python.

3.2.3 Matplotlib

Matplotlib library is used for creating static, animated, and interactive visualizations in Python language.

3.2.4 Seaborn

Seaborn library is based on matplotlib package and is used for data visualization operations. It consists with a high-level graphical interface for plotting attractive and informational statistical plots in python.

3.2.5 pandas_datareader

This library is helpful in providing updated data for remote access for pandas library operations.

3.2.6 Datetime

The datetime module provides classes for manipulating dates and time data for a time series data.

3.2.7 Scipy

SciPy is an open-source library which is used for technical and scientific computations. This library consist of functions for linear algebra, optimization, interpolation, FFT, some special functions, integration, signal and image processing and other tasks which are commonly used for scientific and engineering practices.

3.2.8 Statsmodels

Statsmodels library of python provides the user with classes and functions for calculations with many different statistical algorithms, and also for conducting statistical tests, and for statistical exploration on data.

3.2.9 Pmdarima

Pmdarima is a statistical library designed to complete the void present in time series analysis capabilities of python.

3.2.10 Sklearn

It is a open source ML library for the Python programming language. It features various regression, clustering and classification algorithms including random forests, support vector machines, k-means, gradient boosting and DBSCAN, and is designed to work in conjunction with the Python numerical and scientific libraries like SciPy and NumPy.

3.2.11 Fbprophet

Prophet is a library made for forecasting time series data based on an additive model where non-linear trends in data are made to fit with daily, weekly and yearly seasonality, along with holiday effects. It works best with time series data that have strong seasonal effects and several seasons in historical data of stocks. Prophet is very robust to the missing data, shifts in the trend, and typically handles the outliers well.

Chapter 4

Data

4.1 DATA DESCRIPTION

Data used for this project is day wise historical time series data of stock of past 10 years in numerical form.

Dataset size – 2462 (10 Business years)

Data Source used – Yahoo finance

Dataset Description: Dataset imported from yahoo finance consists of 7 columns of consisting of Open, High, Low, Close, Adj. Close, Volume and Date as index.

Date (Index of the Dataset)

Dates of all Business Days in a year.

Open

Depicts the Opening price of the Security.

Close

Depicts the Closing price of the Security.

High

Depicts the Highest value gained by the stocks in a particular day.

Low

Depicts the Lowest value gained by the stocks in a particular day.

Adj. Close

The adjusted closing price is calculated after analyses of the stock's dividends, stock splits and the new stock offerings which determines a new value of the stock know as adjusted price.

Volume

Volume, or trading volume, is the amount of a security that was traded during a given period of time.

Chapter 5

Statistical parameters used

5.1 DAILY STOCK RETURN

This parameter tells us about how much the stocks gained or lost per day per share. It is calculated by subtracting previous day's closing price from todays' closing price.

5.2 STOCK VOLATILITY

Volatility is a measure of the dispersion of returns for a given stock or the market index. Mostly, the higher the volatility, the riskier the is stock. It is often measured as either the standard deviation or variance between returns from that same stock or the market index.

In the stock markets, volatility is often associated with big swings in the price in either direction of the trend. For ex., when the market gains and loses value more than one percent over a specific time period, this is known as volatility of a market.

$$Daily\ Volatility\ Formula = \sqrt{Variance}$$

$$Annual\ Volatility\ Formula = \sqrt{252} \times \sqrt{Variance}$$

5.3 CUMULATIVE STOCK RETURN

This parameter tells us about how much the stocks gained or lost per share over time, independent of the time period. Cumulative Return is equal to:

$$\frac{(Current\ Price\ of\ the\ Security) - (Original\ Price\ of\ Security)}{OriginalPrice\ of\ Security}$$

5.4 COMPOUNDED ANNUAL GROWTH RATE(CAGR)

CAGR is the rate of return by which tell us that this rate would be required for an company to grow from its starting value to its ending value, it is assumed that the profits gained were again invested at the end of each business year of an investment firm.

$$CAGR = \left(\frac{V_{final}}{V_{Begin}}\right)^{1/t} - 1$$

Where:

CAGR = compound annual growth rate

V_{begin} = beginning value

V_{final} = final value

T = time in years

5.5 CORRELATION OF STOCKS

Correlation(r) is a statistical measure which tells us the amount to which two variables move in relation with each other. In finance, the correlation can measure the movement of a stock price with stock markets benchmark index.

$$r = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum(X - \bar{X})^2}\sqrt{(Y - \bar{Y})^2}}$$

Where:

r = the correlation coefficient

$\bar{X}$ = the average of the observations of variable X

$\bar{Y}$ = the average of the observations of variable Y

5.6 SHARPE RATIO

Sharpe ratio is a measure of risk adjusted returns of a financial portfolio. Sharpe ratio is a measure of excess return one gets over the risk free rate, relative to its standard deviation.

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

Where:

R_p = return of portfolio

R_f = risk free return rate

σ_p = standard deviation of the portfolio excess return

5.7 SIMPLE MOVING AVERAGE

Simple moving average (SMA) is a simple method for technical analysis that smooths out price data of stock market by updating average price of stocks constantly. This average is calculated over a specific period of time, like 10 days, 30 minutes, 20 weeks or any time period the investor chooses. Moving average techniques are very popular and can be calculated for any time frame, suiting both long-term investments and short-term investments.

$$SMA = \frac{A_1 + A_2 + \cdots + A_n}{n}$$

Where:

A_n= The price of a stock during a period n

n = the number of total periods

Chapter 6

Data analysis

6.1 ANALYSIS OBTAINED FROM THE DATA

1 What is change in price of stock over time?

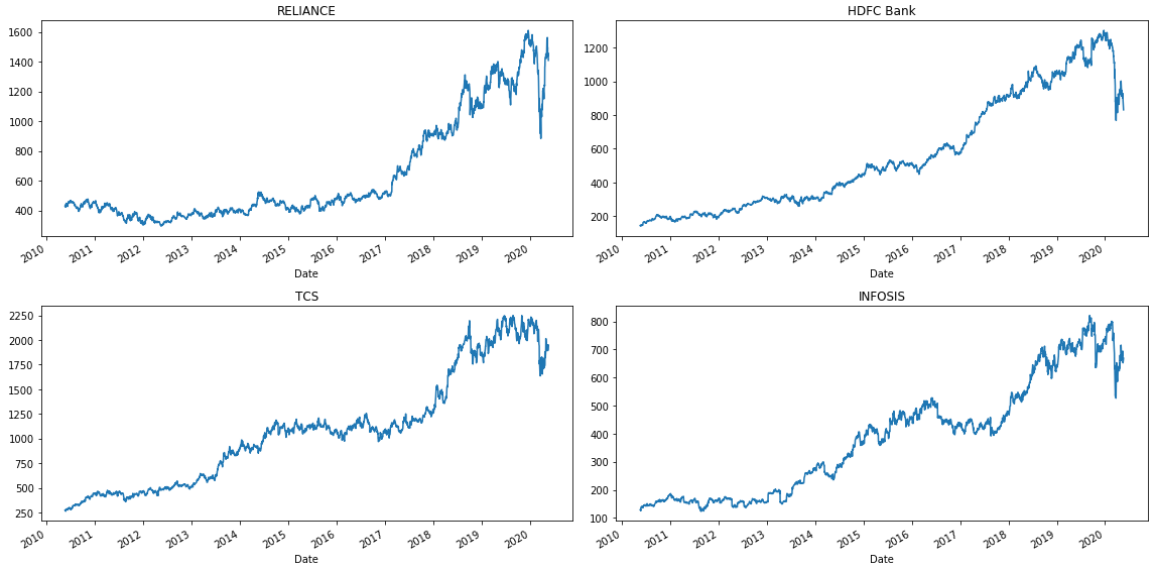


Fig 6.1: Plots of Stock Prices overtime

Prices of stocks can be seen increasing with an upward trend in plot. Moreover, we can observe two Tech companies TCS and INFOSIS have a close similarity in the plots whereas both of them have a big difference in the Stock prices which can be observed on the y axis.

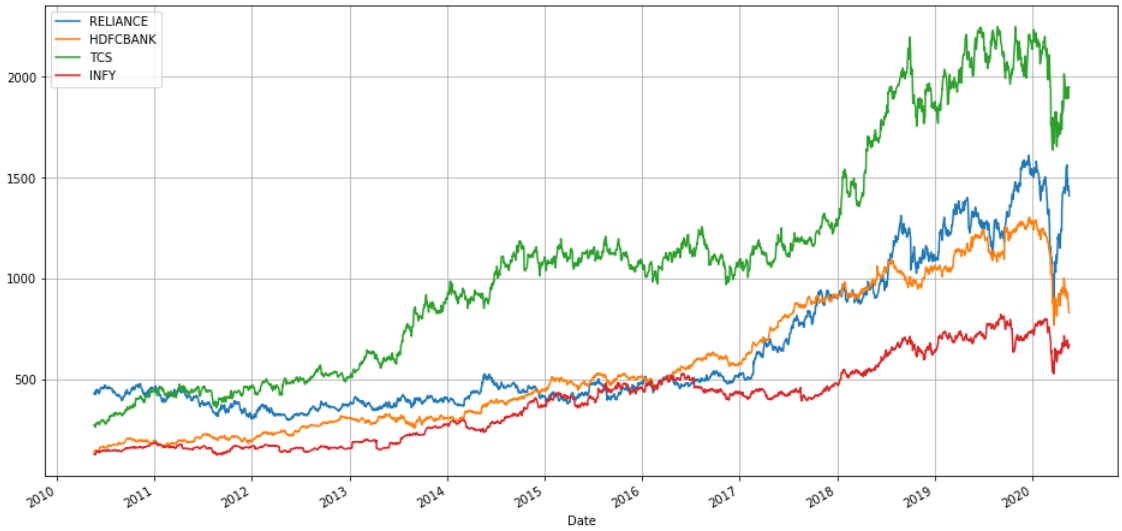


Fig 6.2: Combined plot of stock for price Comparisons

Combined plot of all the stocks shows the clear price difference between TCS and INFOSIS and the value gained by the stock over time. TCS Stocks outperformed other companies in gaining value over the time.

2 What is volume of stocks traded over the time period?

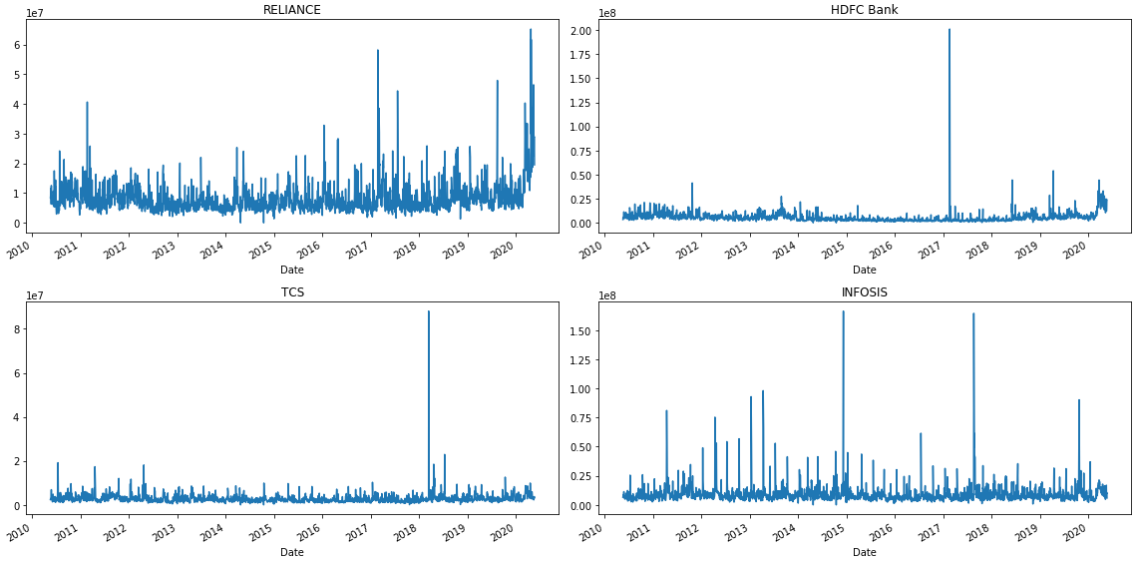


Fig 6.3: Volume Traded of stocks overtime

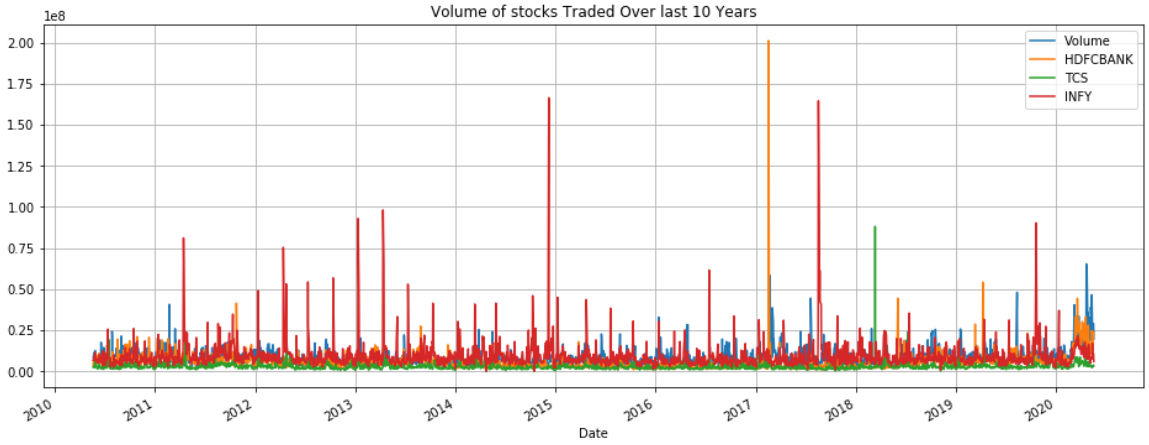


Fig 6.4: Combined plot of Volume Traded overtime

Combined plot of Volume traded of all the four stocks made the plot look very cluttered. To get a clear understanding between the Values we will take rolling mean of volume traded of 30 days.

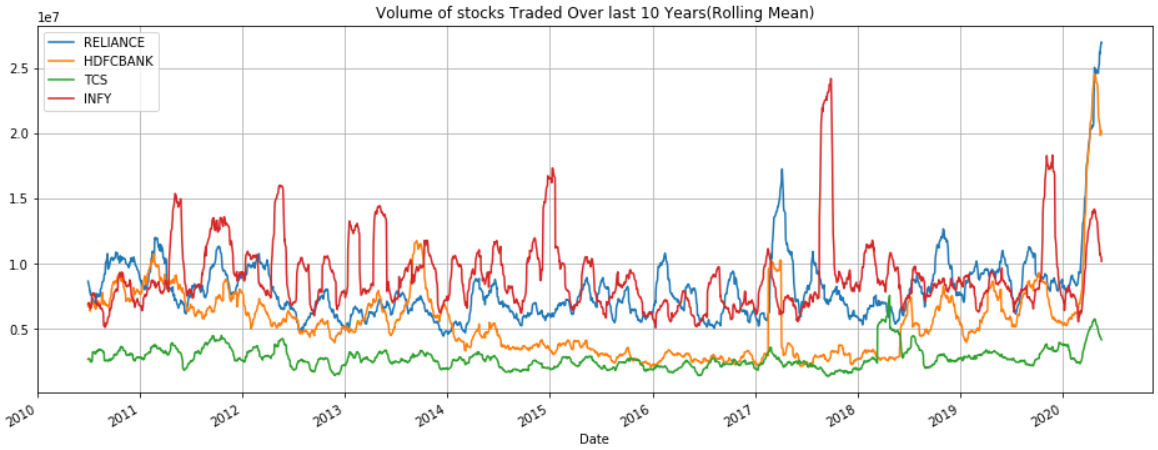


Fig 6.5: Comparative analysis of Volume Traded with rolling mean of 50

By observing this plot, we can analyze that higher is stock price lesser will be the volume traded. For ex. If a person invests 10000 Rs in each of them, higher is the price of the stock lesser will be the number of stocks bought.

3 What was the daily return from the stocks?

Date	Reliance	HDFC	TCS	INFY
2010-05-21	-0.004400	-0.013183	-0.015480	-0.007306
2010-05-24	0.027221	-0.001095	-0.001530	0.007223
2010-05-25	-0.036181	-0.011400	-0.025153	-0.025765
2010-05-26	0.022726	0.013306	0.054964	0.086040
2010-05-27	0.014087	0.037669	0.005217	0.008510
...	...	...	...	...
2020-05-13	0.021217	0.028950	0.000077	0.009452
2020-05-14	-0.040429	-0.036598	-0.024261	-0.051862
2020-05-15	0.016331	-0.006210	-0.004968	-0.008889
2020-05-18	-0.012779	-0.057986	0.027841	0.017783
2020-05-19	-0.022107	-0.007171	0.001568	0.007079

2461 rows × 4 columns

Tab 6.1: Daily Return of the Stocks

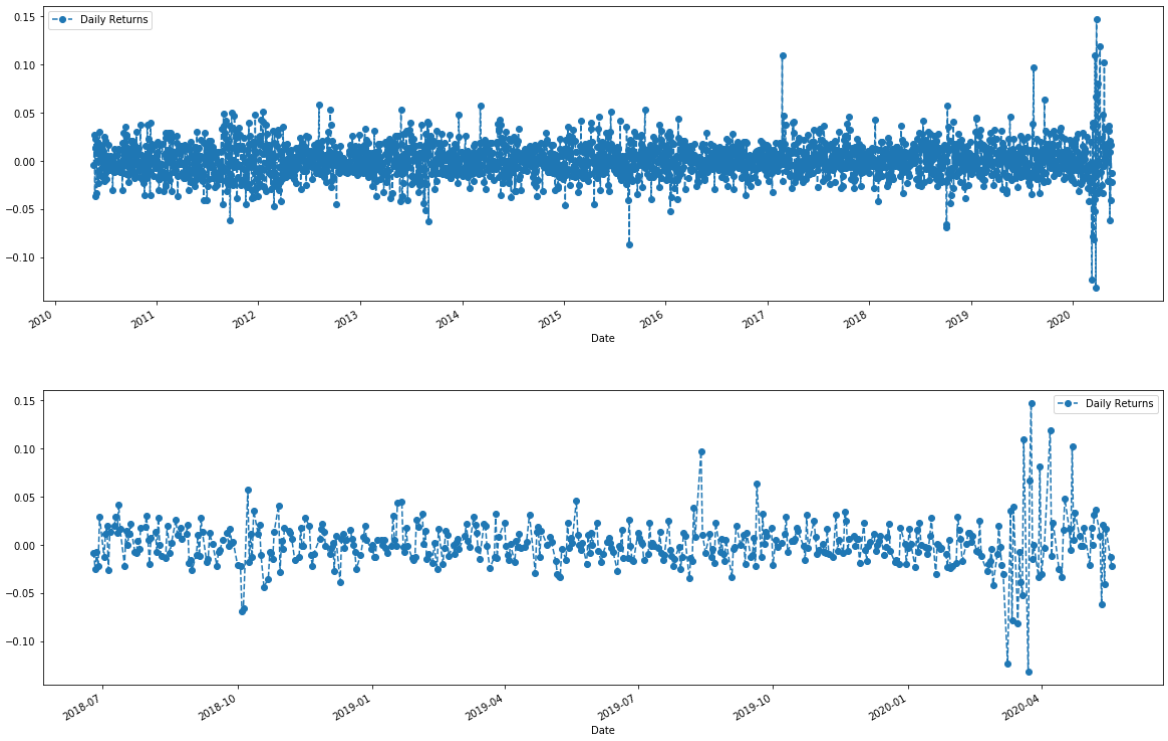


Fig 6.6: Daily Returns plot of Reliance stock

Daily returns can be considered as the value gained or lost by stock with respect to the previous day. We can see a graph of daily returns with mean close to zero.

4 Which stock has higher volatility with respect to others?

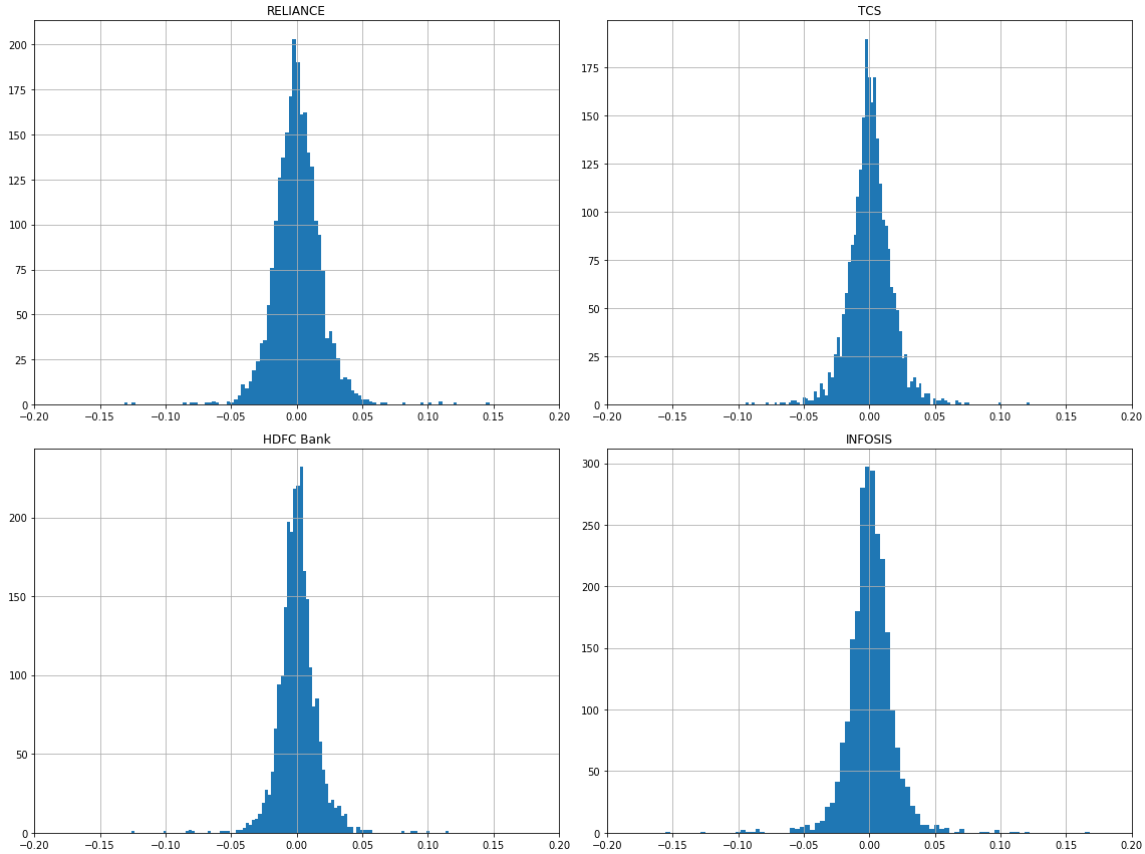


Fig 6.7: Histogram plot to check volatility of stocks

Histogram of daily returns tells us that thicker the histogram will be, more will be its volatility, that is that stock has more risk.

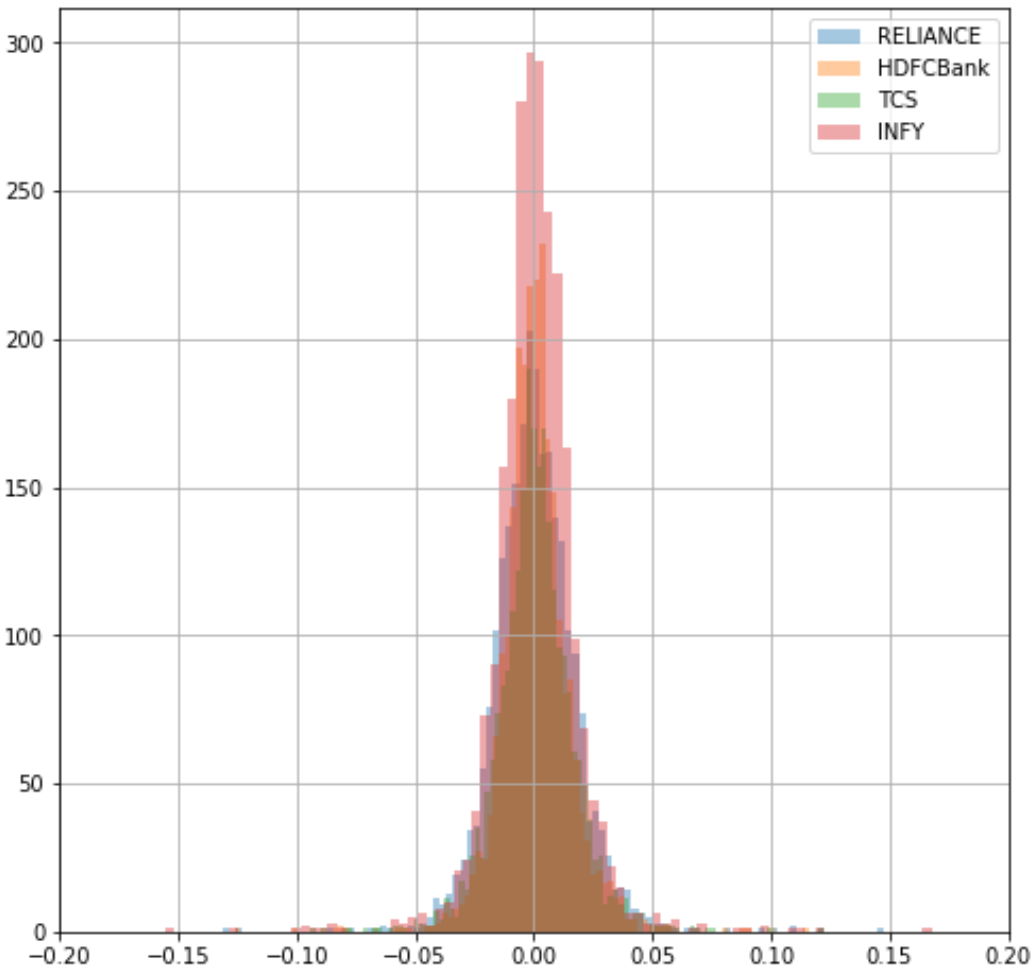


Fig 6.8: Histogram plot to find compare volatility between stocks

This combined plot tells INFOSIS and RELIANCE have more volatility but it doesn't give a very clear observation. Let's proceed to find it out with kernel density plot.

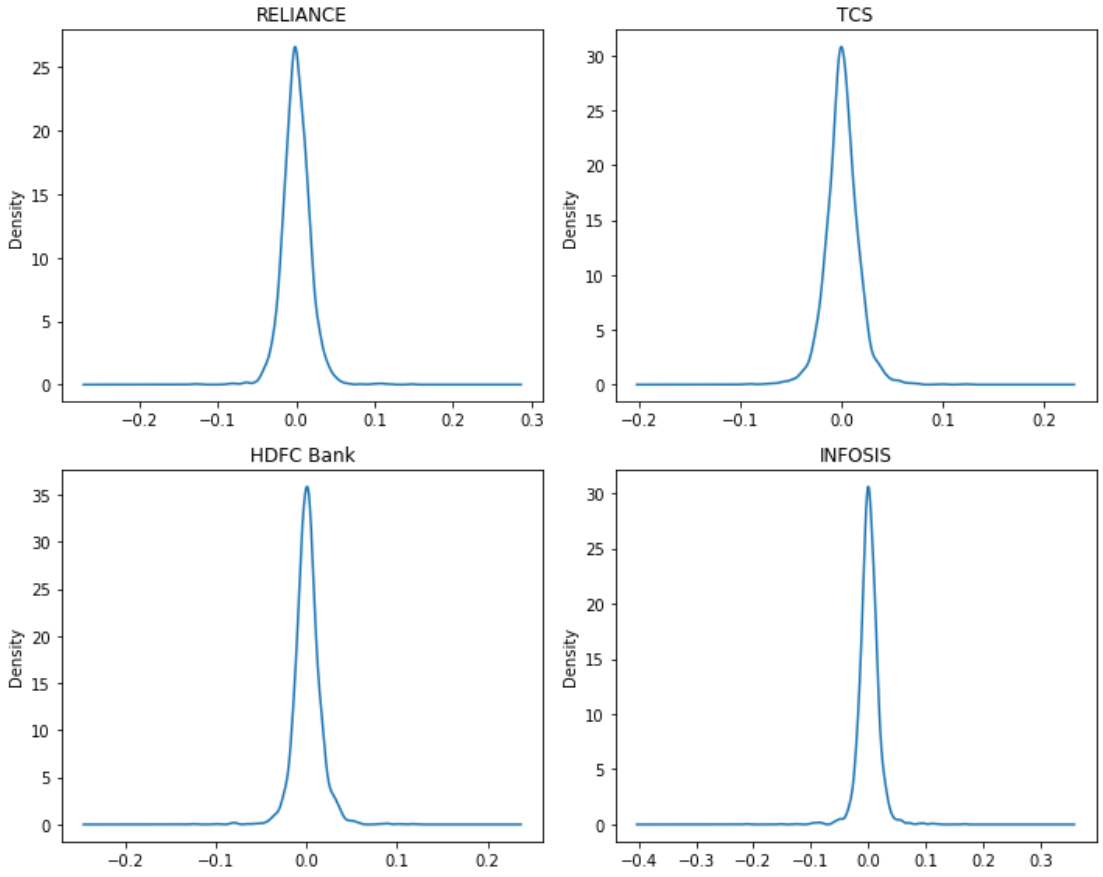


Fig 6.9: Kernel density plot of the stocks

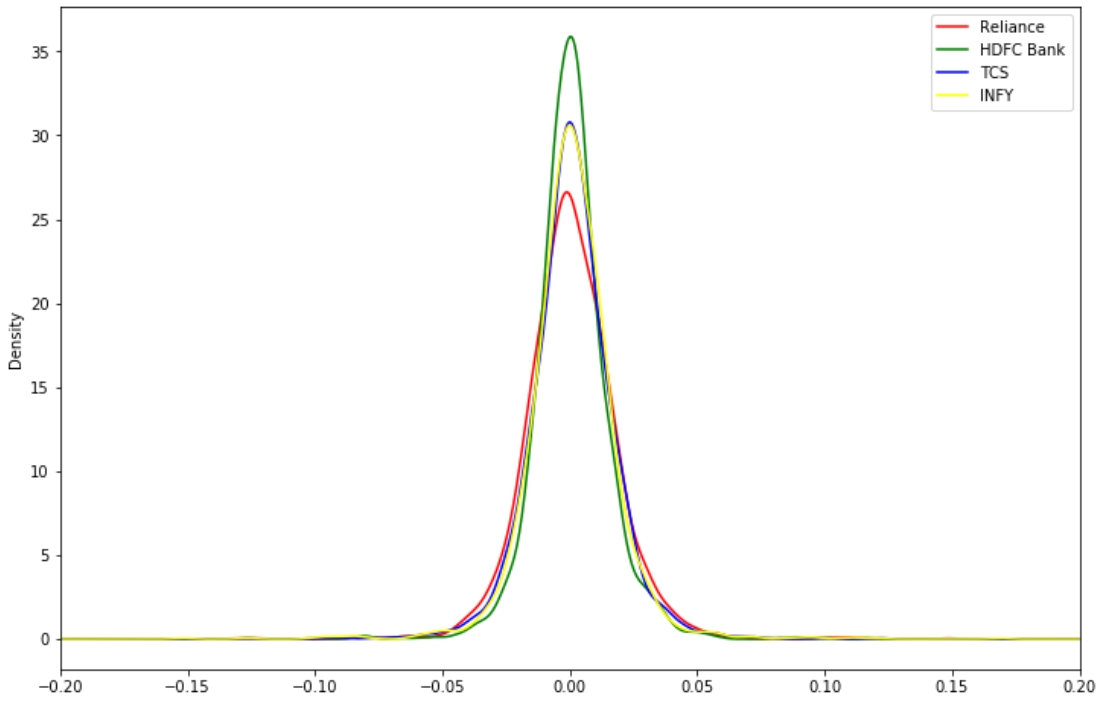


Fig 6.10.1: Kernel Density plot to compare Volatility

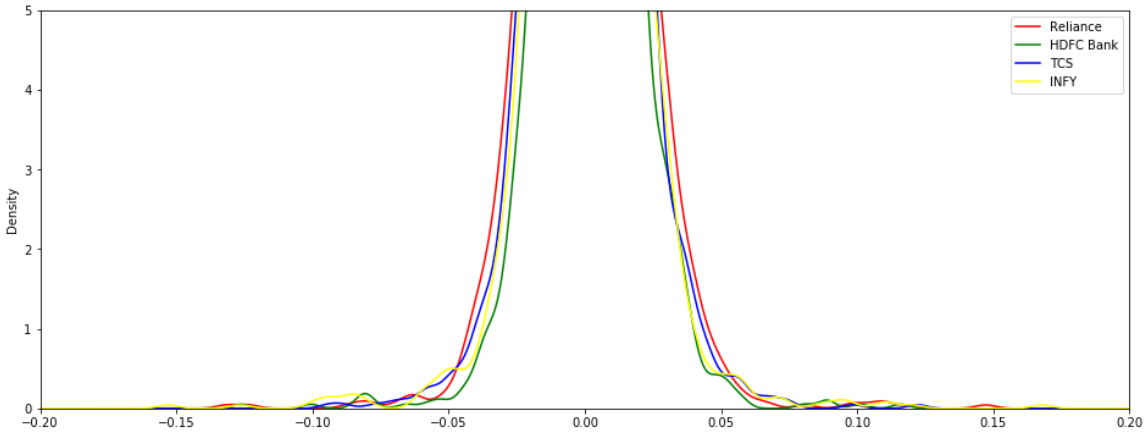


Fig 6.10.2: Kernel Density plot to compare Volatility (Zoomed)

From here we can clearly analyze and tell that Reliance stocks are most volatile, and HDFC Bank Stocks are least volatile.

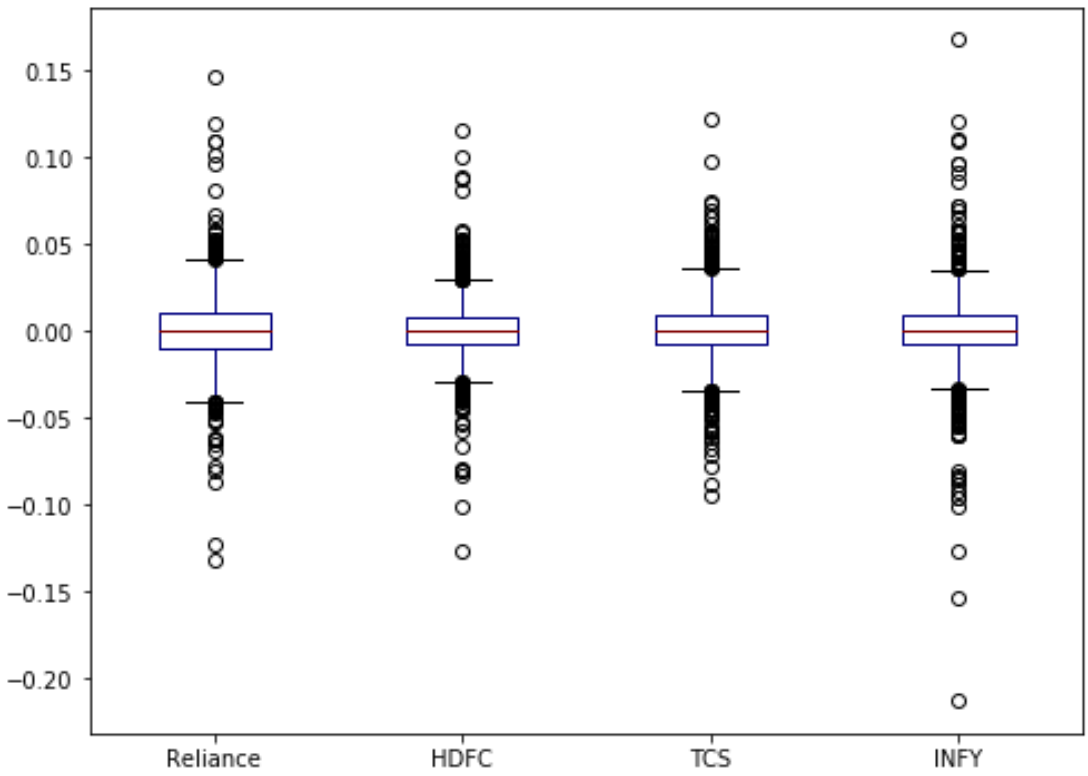


Fig 6.11: Box plot to compare volatility of stocks

Through box plot we can observe that INFOSIS has wider graph due to some outliers, neglecting those above analysis is completely supported.

5 What is the cumulative return of the stocks over the time?

Date	Reliance	HDFC	TCS	INFY
2010-05-20	1.000000	1.000000	1.000000	1.000000
2010-05-21	0.995600	0.986817	0.984520	0.992694
2010-05-24	1.022701	0.985737	0.983014	0.999865
2010-05-25	0.985699	0.974500	0.958288	0.974103
2010-05-26	1.008100	0.987466	1.010959	1.057914
...	...	...	...	...
2020-05-13	3.485026	6.451662	7.129287	5.389721
2020-05-14	3.344130	6.215545	6.956325	5.110199
2020-05-15	3.398742	6.176945	6.921769	5.064777
2020-05-18	3.355308	5.818771	7.114477	5.154845
2020-05-19	3.281134	5.777042	7.125630	5.191338

2462 rows × 4 columns

Tab 6.2: Cumulative Return of the Stocks over time

In the last 10 years, TCS gave us the maximum return, amount invested in TCS stocks have surged up to 7 times in last 10 years. while Reliance gave us minimum return where prices surge up to 3.3 times approx. With least volatility among the stocks HDFC has Stock seems to very promising with its returns.

6 What is the Compounded Annual Growth Rate value of the stocks?

```
Reliance     12.616596
HDFC Bank    19.170991
TCS          21.697688
Infosis      17.903835
Nifty         6.022352
dtype: float64
```

Tab 6.3: CAGR value over last 10 years

In the past 10 years, TCS attained really excellent CAGR value and Reliance have the least Annual growth rate. All the four stocks have tended to perform better than what the market have received over last 10 years.

7 What is the correlation between stocks and between stocks and the market index?

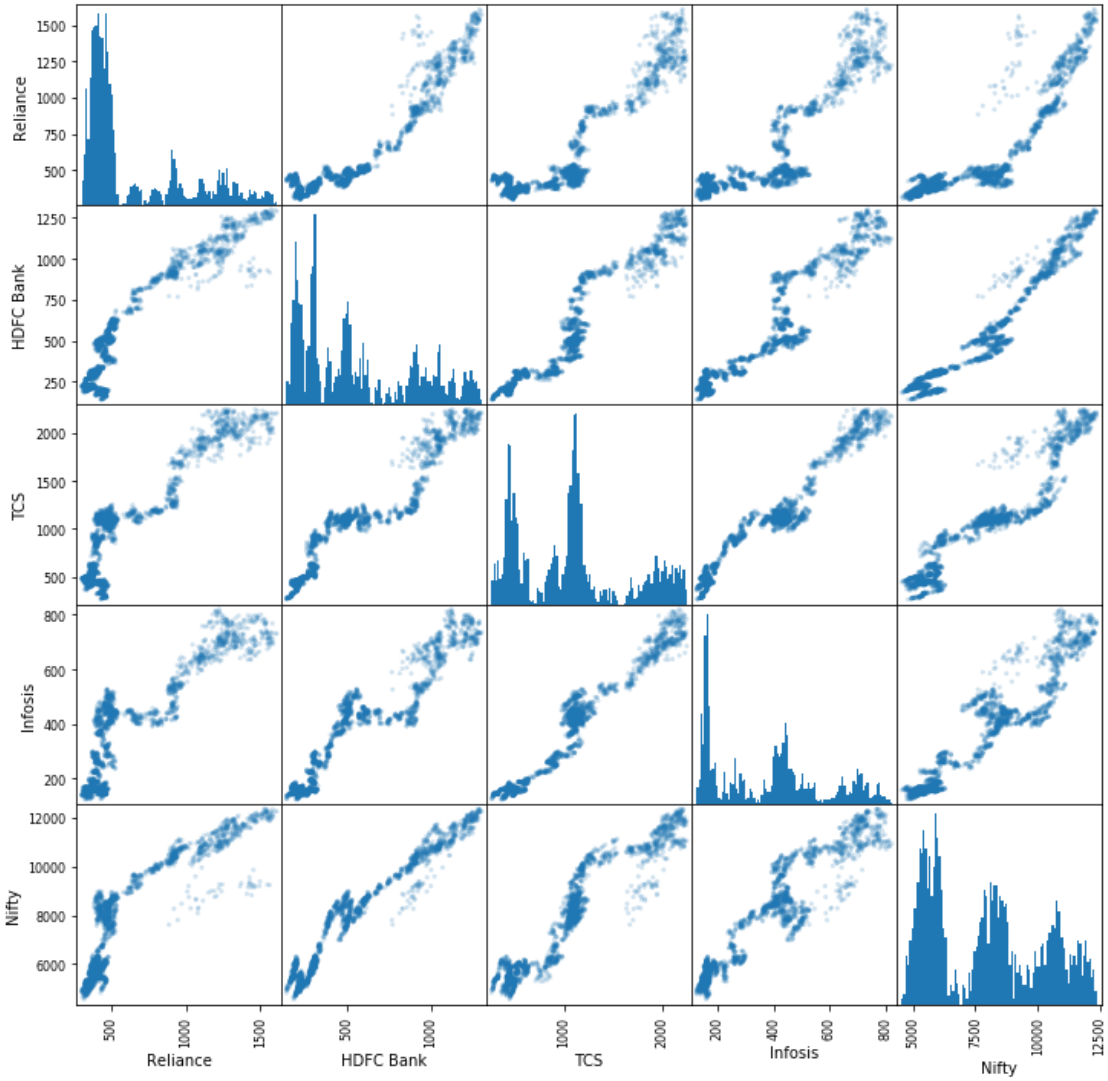


Fig 6.12: Correlation using Scatter plot

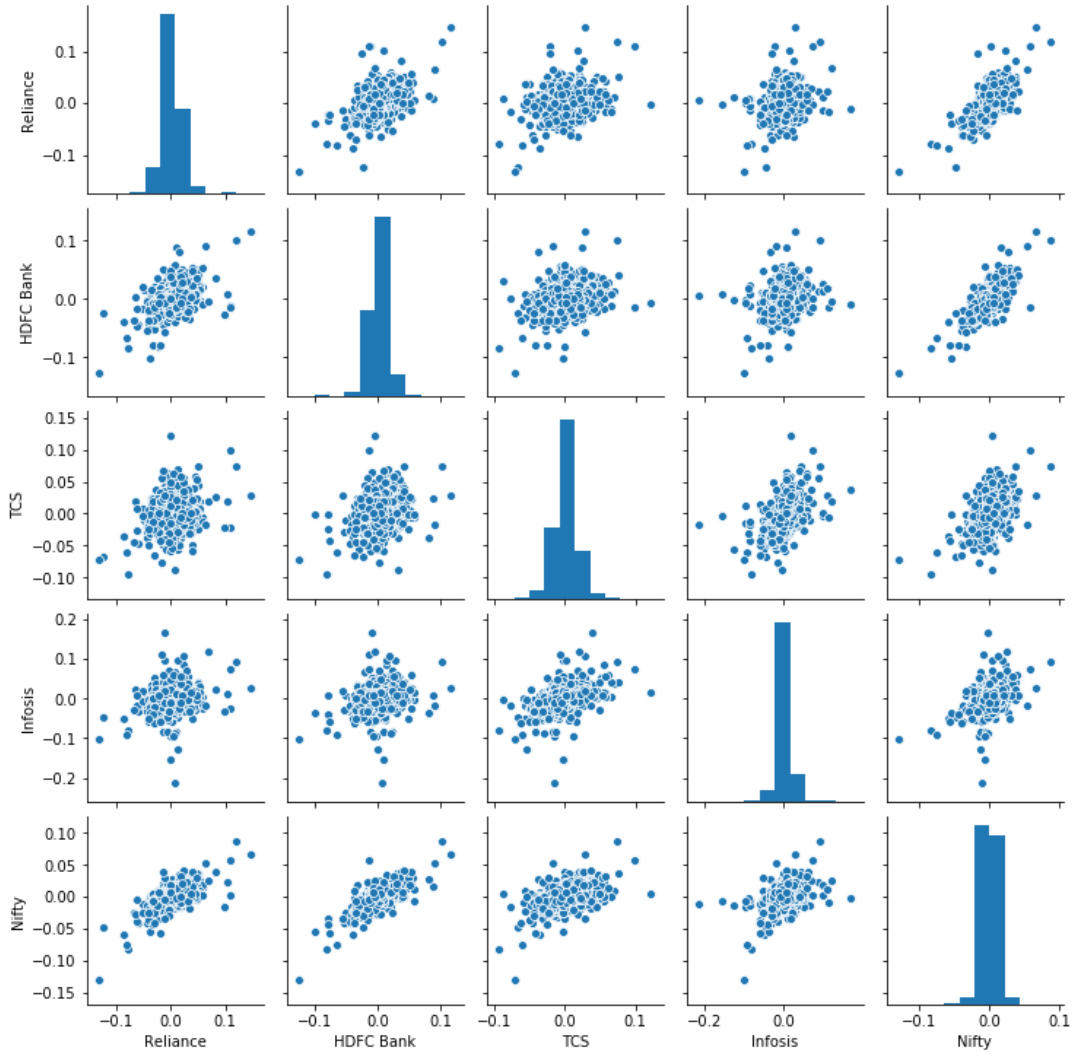


Fig 6.13: Correlation using Scatter plot using Seaborn

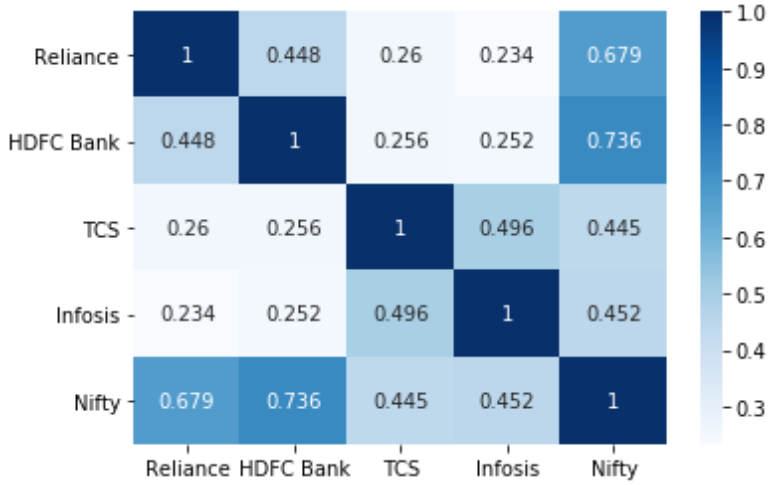


Fig 6.14: Correlation of Daily returns using Heat Map

We can observe market index NIFTY has good correlation with the stocks. Moreover, both the tech companies seem to have a correlation which tends to slightly weak.

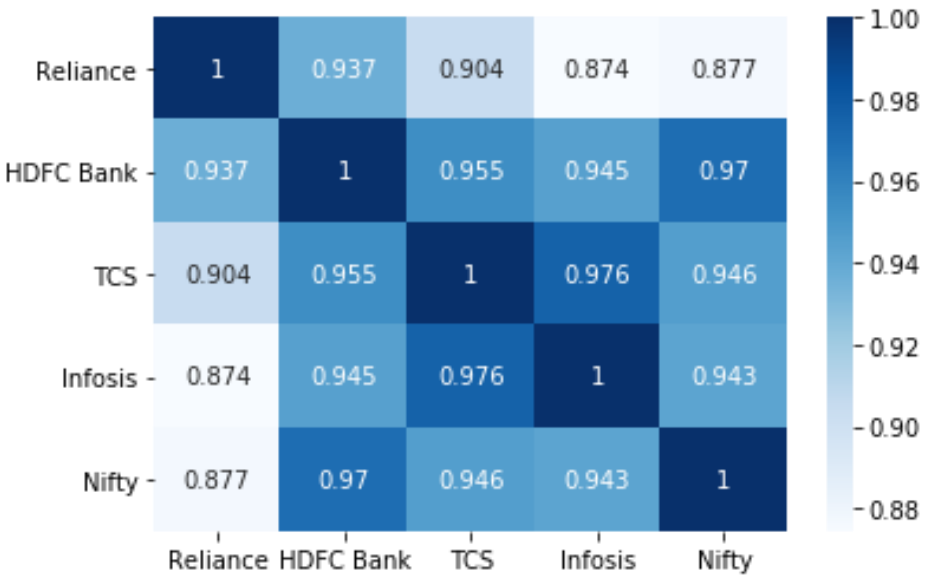


Fig 6.15: Correlation of Closing Price using Heat Map

Market index is closely related with all the four companies. Moreover, we can observe highest correlation between the tech companies.

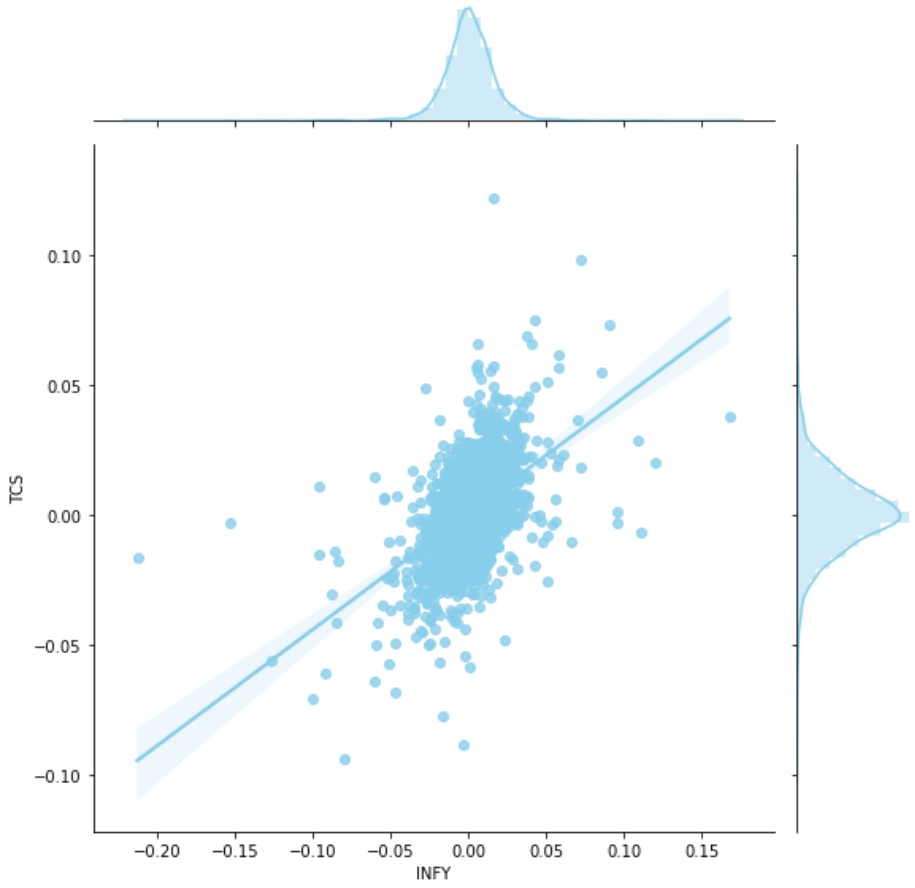


Fig 6.16: Regression Pair plot between TCS and INFOSIS

8 What is the Simple Moving Average of the stocks?

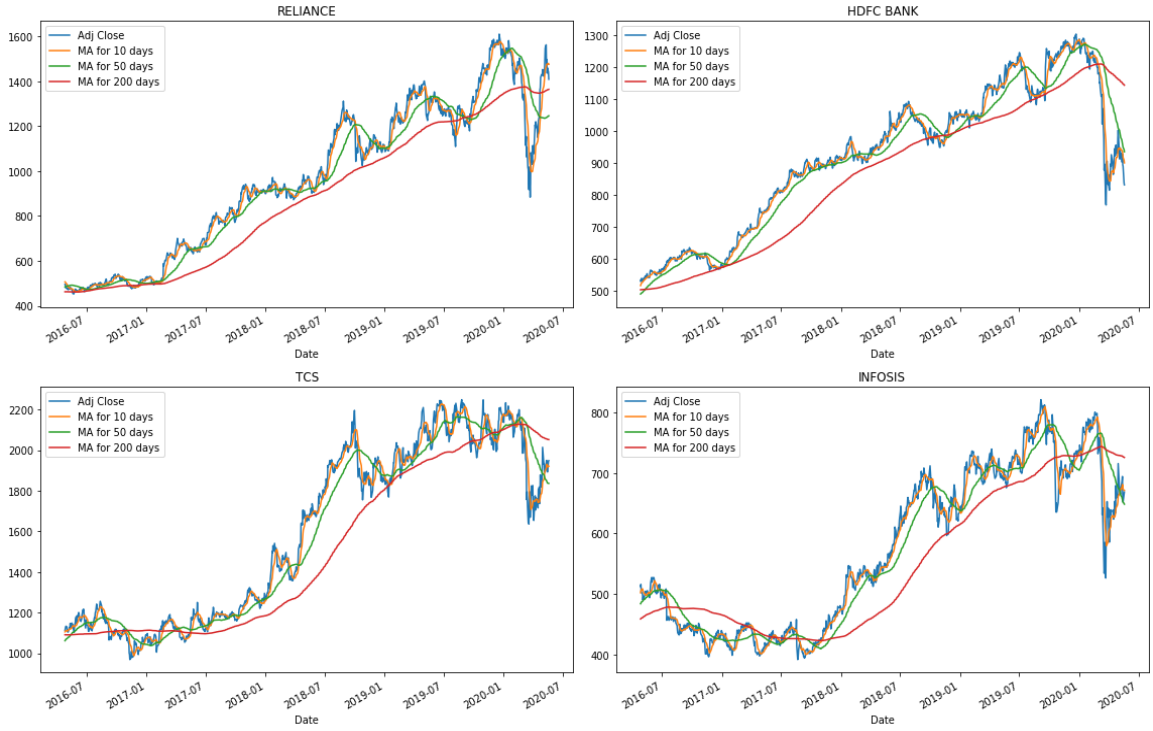


Fig 6.17: Simple Moving Average for 10, 50 and 200 days

Moving averages tells us that this is average or minimum possible return that we are likely to get in the near future. Moving averages change slowly in comparison to the actual price.

9 What was the Annual Sharpe Ratio of the Stocks in the past?

Date	Reliance	HDFC Bank	TCS	Infosis
2011-12-31	-1.290702	-0.392527	-0.124830	-0.590615
2012-12-31	0.576007	2.691502	0.098416	-0.444074
2013-12-31	0.068441	-0.272582	2.776967	1.834415
2014-12-31	-0.207587	1.953733	0.590855	1.130394
2015-12-31	0.353913	0.452002	-0.383376	0.647585
2016-12-31	0.084834	0.356861	-0.238275	-0.484872
2017-12-31	2.624157	3.835127	0.588217	0.148690
2018-12-31	0.637510	0.527094	1.489296	1.083440
2019-12-31	1.063034	0.629196	0.364209	0.258321

Tab 6.4: Annual Sharpe Ratio of last 9 years from 2011 to 2019

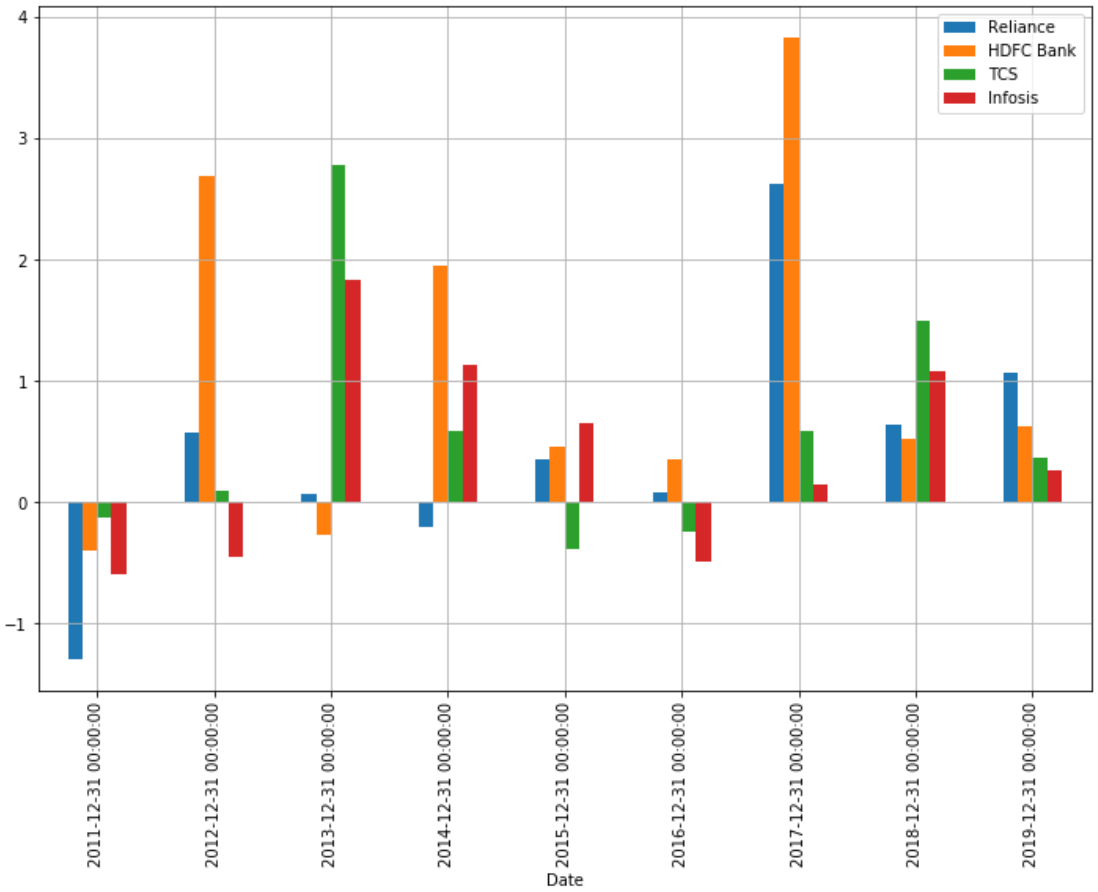


Fig 6.18: Annual Sharpe Ratio Bar Plot of last 9 years from 2011 to 2019

Sharpe ratio also known as Risk Adjusted Return is the measure of excess return of a portfolio of stocks over the risk-free return. It tells us whether the risk taken to earn the return over the risk-free possible return, from an Investment security is worth the taking a risk or not. It helps us in choosing right stocks with minimum risk good amount of return. Sharpe Ratio is a term which is calculated annually, and it changes overtime.

What Sharpe Ratio is considered Good?

ASR< 1 – Bad (Not Worth taking the risk)

1<ASR<2 – Acceptable

2<ASR<3 – Good

ASR>3 – Excellent

Chapter 7

Prediction using models

7.1 AUGMENTED DICKEY FULLER TEST

In AD Fuller test, time series data is tested for the null hypothesis. Null hypothesis states that a unit root is found in a time series data. Alternate hypothesis depends on whether which type of the test is being used on the data, which is usually stationarity or trend-stationarity. For large and complex set of time series models AD Fuller test is used.

The ADF statistic measure which is used for the test is a negative number and more the number is negative, hypothesis will be rejected more strongly which means that there is a unit root with some level of confidence.

```
Results of dickey fuller test
Test Statistics                  -2.036163
p-value                           0.581605
No. of lags used                 21.000000
Number of Observations used    2440.000000
Critical Value (1%)              -3.962485
Critical Value (5%)              -3.412291
Critical Value (10%)             -3.128110
dtype: float64
```

Fig 7.1: Result of ADFuller Test

p-value>0.05 implies data is non stationary.

7.2 SEASONAL DECOMPOSITION

Decomposition of Time series data into Seasonal, Trend and Residual component is known as the seasonal decomposition.

If we consider additive decomposition, then we can write $Y_t = S_t + T_t + R_t$, where Y_t is the data, S_t is the seasonal component, Tt is the trend-cycle component, and R_t is the remainder component, all at period t. Alternatively, a multiplicative decomposition would be written as $Y_t = S_t \times T_t \times R_t$. The additive decomposition is the most appropriate if the magnitude of the seasonal fluctuations, or the variation around the trend-cycle, does not vary with the level of the time series. When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative decomposition is more appropriate. Multiplicative decompositions are common with economic time series. An alternative to using a multiplicative decomposition is to first transform the data until the variation in the series appears to be stable over time, then use an additive decomposition. When a log of transformation has been used, this is equivalent to using a multiplicative decomposition because $Y_t = S_t \times T_t \times R_t$ is equivalent to $\log Y_t = \log S_t + \log T_t + \log R_t$.

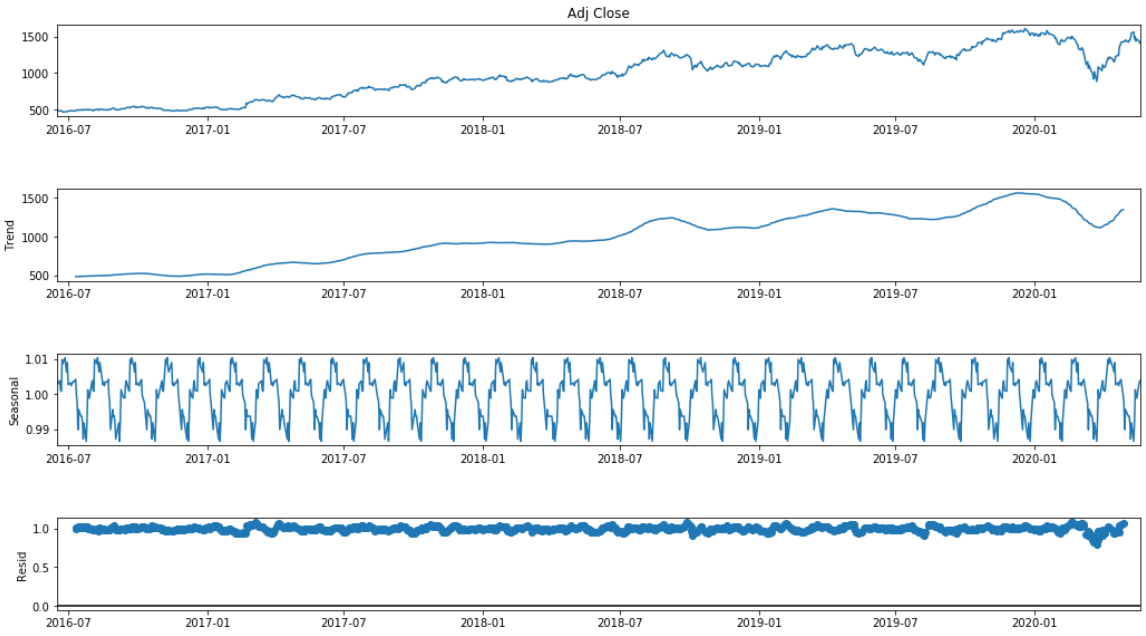


Fig 7.2: Seasonal Decomposition of Reliance Stock into Trend, Seasonal and Residual Components

7.3 FINDING ACF & PACF

On observing the autocorrelation function and partial autocorrelation plots of the differenced series, one can be uncertain in identifying the numbers of AR and/or MA terms needed. The plot of ACF is a bar graph of coefficients of correlation which is plotted between a time series and lags with itself. The PACF plot is a plot between the series and the lags of itself which gives partial correlation coefficients.

Partial correlation between the two variables is the value of correlation between them which cannot be explained by mutual correlations with other variables. For example, if we are regressing variable Y on the variables X_1, X_2, and X_3, the partial correlation between the variables Y and X_3 is the amount of correlation between the variables Y and X_3 which cannot be explained with common correlations with variables X_1 and X_2. This partial correlation can be calculated as the square root of the reduction in variance which is achieved by adding variable X_3 to the regression of variable Y on variables X_1 and X_2.

A partial autocorrelation is the value of correlation between a variable and a lag of itself which cannot be explained by correlations at all the lower order lags. The autocorrelation of a time series Y at lag 1 is the coefficient of correlation between Y_t and Y_{t-1}, which is presumably also the correlation between Y_{t-1} and Y_{t-2}. But if Y_t is correlated with Y_{t-1}, and Y_{t-1} is equally correlated with Y_{t-2}, then we should also expect to find correlation between Y_t and Y_{t-2}. In fact, the amount of correlation we should expect at lag 2 is precisely the square of the lag-1 correlation. Thus, the correlation at lag 1 "propagates" to lag 2 and presumably to higher-order lags. The partial autocorrelation at lag 2 is therefore the difference between the actual correlation at lag 2 and the expected correlation due to the propagation of correlation at lag 1.

Rules to find whether the model is AR and/or MA are:

1. If PACF plot of a differenced series shows a sharp cutoff and/or the lag-1 auto-correlation is positive that is on observing if the series is slightly "underdifferenced" then consider adding an AR term to the model. The lag at which PACF cuts off tells the number of AR terms.

2. If ACF plot of a differenced series shows a sharp cutoff and/or the lag-1 autocorrelation is negative that is, if the series appears slightly "overdifferenced" then consider adding an MA term to the model. The lag at which the ACF cuts off tell the number of MA terms.

(ACF) **Autocorrelation Function**	**(PACF)** **Partial Autocorrelation Function**
Finds the correlation between two values by taking all the past values in time series under consideration (Does not remove the effect of shorter lags autocorrelation and considers them all while estimating longer lags).	Does not take all the past values and instead considers only one past value for finding current one (Removes the effect of shorter lags autocorrelation for estimating longer lags).
There is more than one time lag in values of times series while finding out the correlation between two values.	There is only one time lag between current and one past value.
Uses indirect impacts to the observed value.	Uses direct impact of one past value on the current value.
Does not use coefficient since this type compares all values from the past for finding out the current value.	Uses coefficient since that gives the multiplier effect of one past value to the current value for finding the latter aptly.

Tab 7.1: Difference between ACF and PACF

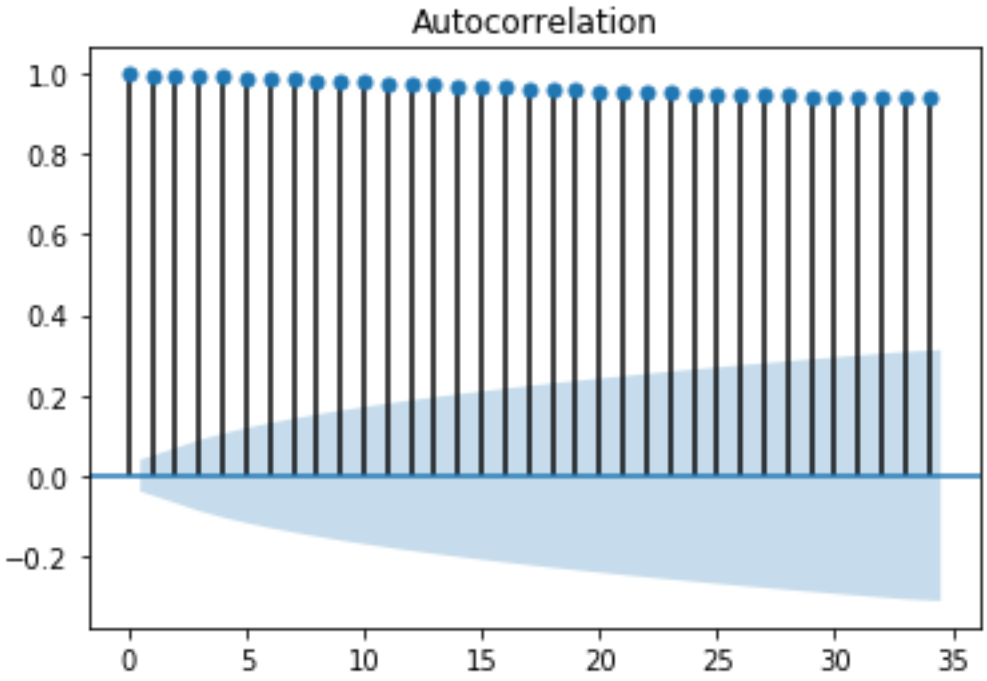


Fig 7.3: ACF Plot of Reliance Stock

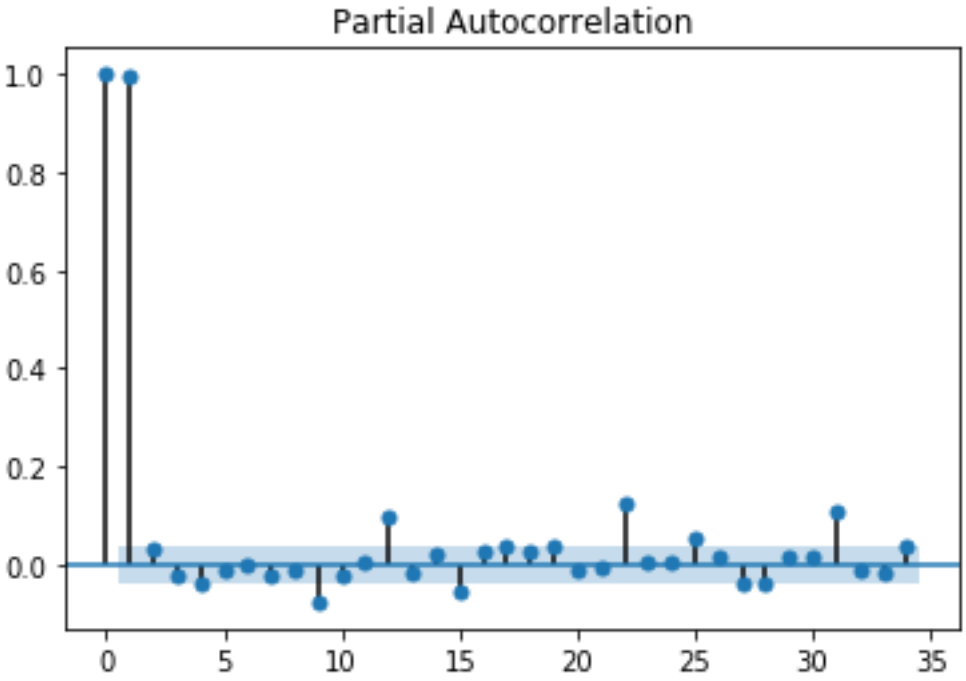


Fig 7.4: PACF Plot of Reliance Stock

7.4 AUTO ARIMA

Auto Arima function does the job of finding best possible values of p, d, q for our model. Model with lowest AIC value is selected and gives us the required values of p, d, q for our ARIMA model.

```
Performing stepwise search to minimize aic
Fit ARIMA(0,1,0)x(0,0,0,0) [intercept=True]; AIC=-11892.502, BIC=-11881.100, Time=0.295 seconds
Fit ARIMA(1,1,0)x(0,0,0,0) [intercept=True]; AIC=-11894.013, BIC=-11876.910, Time=0.192 seconds
Fit ARIMA(0,1,1)x(0,0,0,0) [intercept=True]; AIC=-11894.060, BIC=-11876.956, Time=0.472 seconds
Fit ARIMA(0,1,0)x(0,0,0,0) [intercept=False]; AIC=-11892.525, BIC=-11886.824, Time=0.096 seconds
Fit ARIMA(1,1,1)x(0,0,0,0) [intercept=True]; AIC=-11892.042, BIC=-11869.237, Time=1.141 seconds
Fit ARIMA(0,1,2)x(0,0,0,0) [intercept=True]; AIC=-11892.099, BIC=-11869.294, Time=0.523 seconds
Fit ARIMA(1,1,2)x(0,0,0,0) [intercept=True]; AIC=-11891.298, BIC=-11862.792, Time=0.336 seconds
Total fit time: 3.067 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2212
Model:               SARIMAX(0, 1, 1)   Log Likelihood                5950.030
Date:                Wed, 20 May 2020   AIC                         -11894.060
Time:                        04:01:43   BIC                         -11876.956
Sample:                             0   HQIC                        -11887.811
                                - 2212
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0005      0.000      1.347      0.178      -0.000       0.001
ma.L1          0.0391      0.018      2.209      0.027       0.004       0.074
sigma2         0.0003   5.93e-06     45.360      0.000       0.000       0.000
===================================================================================
Ljung-Box (Q):                       55.32   Jarque-Bera (JB):               278.59
Prob(Q):                              0.05   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                             0.02
Prob(H) (two-sided):                  0.01   Kurtosis:                         4.74
===================================================================================
```

Fig 7.5: Summary of Auto Arima

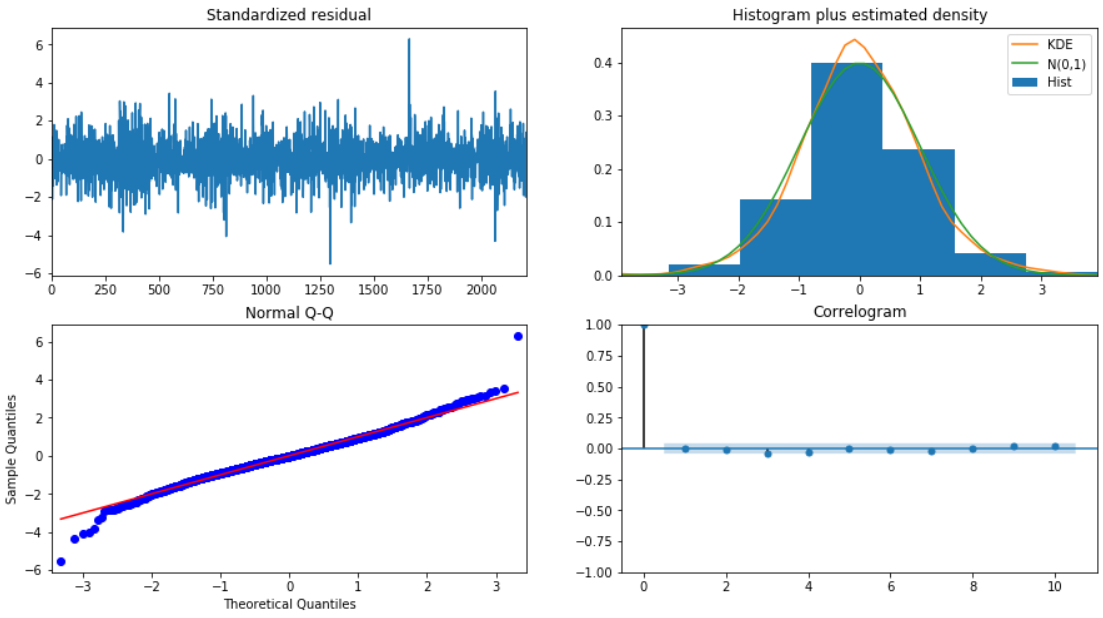


Fig 7.6: Plot Diagnostic of Auto ARIMA

7.5 ARIMA MODEL

ARIMA stands for Auto-Regressive Integrated Moving Average. Here, “Auto-Regressive term” means the lags in a stationary series of a forecasting equation, “moving average term” means the lags in the forecast errors, and “integrated” means the time series data is made stationary by using differencing method on the series. Special cases of a ARIMA model are autoregressive models, random-walk, exponential smoothing models, and random-trend models.

For forecasting a time series generally ARIMA models are used, which by differencing can be made to be “stationary” series, while in conjunction with non-linear transformations like deflating or logging. If statistical measures such as mean, variance and covariance are all constant over the time then the time series data is considered to be stationary. A series which has no trend is known as stationary, therefore variations in the series are around its mean and they have a constant amplitude, and it wiggles in a consistent continuous pattern, that is its short-term random time patterns always look the same in a statistical sense. This condition means that its autocorrelation will remain constant over the time. Such a random variable with such pattern can be considered as a combination of signal and noise, where the signal could be fast or slow pattern of mean reversion or sinusoidal oscillation or rapid alternation in sign, and so it could also have a seasonal component. It can be considered as a “filter” which separates the signal and the noise. The signal produced is then used to derive the forecasts results.

Equation of forecasting using ARIMA for a stationary time series is a regression-type equation where predictors are the lags of the dependent variable and/or lags of the forecast errors. So,

Predicted value of Y variable = a constant and/or a weighted sum of one or more recent values of Y and/or a weighted sum of one or more recent values of the errors.

A nonseasonal ARIMA model is generalized as an "ARIMA (p, d, q)" model, where:

p - Number of autoregressive terms,

d - Number of nonseasonal differences needed for stationarity

q - Number of lagged forecasted errors in prediction equation.

First order Autoregressive model - ARIMA (1,0,0)

Random walk model - ARIMA (0,1,0)

Differenced First order Autoregressive model - ARIMA (1,1,0)

Simple Exponential Smoothing model - ARIMA (0,1,1) without constant.

Simple Exponential Smoothing with some growth - ARIMA (0,1,1) with constant.

Linear Exponential Smoothing model - ARIMA (0,2,1) or (0,2,2) without constant.

Damped-trend Linear Exponential Smoothing model - ARIMA (1,1,2) without constant.

ARIMA Model Results

Dep. Variable:	D.Adj Close	No. Observations:	2211
Model:	ARIMA(0, 1, 1)	Log Likelihood	5950.032
Method:	css-mle	S.D. of innovations	0.016
Date:	Wed, 20 May 2020	AIC	-11894.063
Time:	04:01:57	BIC	-11876.960
Sample:	1	HQIC	-11887.815

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.0005	0.000	1.355	0.175	-0.000	0.001
ma.L1.D.Adj Close	0.0404	0.021	1.892	0.058	-0.001	0.082

Roots

	Real	Imaginary	Modulus	Frequency
MA.1	-24.7801	+0.0000j	24.7801	0.5000

Fig 7.7: ARIMA model forecast results

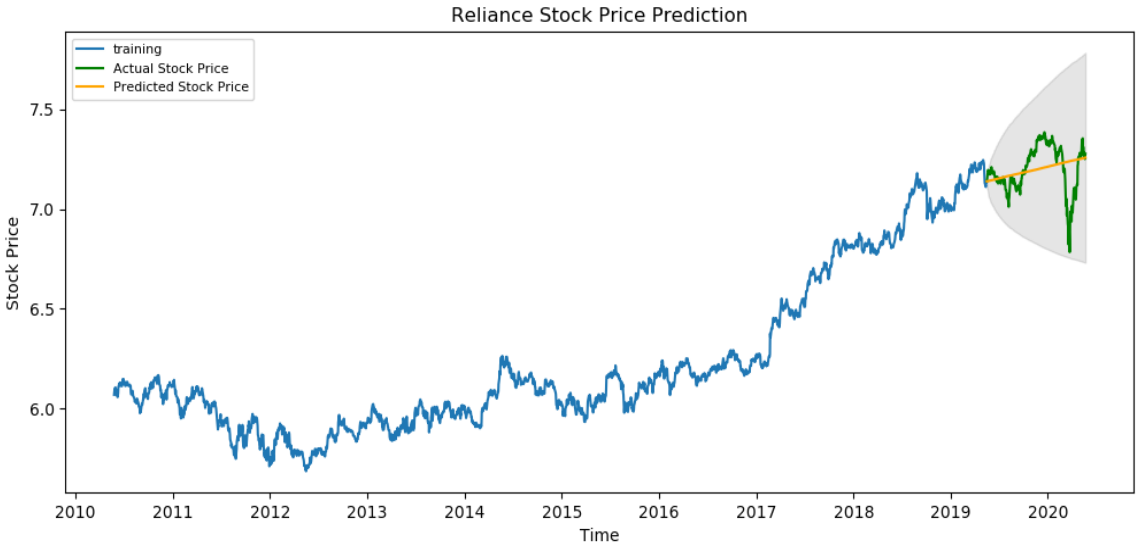


Fig 7.8: ARIMA forecasting plot of future values with 95% confidence

Results:

```
MSE: 0.013511206582547987
MAE: 0.0852704149709506
RMSE: 0.11623771583504205
MAPE: 0.011900728052480233
```

7.6 MONTE CARLO SIMULATION

There are situations when problem comprises of random variables which cannot be predicted easily then probability of different possible outcomes are modeled, this is known as monte carlo simulation. This method is useful in understanding the impact of risk and uncertainty of prediction and forecasting models during analysis. Monte Carlo model can also be used to solve various problems such as in fields of engineering, supply chain, science and finance. This model is also referred as multiple probability simulation model.

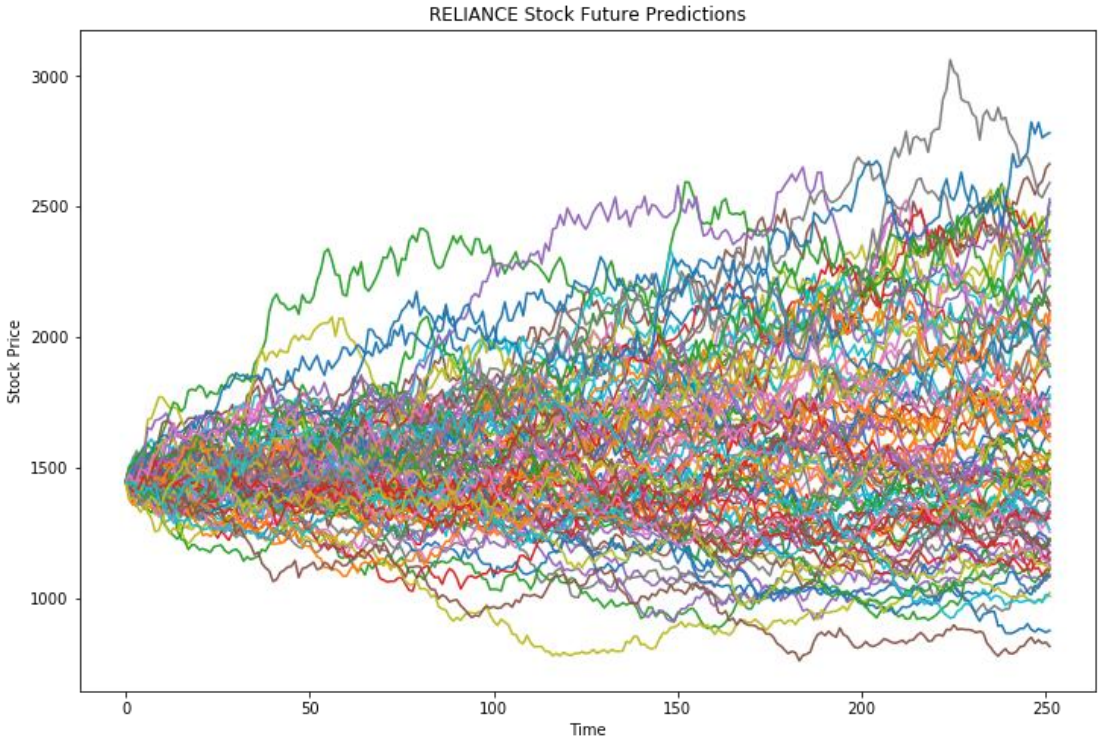


Fig 7.9: Monte Carlo Simulation of Reliance of 1 Business year

7.7 FORECASTING WITH PROPHET PACKAGE

The Blue line in center implies the future trend of average stock prices while the actual value can vary within the blue band. The blue band helps in tracking in what range can be the future stock prices can lie. Black scattered points are the actual stock data over the time period.

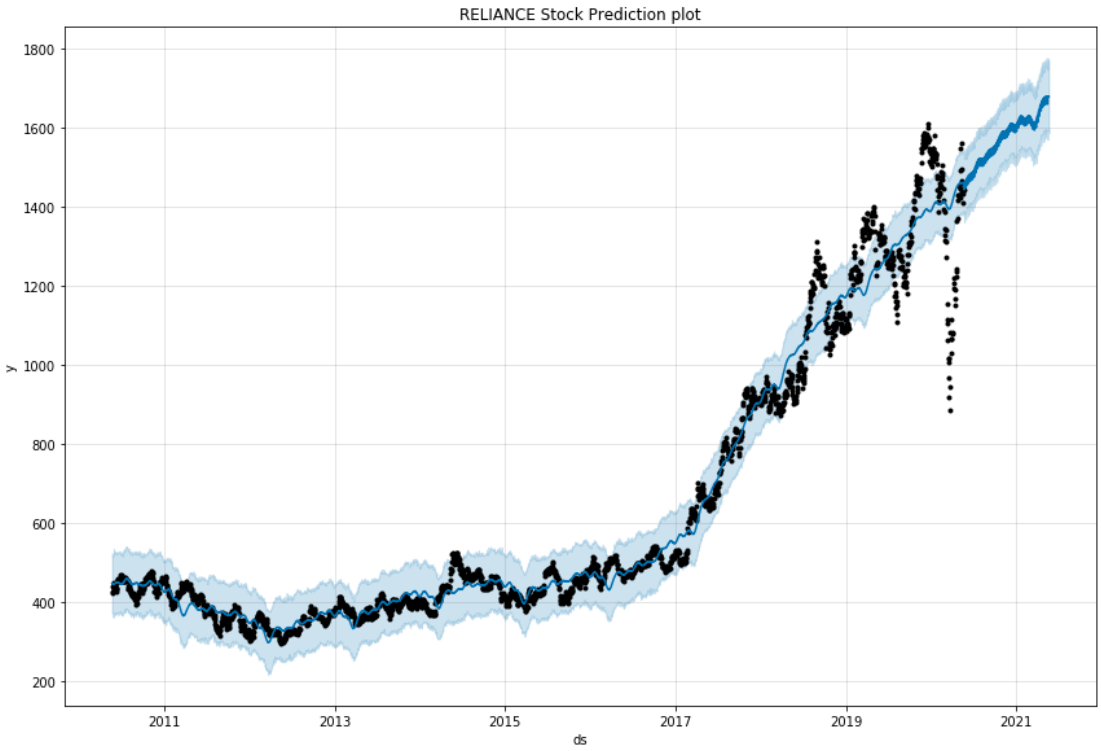


Fig 7.10: Prophet Forecast of Reliance of Next one year

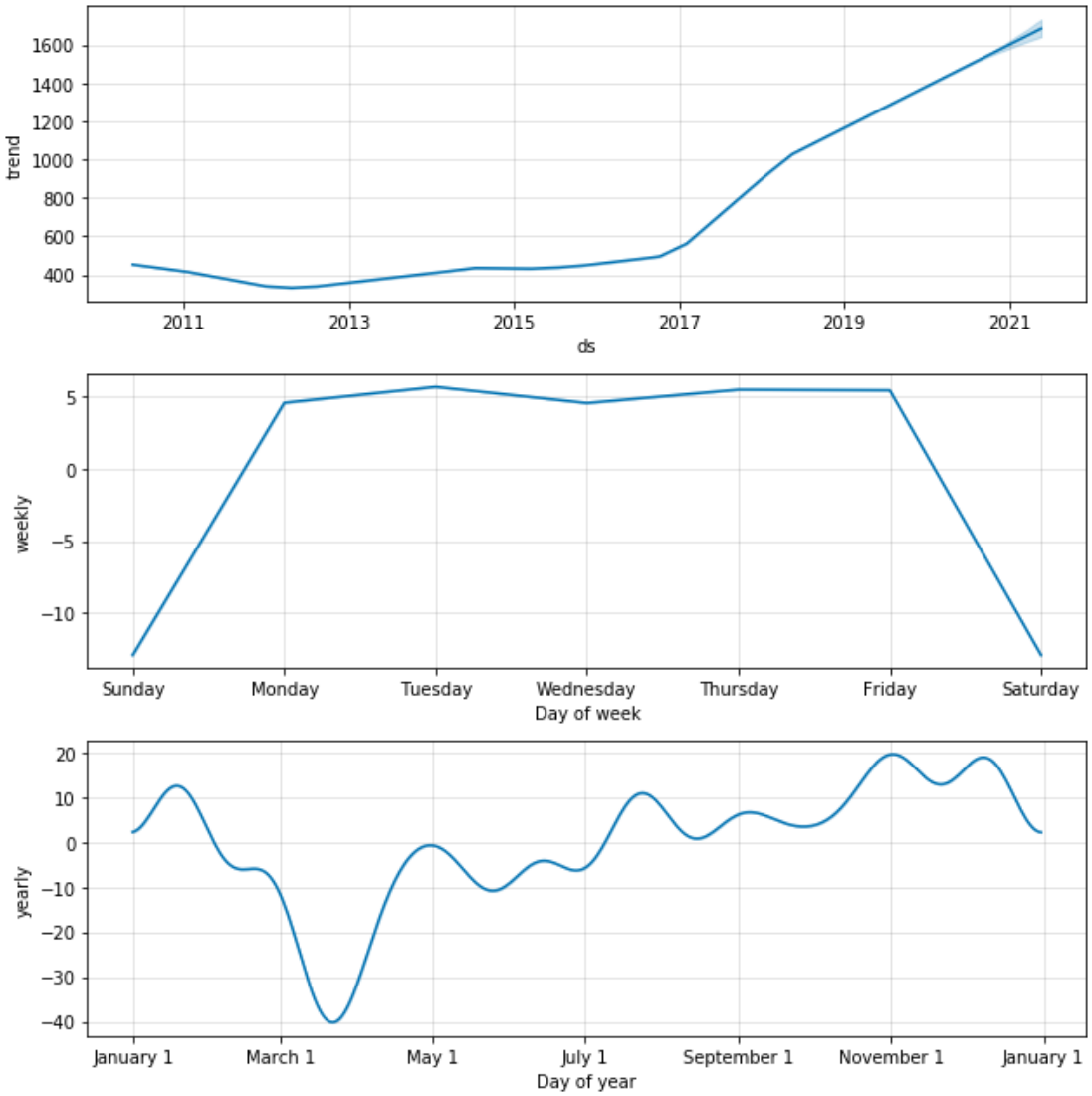


Fig 7.11: Decomposition into daily trend, weekly trend and yearly trend

Using prophet data can be properly distributed into overall trend, weekly trend, and yearly trend. Here we can observe yearly trend have some seasonality.

Chapter 8

Conclusion & future work

8.1 CONCLUSION

In this study, we could analyze and understand the risk involved in the stock price and are well accounted by statistical terms such as CAGR, Sharpe Ratio volatility and the cumulative return. This study has proved to be successful in comparison analysis between stocks and get stock with lesser risk and good return.

We also found out that ARIMA model performed really well in predicting the stock prices. Mont Carlo also gave us a overview of possible future trends. Prophet library by Facebook worked well gave us fulfilling results of future prices.

8.2 FUTURE WORK

In this study we worked with linear model for the forecasting of the future price. But in there no evidence that stock data was linear, this motivates to work on forecasting using the non-linear data in the near future.

Maybe with some statistical measurements we can develop a model which can help in choosing the right stock.

REFERENCES

[1] Banerjee, D. (2014, January). Forecasting of Indian stock market using time-series ARIMA model. In *2014 2nd International Conference on Business and Information Management (ICBIM)* (pp. 131-135). IEEE.

[2] Viswam, N., & Reddy, G. S. (2018). Stock market prediction using time series analysis.

[3] Angadi, M. C., & Kulkarni, A. P. (2015). Time Series Data Analysis for Stock Market Prediction using Data Mining Techniques with R. International Journal of Advanced Research in Computer Science, 6(6).

[4] Devi, B. U., Sundar, D., & Alli, P. (2013). An effective time series analysis for stock trend prediction using ARIMA model for nifty midcap-50. *International Journal of Data Mining & Knowledge Management Process*, *3*(1), 65.

[5] Varghese, A., Tarhen, H., Shaikh, A., Banik, P., & Ramadasi, A. (2016). Stock Market Prediction Using Time Series. International Journal on Recent and Innovation Trends in Computing and Communication, 4(5), 427-430.

[6] Sharma, A., Modak, S., & Sridhar, E. (2019). Data Visualization and Stock Market and Prediction.

[7] Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017, September). Stock price prediction using LSTM, RNN and CNN-sliding window model. In 2017 international conference on advances in computing, communications and informatics (icacci) (pp. 1643-1647). IEEE.

[8] Mondal, P., Shit, L., & Goswami, S. (2014). Study of effectiveness of time series modeling (ARIMA) in forecasting stock prices. International Journal of Computer Science, Engineering and Applications, 4(2), 13.

[9] Wang, J., & Wang, J. (2015). Forecasting stock market indexes using principle component analysis and stochastic time effective neural networks. Neurocomputing, 156, 68-78.

[10] CA Rachana Phadke Ranade. (2020). Retrieved 22 May 2020, from https://www.youtube.com/user/rachanaphadke

[11] (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=edvg4eHi_Mw&list=PLdYtcvznoqMfEIoIrOzN5ow79u8QSIdzU&index=15&t=5307s

[12] Edureka Lecture. (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=e8Yw4alG16Q&list=PLdYtcvznoqMfEIoIrOzN5ow79u8QSIdzU&index=19&t=577s

[13] Tutorial video. (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=QIUxPv5PJOY&list=PLdYtcvznoqMfEIoIrOzN5ow79u8QSIdzU&index=26&t=0s

[14] Intellipat tutorial. (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=QtYOI-9R1vo&list=PLdYtcvznoqMfEIoIrOzN5ow79u8QSIdzU&index=24&t=0s

[15] Tutorial. (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=AF8zgxLukg4&list=LLKkSctU6i5sUIIX2LHyomjQ&index=3&t=0s

[16] Krish Naik. (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=2XGSIlgUBDI

[17] Time series analysis. (2020). Retrieved 22 May 2020, from https://www.youtube.com/watch?v=MmC4b7gPY0Q&t=561s

[18] Predict Stock Prices Using Python & Machine Learning. (2020). Retrieved 22 May 2020, from https://medium.com/@randerson112358/predict-stock-prices-using-python-machine-learning-53aa024da20a

[19] codes), S. (2020). Stock Price Prediction Using Machine Learning | Deep Learning. Retrieved 22 May 2020, from https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/

[20] Simulating future stock prices using Monte Carlo methods in Python. (2020). Retrieved 22 May 2020, from https://www.interviewqs.com/blog/intro_monte_carlo

[21] How to Use the Sharpe Ratio to Analyze Portfolio Risk and Return. (2020). Retrieved 22 May 2020, from https://www.investopedia.com/terms/s/sharperatio.asp

[22] Time Series Analysis - Statistics Solutions. (2020). Retrieved 22 May 2020, from https://www.statisticssolutions.com/time-series-analysis/

[23] The Complete Guide to Time Series Analysis and Forecasting. (2020). Retrieved 22 May 2020, from https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775

[24] Brownlee, J. (2020). What Is Time Series Forecasting?. Retrieved 22 May 2020, from https://machinelearningmastery.com/time-series-forecasting/

APPENDICES

Complete Code Snippets:

jupyter Final Capstone Last Checkpoint: 12 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run Markdown

Importing Libraries

In [202]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import pandas_datareader as web
from datetime import datetime
from scipy.stats import norm
import warnings
import math

%matplotlib inline
warnings.filterwarnings('ignore')
```

Importing Datasets using pandas datareader

In [2]:
```
end = datetime.now()
start = datetime(end.year-10, end.month, end.day)

rel = web.DataReader('RELIANCE.NS','yahoo',start,end)
hdfc = web.DataReader('HDFCBANK.NS','yahoo',start,end)
tcs = web.DataReader('TCS.NS','yahoo',start,end)
infy = web.DataReader('INFY.NS','yahoo',start,end)
```

In [3]:
```
rel.head()
```

Out[3]:

Date	High	Low	Open	Close	Volume	Adj Close
2010-05-20	506.500000	496.875000	500.500000	499.975006	8492908.0	429.394226
2010-05-21	499.924988	488.049988	494.950012	497.774994	8762694.0	427.504791
2010-05-24	524.474976	508.225006	509.000000	511.325012	11304034.0	439.141998
2010-05-25	512.500000	491.375000	512.500000	492.825012	11541592.0	423.253540
2010-05-26	505.000000	495.500000	497.450012	504.024994	6275326.0	432.872467

In [4]:
```
hdfc.head()
```

Out[4]:

Date	High	Low	Open	Close	Volume	Adj Close
2010-05-20	187.639999	183.630005	187.399994	185.095001	3389000.0	143.784668
2010-05-21	185.000000	180.110001	182.000000	182.654999	5498620.0	141.889206
2010-05-24	186.750000	181.104996	183.520004	182.455002	4810910.0	141.733871
2010-05-25	182.000000	178.500000	181.179993	180.375000	5534350.0	140.118088
2010-05-26	185.324997	181.229996	181.279999	182.774994	7001390.0	141.982437

In [5]:
```
tcs.head()
```

Out[5]:

Date	High	Low	Open	Close	Volume	Adj Close
2010-05-20	366.149994	358.075012	364.5	365.000000	2398092.0	273.470551
2010-05-21	362.950012	342.625000	355.0	359.350006	2675092.0	269.237335
2010-05-24	365.000000	356.350006	362.5	358.799988	2885428.0	268.825287
2010-05-25	357.000000	345.774994	354.0	349.774994	2297366.0	262.063446
2010-05-26	372.399994	352.049988	354.0	369.000000	3689464.0	276.467468

In [6]:
```
infy.head()
```

Out[6]:

Date	High	Low	Open	Close	Volume	Adj Close
2010-05-20	329.319000	323.312012	326.500000	325.093994	6508872.0	128.791458
2010-05-21	326.000000	319.794006	321.000000	322.718994	7127200.0	127.850540
2010-05-24	328.750000	322.406006	326.375000	325.049988	7186104.0	128.774033
2010-05-25	323.355988	313.763000	323.355988	316.674988	8838128.0	125.456123
2010-05-26	329.987000	319.531006	320.631012	327.631012	7868856.0	136.250336

In [7]:
```
rel.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2462 entries, 2010-05-20 to 2020-05-19
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       2462 non-null   float64
 1   Low        2462 non-null   float64
 2   Open       2462 non-null   float64
 3   Close      2462 non-null   float64
 4   Volume     2462 non-null   float64
 5   Adj Close  2462 non-null   float64
dtypes: float64(6)
memory usage: 134.6 KB
```

In [8]:
```
hdfc.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2462 entries, 2010-05-20 to 2020-05-19
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       2462 non-null   float64
 1   Low        2462 non-null   float64
 2   Open       2462 non-null   float64
 3   Close      2462 non-null   float64
 4   Volume     2462 non-null   float64
 5   Adj Close  2462 non-null   float64
dtypes: float64(6)
memory usage: 134.6 KB
```

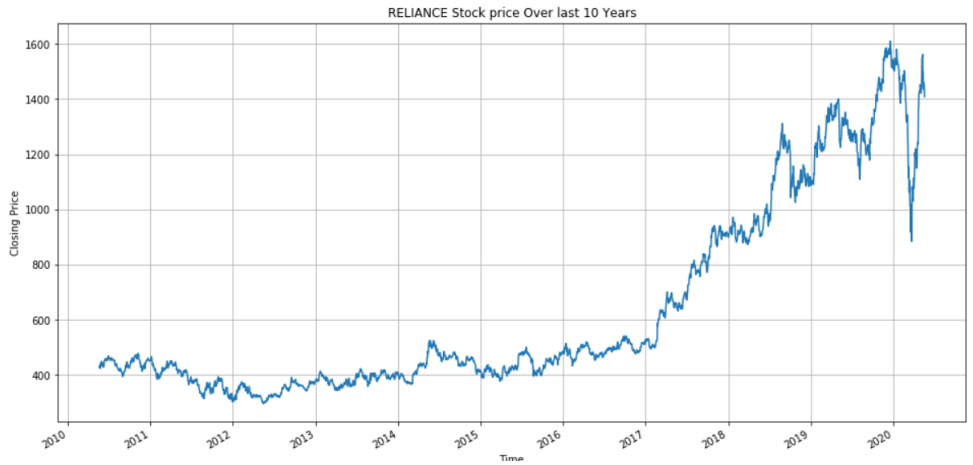
In [9]:
```
tcs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2462 entries, 2010-05-20 to 2020-05-19
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       2462 non-null   float64
 1   Low        2462 non-null   float64
 2   Open       2462 non-null   float64
 3   Close      2462 non-null   float64
 4   Volume     2462 non-null   float64
 5   Adj Close  2462 non-null   float64
dtypes: float64(6)
memory usage: 134.6 KB
```

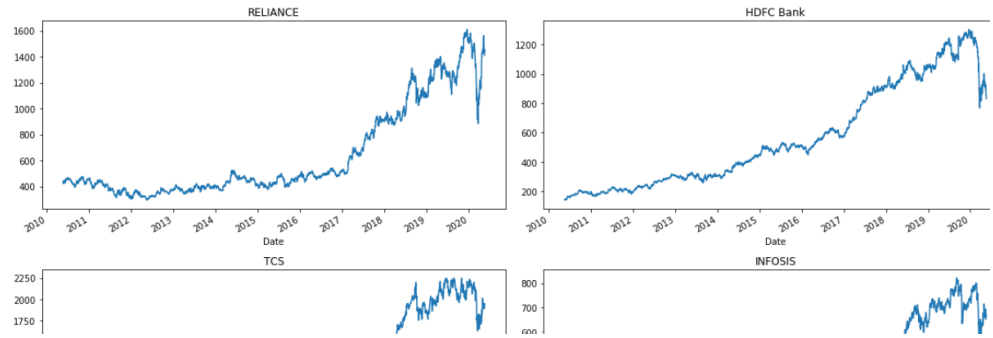
In [10]:
```
infy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 2462 entries, 2010-05-20 to 2020-05-19
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   High       2462 non-null   float64
 1   Low        2462 non-null   float64
 2   Open       2462 non-null   float64
 3   Close      2462 non-null   float64
 4   Volume     2462 non-null   float64
 5   Adj Close  2462 non-null   float64
dtypes: float64(6)
memory usage: 134.6 KB
```

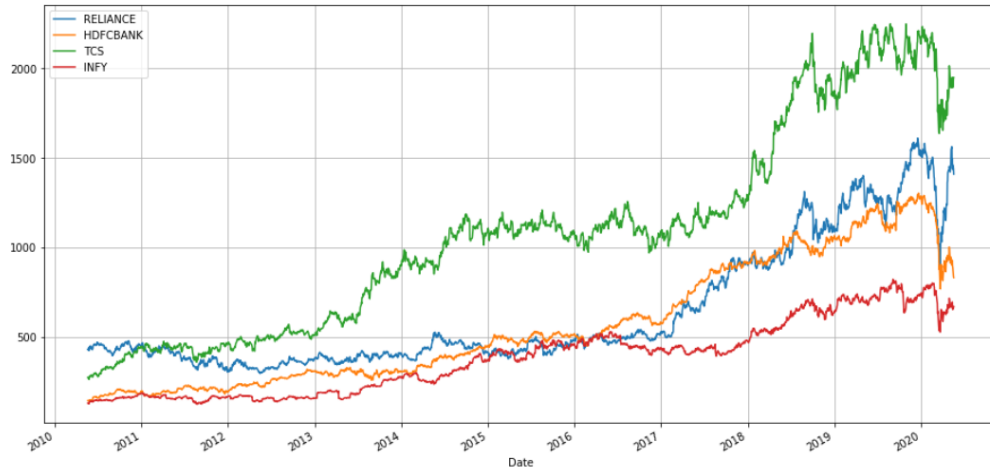
In [12]:
```
#Closing Price Trend of Reliance over the last 10 years
rel['Adj Close'].plot(figsize=(16,8),title='RELIANCE Stock price Over last 10 Years',grid=True)
plt.xlabel('Time')
plt.ylabel('Closing Price')
plt.show()
```



In [13]:
```
#Closing Price Trend of Stocks over the last 10 years
fig,axes = plt.subplots(nrows=2,ncols=2)
fig.set_figheight(8)
fig.set_figwidth(16)

rel['Adj Close'].plot(ax=axes[0,0])
axes[0,0].set_title('RELIANCE')
hdfc['Adj Close'].plot(ax=axes[0,1])
axes[0,1].set_title('HDFC Bank')
tcs['Adj Close'].plot(ax=axes[1,0])
axes[1,0].set_title('TCS')
infy['Adj Close'].plot(ax=axes[1,1])
axes[1,1].set_title('INFOSIS')

fig.tight_layout()
```

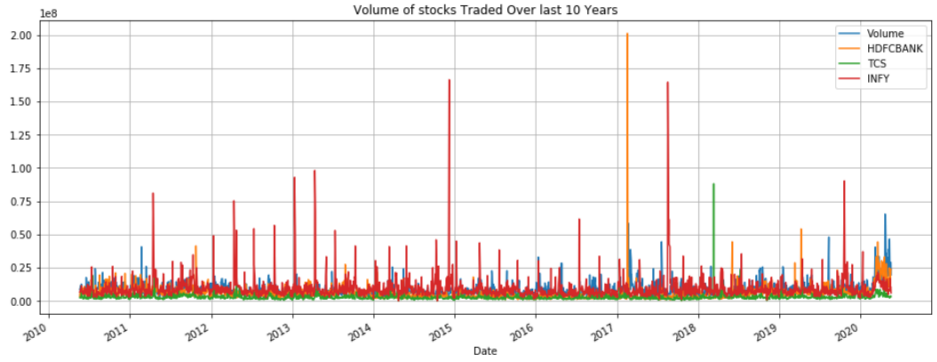


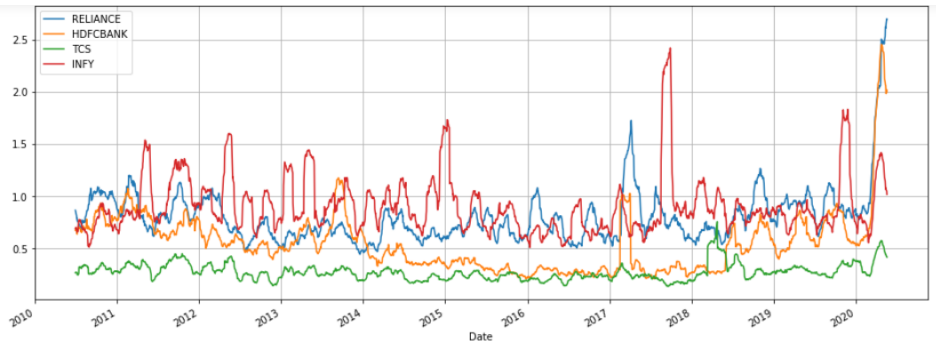
In [14]:
```
#Closing Price trend Comparision analysis over last 10 years
rel['Adj Close'].plot(figsize=(16,8), label='RELIANCE',grid=True)
hdfc['Adj Close'].plot(label='HDFCBANK',grid=True)
tcs['Adj Close'].plot(label='TCS',grid=True)
infy['Adj Close'].plot(label='INFY',grid=True)
plt.legend();
```



10 years ago, a unit share price TCS stock was less Reliance, But over the years TCS Stocks gained more value than the Reliance Stocks. Whereas observe from the plot that HDFC Bank and INFOSIS had similar price for a unit share of the stock but over the years HDFCBANK gave more profit to shareholders as its Stocks performed well in comparision to INFOSIS stocks.

In [187]:
```
rel['Volume'].plot(figsize=(16,6),title='Volume of stocks Traded Over last 10 Years',grid=True)
hdfc['Volume'].plot(label='HDFCBANK',grid=True)
tcs['Volume'].plot(label='TCS',grid=True)
infy['Volume'].plot(label='INFY',grid=True)
plt.legend();
plt.show()
rel['Volume'].rolling(30).mean().plot(figsize=(16,6),label = 'RELIANCE',title='Volume of stocks Traded Over last 10 Years(Rolling
hdfc['Volume'].rolling(30).mean().plot(label='HDFCBANK',grid=True)
tcs['Volume'].rolling(30).mean().plot(label='TCS',grid=True)
infy['Volume'].rolling(30).mean().plot(label='INFY',grid=True)
plt.legend();
```





Calculating Daily Returns

Daily return is the change in price with respect to the previous day.

Daily Return = (Price of share on Day 2 - Price of share on Day 1) / Price of share on Day 1

Daily return can be either positive or negative. It is positive when a stock gains value with respect to previous day and is negative when stock looses value.

In [17]:
```
#Daily Returns
rel['Daily Returns']=rel['Adj Close'].pct_change()
hdfc['Daily Returns']=hdfc['Adj Close'].pct_change()
tcs['Daily Returns']=tcs['Adj Close'].pct_change()
infy['Daily Returns']=infy['Adj Close'].pct_change()
```

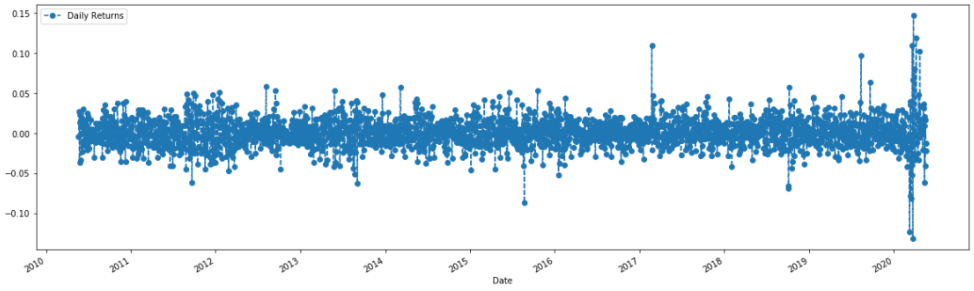
In [18]:
```
rel.tail()
```

In [18]:
```
rel.tail()
```

Out[18]:

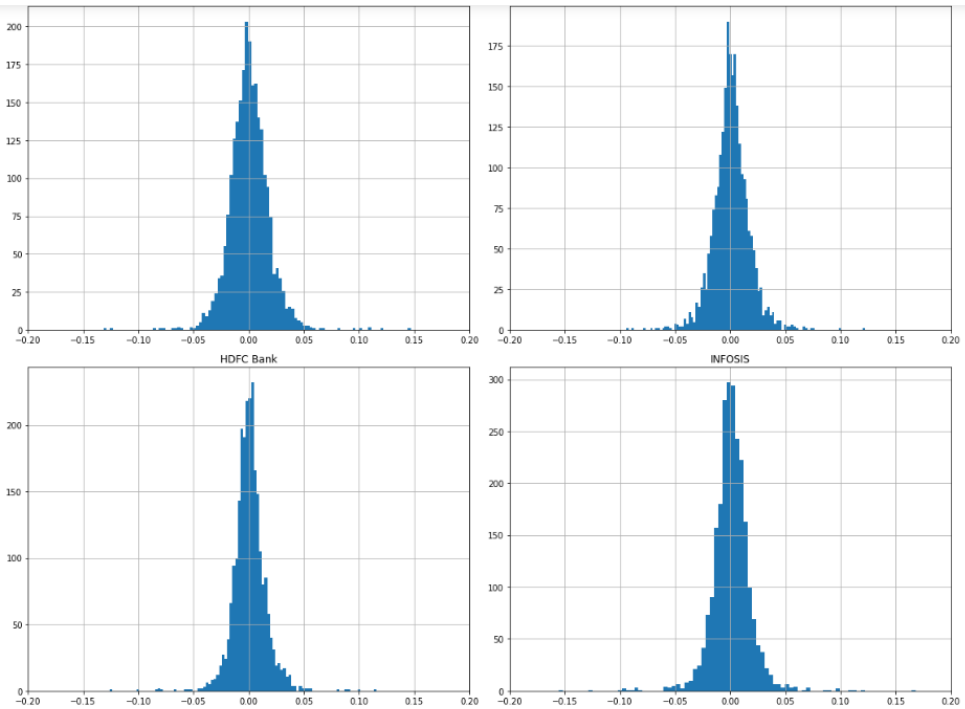
Date	High	Low	Open	Close	Volume	Adj Close	Daily Returns
2020-05-13	1527.000000	1454.000000	1527.0	1496.449951	30657492.0	1496.449951	0.021217
2020-05-14	1496.699951	1430.050049	1469.0	1435.949951	22736607.0	1435.949951	-0.040429
2020-05-15	1466.699951	1415.099976	1444.0	1459.400024	28683258.0	1459.400024	0.016331
2020-05-18	1482.000000	1428.000000	1470.0	1440.750000	28980385.0	1440.750000	-0.012779
2020-05-19	1461.699951	1403.250000	1457.0	1408.900024	19519940.0	1408.900024	-0.022107

In [188]:
```
rel['Daily Returns'].plot(figsize=(20,6), legend=True, linestyle='--', marker='o')
plt.show()
rel['Daily Returns'].iloc[2000:].plot(figsize=(20,6), legend=True, linestyle='--', marker='o');
```



Lets Plot histogram of daily returns of stocks.

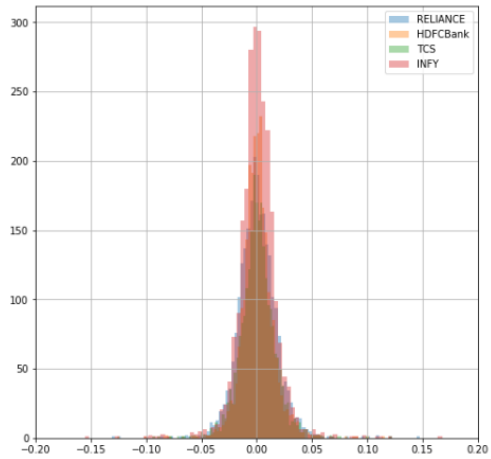
In [20]:
```
#Histogram
fig,axes = plt.subplots(nrows=2,ncols=2)
fig.set_figheight(12)
fig.set_figwidth(16)

rel['Daily Returns'].hist(ax=axes[0,0],bins=100)
axes[0,0].set_title('RELIANCE')
axes[0,0].set_xlim(-0.2,0.2)
tcs['Daily Returns'].hist(ax=axes[0,1],bins=100)
axes[0,1].set_title('TCS')
axes[0,1].set_xlim(-0.2,0.2)
hdfc['Daily Returns'].hist(ax=axes[1,0],bins=100)
axes[1,0].set_title('HDFC Bank')
axes[1,0].set_xlim(-0.2,0.2)
infy['Daily Returns'].hist(ax=axes[1,1],bins=100)
axes[1,1].set_title('INFOSIS')
axes[1,1].set_xlim(-0.2,0.2)
fig.show()

fig.tight_layout()
```



We can observe the shape of the histogram is different for each of the stock. Thickness of the histogram tell us about the volatility of the stock as the daily return values do not fall close to zero.

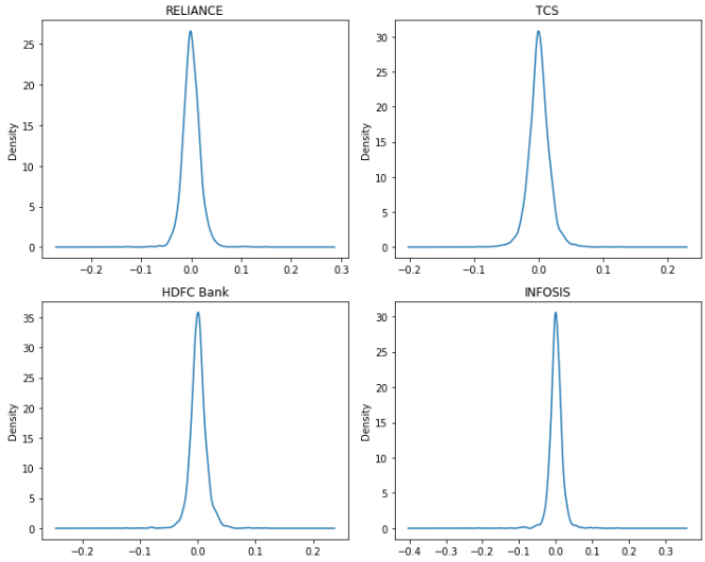
- Higher the Volatility higher is the risk of investing.
- Higher volatility also denotes higher returns or the higher losses.

In [21]:
```
rel['Daily Returns'].hist(bins=100,label='RELIANCE',figsize=(8,8),alpha=0.4)
hdfc['Daily Returns'].hist(bins=100,label='HDFCBank',figsize=(8,8),alpha=0.4)
tcs['Daily Returns'].hist(bins=100,label='TCS',alpha=0.4)
infy['Daily Returns'].hist(bins=100,label='INFY',alpha=0.4)
plt.xlim(-0.2,0.2)
plt.legend();
```



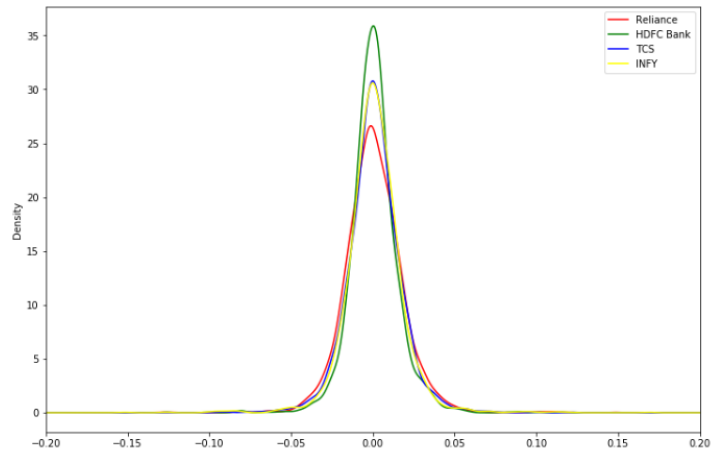
This plot does not give much clearer picture. Lets try implementing it through Kernel density plot.

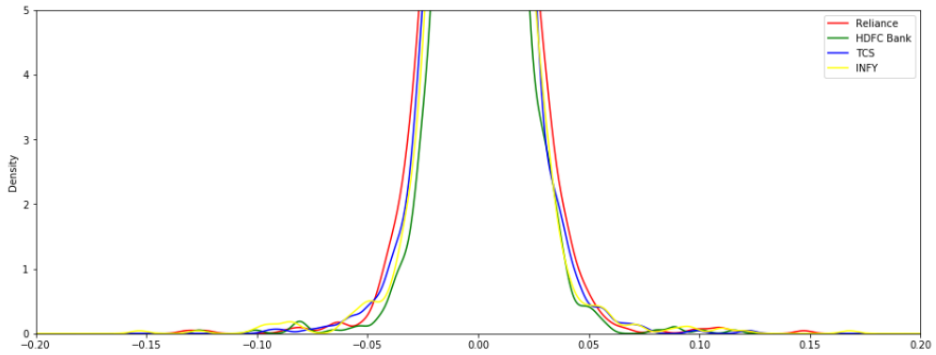
In [22]:
```
#Kernel Density plot to check
fig,axes = plt.subplots(nrows=2,ncols=2)
fig.set_figheight(8)
fig.set_figwidth(10)

rel['Daily Returns'].plot(kind='kde',ax=axes[0,0])
axes[0,0].set_title('RELIANCE')
#axes[0,0].set_xlim(-0.2,0.2)
tcs['Daily Returns'].plot(kind='kde',ax=axes[0,1])
axes[0,1].set_title('TCS')
#axes[0,1].set_xlim(-0.2,0.2)
hdfc['Daily Returns'].plot(kind='kde',ax=axes[1,0])
axes[1,0].set_title('HDFC Bank')
#axes[1,0].set_xlim(-0.2,0.2)
infy['Daily Returns'].plot(kind='kde',ax=axes[1,1])
axes[1,1].set_title('INFOSIS')
#axes[1,1].set_xlim(-0.2,0.2)
fig.show()

fig.tight_layout()
```



There is a observable difference between these plot and it tells Reliance and TCS have more volatility.

In [23]:
```
rel['Daily Returns'].plot(kind='kde', color='red', label='Reliance',figsize=(12,8))
hdfc['Daily Returns'].plot(kind='kde', color='green', label='HDFC Bank')
tcs['Daily Returns'].plot(kind='kde', color='blue', label='TCS')
infy['Daily Returns'].plot(kind='kde', color='yellow', label='INFY')
plt.xlim(-0.2,0.2)
plt.legend();
plt.show()
#Zoomed graph
rel['Daily Returns'].plot(kind='kde', color='red', label='Reliance',figsize=(16,6))
hdfc['Daily Returns'].plot(kind='kde', color='green', label='HDFC Bank')
tcs['Daily Returns'].plot(kind='kde', color='blue', label='TCS')
infy['Daily Returns'].plot(kind='kde', color='yellow', label='INFY')
plt.xlim(-0.2,0.2)
plt.ylim(0,5)
plt.legend();
```





This gives us a clear comparable observation that Reliance stock is most volatile and HDFC Bank has least volatility.

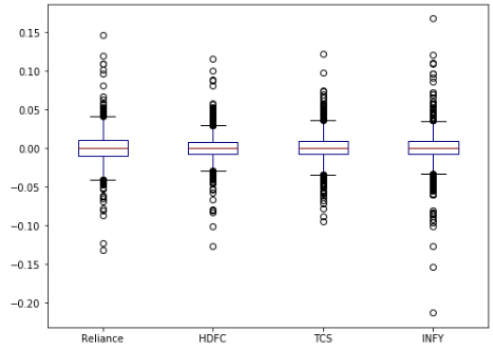
In [24]:
```
Stocks_Daily_Ret = pd.concat([rel['Daily Returns'],hdfc['Daily Returns'],tcs['Daily Returns'], infy['Daily Returns']],axis=1)
Stocks_Daily_Ret.columns = ['Reliance','HDFC','TCS','INFY']
Stocks_Daily_Ret.dropna()
```

Out[24]:

Date	Reliance	HDFC	TCS	INFY
2010-05-21	-0.004400	-0.013183	-0.015480	-0.007306
2010-05-24	0.027221	-0.001095	-0.001530	0.007223
2010-05-25	-0.036181	-0.011400	-0.025153	-0.025765
2010-05-26	0.022726	0.013306	0.054964	0.086040
2010-05-27	0.014087	0.037669	0.005217	0.008510
...	...	...	...	...
2020-05-13	0.021217	0.028950	0.000077	0.009452
2020-05-14	-0.040429	-0.036598	-0.024261	-0.051862
2020-05-15	0.016331	-0.006210	-0.004968	-0.008889
2020-05-18	-0.012779	-0.057986	0.027841	0.017783
2020-05-19	-0.022107	-0.007171	0.001568	0.007079

2461 rows × 4 columns

In [25]:
```
Stocks_Daily_Ret.plot(kind='box',figsize=(8,6),colormap='jet');
```



Box plot gives us clear much clearer analysis, Infosis stocks have two outliers, neglecting that reliance is the most volatile and HDFC is the least

Calculating Cumulative Returns of the Stocks

Cumulative returns is the total return earned in an investment irrespective of the time. It is the calculated by dividing current price of the stock by the price stock were bought.

Cumulative return cannot be negative.

Cumulative Return = Current price of a unit share / Price at which the share was bought

In [26]:
```
#Cumulative Returns
rel['Cumulative Returns']=rel['Adj Close']/rel['Adj Close'][0]
hdfc['Cumulative Returns']=hdfc['Adj Close']/hdfc['Adj Close'][0]
tcs['Cumulative Returns']=tcs['Adj Close']/tcs['Adj Close'][0]
infy['Cumulative Returns']=infy['Adj Close']/infy['Adj Close'][0]
```

In [27]:
```
Stocks_Cumulative_Ret = pd.concat([rel['Cumulative Returns'],hdfc['Cumulative Returns'],tcs['Cumulative Returns'], infy['Cumulati
Stocks_Cumulative_Ret.columns = ['Reliance','HDFC','TCS','INFY']
Stocks_Cumulative_Ret
```

Out[27]:

Date	Reliance	HDFC	TCS	INFY
2010-05-20	1.000000	1.000000	1.000000	1.000000
2010-05-21	0.995600	0.986817	0.984520	0.992694
2010-05-24	1.022701	0.985737	0.983014	0.999865
2010-05-25	0.985699	0.974500	0.958288	0.974103
2010-05-26	1.008100	0.987466	1.010959	1.057914
...	...	...	...	...
2020-05-13	3.485026	6.451662	7.129287	5.389721
2020-05-14	3.344130	6.215545	6.956325	5.110199
2020-05-15	3.398742	6.176945	6.921769	5.064777
2020-05-18	3.355308	5.818771	7.114477	5.154845
2020-05-19	3.281134	5.777042	7.125630	5.191338

2462 rows × 4 columns

In the last 10 years, TCS gave us the maximum return, amount invested in TCS stock have surged upto 7 times in last 10 years. while Reliance gave us minimum return where prices surge upto 3.3 times approx.

Compounded Annual Growth Rate(CAGR) over last 10 Years

In [28]:
```
Stock_Close = pd.concat([rel['Adj Close'],hdfc['Adj Close'],tcs['Adj Close'],infy['Adj Close']],axis=1)
Stock_Close.columns = ['Reliance','HDFC Bank','TCS','Infosis']
Stock_Close
```

Out[28]:

Date	Reliance	HDFC Bank	TCS	Infosis
2010-05-20	429.394226	143.784668	273.470551	128.791458
2010-05-21	427.504791	141.889206	269.237335	127.850540
2010-05-24	439.141998	141.733871	268.825287	128.774033
2010-05-25	423.253540	140.118088	262.063446	125.456123
2010-05-26	432.872467	141.982437	276.467468	136.250336
...	...	...	...	...
2020-05-13	1496.449951	927.650024	1949.650024	694.150024
2020-05-14	1435.949951	893.700012	1902.349976	658.150024
2020-05-15	1459.400024	888.150024	1892.900024	652.299988
2020-05-18	1440.750000	836.650024	1945.599976	663.900024
2020-05-19	1408.900024	830.650024	1948.650024	668.599976

2462 rows × 4 columns

In [29]:
```
nifty = web.DataReader('^NSEI','yahoo', start, end)
nifty.head()
```

Out[29]:

Date	High	Low	Open	Close	Volume	Adj Close
2010-05-20	4980.250000	4924.299805	4924.299805	4947.600098	0.0	4947.600098
2010-05-21	4946.700195	4842.299805	4946.700195	4931.149902	0.0	4931.149902
2010-05-24	5029.549805	4923.450195	4944.299805	4943.950195	0.0	4943.950195
2010-05-25	4946.600098	4786.450195	4945.299805	4806.750000	0.0	4806.750000
2010-05-26	4925.450195	4807.299805	4807.299805	4917.399902	0.0	4917.399902

In [30]:
```
nifty['Cumulative Return'] = nifty['Adj Close'][-1]/nifty['Adj Close'][0]
nifty.tail()
```

Out[30]:

Date	High	Low	Open	Close	Volume	Adj Close	Cumulative Return
2020-05-13	9584.500000	9351.099609	9584.200195	9383.549805	846400.0	9383.549805	1.794628
2020-05-14	9281.099609	9119.750000	9213.950195	9142.750000	602600.0	9142.750000	1.794628
2020-05-15	9182.400391	9050.000000	9182.400391	9136.849609	575900.0	9136.849609	1.794628
2020-05-18	9158.299805	8806.750000	9158.299805	8823.250000	773000.0	8823.250000	1.794628
2020-05-19	9030.349609	8855.299805	8961.700195	8879.099609	0.0	8879.099609	1.794628

In [190]:
```
Stock_Close=Stock_Close.join(nifty['Adj Close'],on='Date')
Stock_Close.columns = ['Reliance','HDFC Bank','TCS','Infosis','Nifty']
Stock_Close
```

Out[190]:

Date	Reliance	HDFC Bank	TCS	Infosis	Nifty
2010-05-20	429.394226	143.784668	273.470551	128.791458	4947.600098
2010-05-21	427.504791	141.889206	269.237335	127.850540	4931.149902
2010-05-24	439.141998	141.733871	268.825287	128.774033	4943.950195
2010-05-25	423.253540	140.118088	262.063446	125.456123	4806.750000
2010-05-26	432.872467	141.982437	276.467468	136.250336	4917.399902
...	...	...	...	...	...
2020-05-13	1496.449951	927.650024	1949.650024	694.150024	9383.549805
2020-05-14	1435.949951	893.700012	1902.349976	658.150024	9142.750000
2020-05-15	1459.400024	888.150024	1892.900024	652.299988	9136.849609
2020-05-18	1440.750000	836.650024	1945.599976	663.900024	8823.250000
2020-05-19	1408.900024	830.650024	1948.650024	668.599976	8879.099609

2462 rows × 5 columns

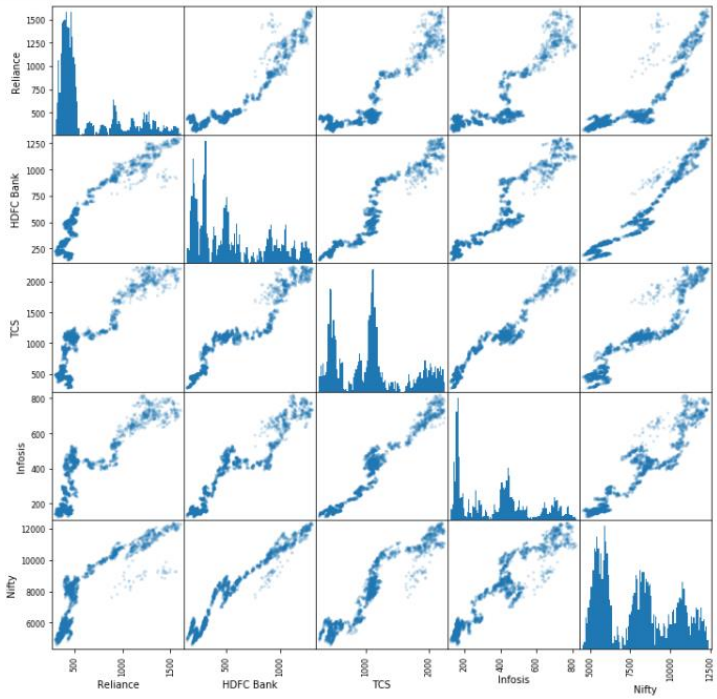
In [32]:
```
CAGR = ((Stock_Close.iloc[-1]/Stock_Close.iloc[0])**(1/10) - 1)*100
CAGR
```

Out[32]:
```
Reliance     12.616596
HDFC Bank    19.170991
TCS          21.697688
Infosis      17.903835
Nifty         6.022352
dtype: float64
```

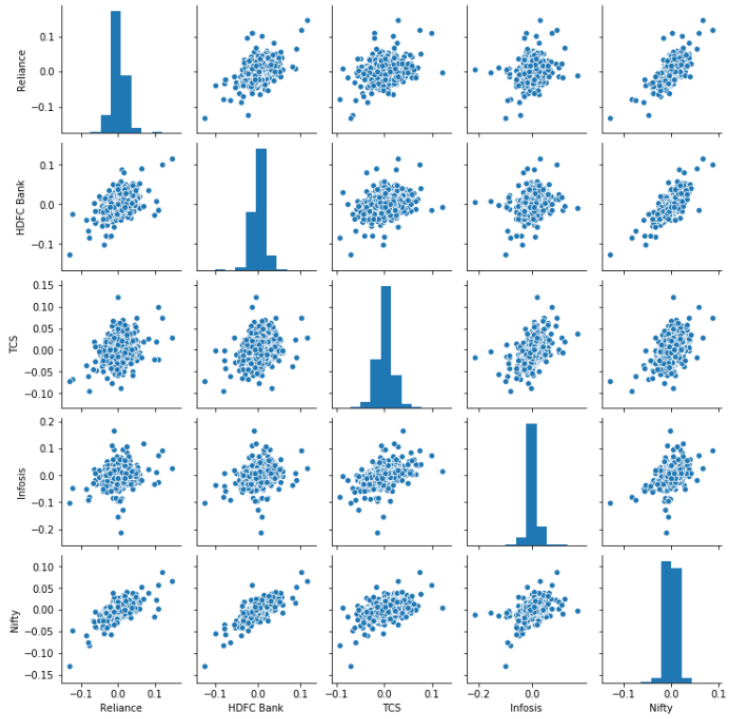
Lets find out Correlation between the stocks

In [33]:
```
from pandas.plotting import scatter_matrix
```

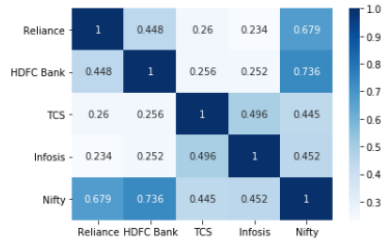
In [34]:
```
scatter_matrix(Stock_Close,figsize=(12,12),alpha=0.2,hist_kwds={'bins':100});
```



In [191]:
```
sns.pairplot(Stock_Close.pct_change().dropna(),size=2);
```

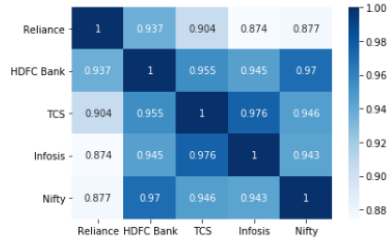


In [36]:
```
#Using Heatmap
sns.heatmap(Stock_Close.pct_change(1).corr(),annot=True,fmt=".3g",cmap='Blues');
```



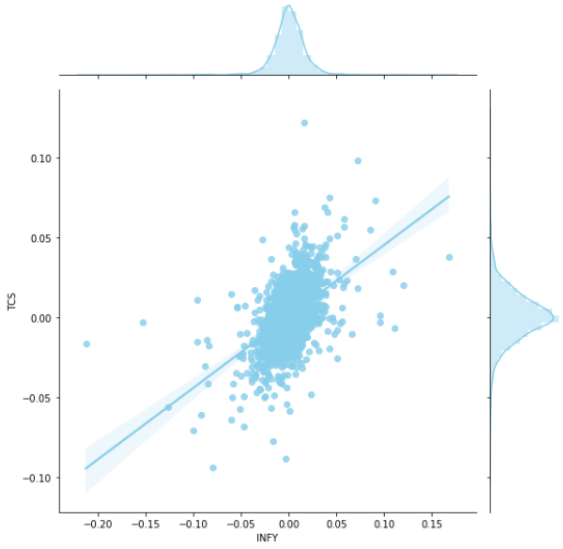
Plot shows that TCS and Infosis are very likely correlated to each other and yes we can consider that as they are tech companies. Correlation here is very weak. HDFC and reliance have good correlation with the market index.

In [37]:
```
#Correlation between closing prices of the stocks using heat map
sns.heatmap(Stock_Close.corr(),annot=True,fmt=".3g",cmap='Blues');
```



Market Index is closely related to all the four stocks. Moreover we observe here is that TCS and INFOSIS have wonderful correlation between them as both of them are TECH companies.

In [38]:
```
sns.jointplot('INFY','TCS',Stocks_Daily_Ret, kind='reg', size=8, color='skyblue');
```

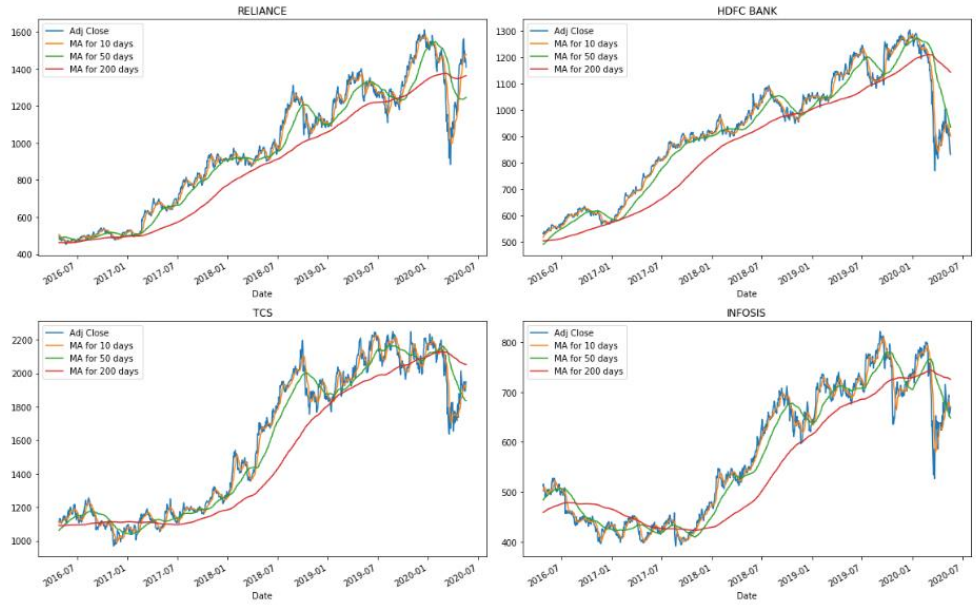


Calculating Simple Moving Average of the Stocks

Why do we calculate simple moving average? Moving averages tells us that this is average or minimum possible return that we are likely to get in the near future. Moving averages change slowly in comparison to the actual price.

In [39]:
```
#Calculating Moving Averages
rel['MA for 10 days'] = rel['Adj Close'].rolling(10).mean()
rel['MA for 50 days'] = rel['Adj Close'].rolling(50).mean()
rel['MA for 200 days'] = rel['Adj Close'].rolling(200).mean()

hdfc['MA for 10 days'] = hdfc['Adj Close'].rolling(10).mean()
hdfc['MA for 50 days'] = hdfc['Adj Close'].rolling(50).mean()
hdfc['MA for 200 days'] = hdfc['Adj Close'].rolling(200).mean()

tcs['MA for 10 days'] = tcs['Adj Close'].rolling(10).mean()
tcs['MA for 50 days'] = tcs['Adj Close'].rolling(50).mean()
tcs['MA for 200 days'] = tcs['Adj Close'].rolling(200).mean()

infy['MA for 10 days'] = infy['Adj Close'].rolling(10).mean()
infy['MA for 50 days'] = infy['Adj Close'].rolling(50).mean()
infy['MA for 200 days'] = infy['Adj Close'].rolling(200).mean()
```

In [40]:
```
fig, axes = plt.subplots(nrows=2, ncols=2)
fig.set_figheight(10)
fig.set_figwidth(16)

rel[['Adj Close', 'MA for 10 days', 'MA for 50 days', 'MA for 200 days']][len(rel)-1000:].plot(ax=axes[0,0])
axes[0,0].set_title('RELIANCE')

hdfc[['Adj Close', 'MA for 10 days', 'MA for 50 days', 'MA for 200 days']][len(rel)-1000:].plot(ax=axes[0,1])
axes[0,1].set_title('HDFC BANK')

tcs[['Adj Close', 'MA for 10 days', 'MA for 50 days', 'MA for 200 days']][len(rel)-1000:].plot(ax=axes[1,0])
axes[1,0].set_title('TCS')

infy[['Adj Close', 'MA for 10 days', 'MA for 50 days', 'MA for 200 days']][len(rel)-1000:].plot(ax=axes[1,1])
axes[1,1].set_title('INFOSIS')

fig.tight_layout()
```



Calculating Sharpe Ratio

Sharpe ratio also known as Risk Adjusted Return is the measure of excess return of a portfolio of stocks over the risk free return. It tells us whether the risk taken to earn the return over the risk free possible return, from a Investment security is worth the taking a risk or not.

It helps us in choosing right stocks with minimum risk good amount of return. Sharpe Ratio is a term which is calculated annually, and it changes overtime.

Sharpe ratio = (Annual Return - Risk Free Return) / Average Annual StDev.

In [41]:
```
#Risk Free Return that we get from other form investments such as Govt. bonds, fixed deposits, banks etc...
risk_free_ret = 7     # Approx. 7 perc as per India
```

In [42]:
```
Stock_Close.drop(['Nifty'],axis=1,inplace=True)
```

In [43]:
```
def last_day(find_val):
    return find_val[-1]

lastdayofyear = Stock_Close.resample('A').apply(last_day)[1:10]
lastdayofyear
```

Out[43]:

Date	Reliance	HDFC Bank	TCS	Infosis
2011-12-31	302.673889	190.174362	455.559967	163.997940
2012-12-31	375.752014	307.147858	506.872894	156.448349
2013-12-31	409.672668	306.454132	904.217163	271.933228
2014-12-31	415.310120	445.441376	1098.123291	367.722137
2015-12-31	483.958435	514.223938	1083.945679	459.878082
2016-12-31	527.132874	582.657654	1090.948364	439.675659
2017-12-31	910.363464	916.567261	1295.471191	480.711365
2018-12-31	1115.074951	1051.955933	1847.181519	635.008240
2019-12-31	1514.050049	1272.099976	2141.158203	731.150024

In [44]:
```
def first_day(find_val):
    return find_val[0]

firstdayofyear = Stock_Close.resample('A').apply(first_day)[1:10]
firstdayofyear
```

Out[44]:

Date	Reliance	HDFC Bank	TCS	Infosis
2011-12-31	453.291138	198.352493	442.056122	184.670242
2012-12-31	308.614227	190.174362	462.605560	166.445450
2013-12-31	376.266724	309.818359	510.222900	155.783890
2014-12-31	406.743805	306.132019	896.407471	270.556396
2015-12-31	413.795502	445.652008	1092.680054	368.066681
2016-12-31	484.316193	517.360168	1074.845093	459.815765
2017-12-31	528.447693	578.261780	1089.011597	435.542542
2018-12-31	899.194580	907.805054	1268.806396	476.790222
2019-12-31	1114.826294	1065.045288	1856.695312	640.886658

In [45]:
```
#Annual Return in Stock price over each year
ann_ret = (lastdayofyear/firstdayofyear - 1) * 100
ann_ret
```

Out[45]:

Date	Reliance	HDFC Bank	TCS	Infosis
2011-12-31	-33.227486	-4.123029	3.054781	-11.194171
2012-12-31	21.754599	61.508551	9.569131	-6.006233
2013-12-31	8.878262	-1.085871	77.220027	74.557991
2014-12-31	2.106071	45.506301	22.502693	35.913304
2015-12-31	16.955944	15.386878	-0.799353	24.944231
2016-12-31	8.840646	12.621282	1.498195	-4.380038
2017-12-31	72.271253	58.503863	18.958439	10.370703
2018-12-31	24.008193	15.879057	45.584190	33.183990
2019-12-31	35.810400	19.440928	15.320925	14.084139

In [46]:
```
#Annual Standard Deviation of Stock over each year
ann_stdev = math.sqrt(252) * Stock_Close.pct_change(1).resample('A').std()[1:10] * 100
ann_stdev
```

Out[46]:

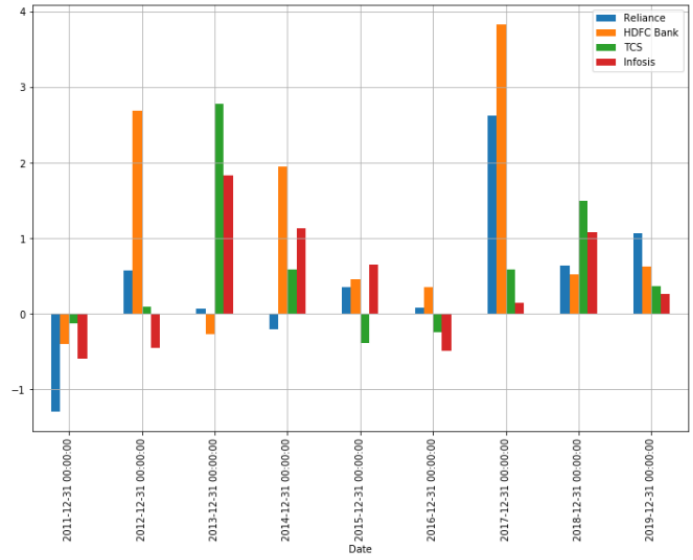
Date	Reliance	HDFC Bank	TCS	Infosis
2011-12-31	31.167137	28.337011	31.604627	30.805491
2012-12-31	25.615307	20.252097	26.104814	29.288437
2013-12-31	27.443574	29.663946	25.286590	36.828087
2014-12-31	23.575278	19.709092	26.237749	25.578076
2015-12-31	28.131053	18.554939	20.343863	27.709452
2016-12-31	21.697092	15.752042	23.090122	23.470196
2017-12-31	24.873231	13.429505	20.329980	22.669295
2018-12-31	26.679088	16.845300	25.907678	24.167460
2019-12-31	27.102041	19.772746	22.846581	27.423787

In [47]:
```
#Annual Sharpe Ratio of Last 9 Years
ASR =(ann_ret - risk_free_ret)/ann_stdev
ASR
```

Out[47]:

Date	Reliance	HDFC Bank	TCS	Infosis
2011-12-31	-1.290702	-0.392527	-0.124830	-0.590615
2012-12-31	0.576007	2.691502	0.098416	-0.444074
2013-12-31	0.068441	-0.272582	2.776967	1.834415
2014-12-31	-0.207587	1.953733	0.590855	1.130394
2015-12-31	0.353913	0.452002	-0.383376	0.647585
2016-12-31	0.084834	0.356861	-0.238275	-0.484872
2017-12-31	2.624157	3.835127	0.588217	0.148690
2018-12-31	0.637510	0.527094	1.489296	1.083440
2019-12-31	1.063034	0.629196	0.364209	0.258321

In [48]:
```
ASR.plot(kind = 'bar',figsize=(12,8),grid=True);
```



Augmented Dickey Fuller test

AD Fuller test is a test of Stationarity of a time series data.

In [49]:
```
from statsmodels.tsa.stattools import adfuller
```

In [50]:
```
def test_stationarity(timeseries):


    print("Results of dickey fuller test")
    adft = adfuller(timeseries,regression='ct',autolag='AIC')
    output = pd.Series(adft[0:4],index=['Test Statistics','p-value','No. of lags used','Number of Observations used'])
    for key,values in adft[4].items():
        output['Critical Value (%s)'%key] =  values
    print(output)

test_stationarity(rel['Adj Close'])
```

```
Results of dickey fuller test
Test Statistics                  -2.036163
p-value                           0.581605
No. of lags used                 21.000000
Number of Observations used    2440.000000
Critical Value (1%)              -3.962485
Critical Value (5%)              -3.412291
Critical Value (10%)             -3.128110
dtype: float64
```

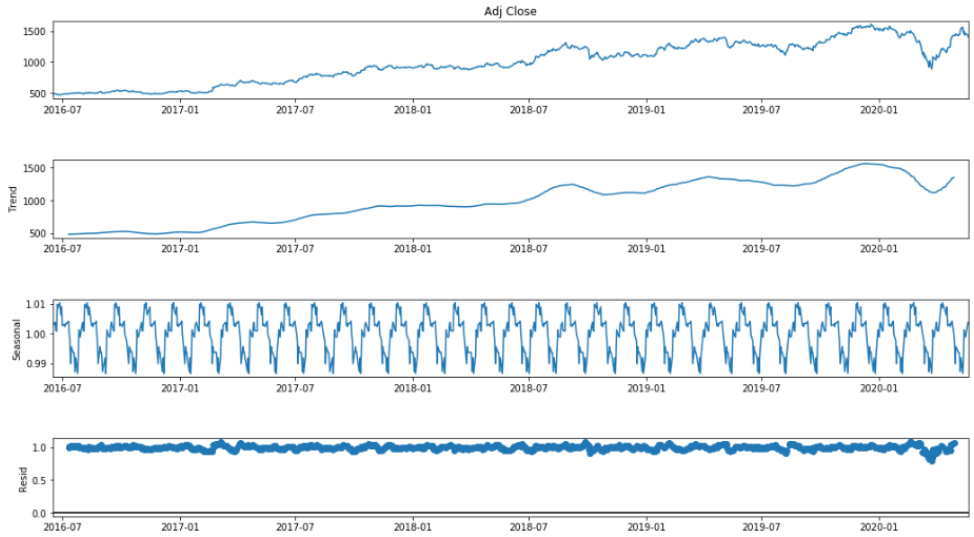
p-value>0.05 implies data is not stationary

Decompostion of Time Series Data into Trend, Seasonality and Residual Components

In [51]:
```
from statsmodels.tsa.seasonal import seasonal_decompose
```

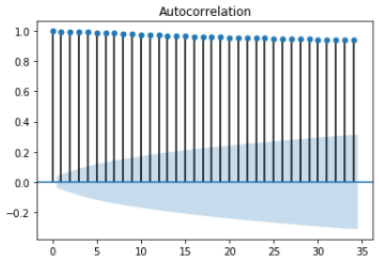
In [52]:
```
result = seasonal_decompose(rel['Adj Close'][1500:], model='multiplicative', freq = 30)
fig = plt.figure()
fig = result.plot()
fig.set_size_inches(16, 9)
```

```
<Figure size 432x288 with 0 Axes>
```

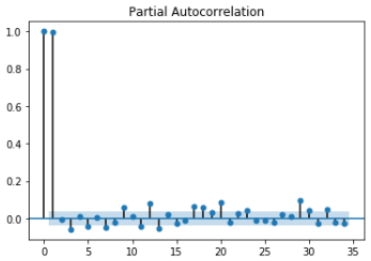


Calculating ACF & PACF

In [53]:
```
from statsmodels.graphics.tsaplots import plot_acf
fig=plot_acf(rel['Adj Close'])
```



In [54]:
```
from statsmodels.graphics.tsaplots import plot_pacf
fig=plot_pacf(rel['Adj Close'])
```



Auto Regressive Integrated Moving Average model

In [127]:
```
#splitting data into training and testing dataset
rel_log = np.log(rel['Adj Close'])
train_data, test_data = rel_log[3:int(len(rel_log)*0.9)], rel_log[int(len(rel_log)*0.9):]
plt.figure(figsize=(16,8))
plt.grid(True)
plt.xlabel('Dates')
plt.ylabel('Closing Prices')
plt.plot(rel_log, 'green', label='Train Data')
plt.plot(test_data, 'blue', label='Test Data')
plt.legend();
```

We will use auto arima method to calculate p, d, q values for the arima model. Auto arima will give us values of p, d, q for the best fit of the arima model.

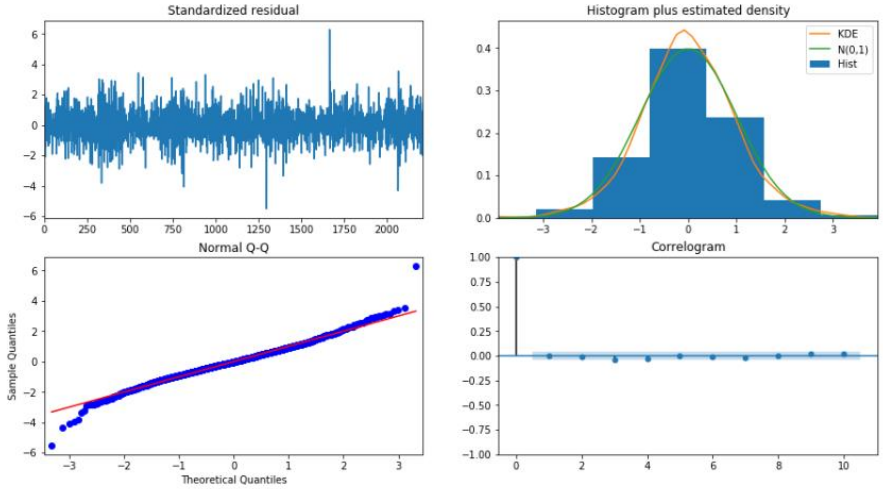
In [128]:
```
from statsmodels.tsa.arima_model import ARIMA
from pmdarima.arima import auto_arima
```

In [129]:
```
model_autoARIMA = auto_arima(train_data, start_p=0, start_q=0, test='adf', max_p=3, max_q=3, m=1, d=None, seasonal=False,
                             start_P=0, D=0, trace=True, error_action='ignore', suppress_warnings=True, stepwise=True)
print(model_autoARIMA.summary())
```

```
Performing stepwise search to minimize aic
Fit ARIMA(0,1,0)x(0,0,0,0) [intercept=True]; AIC=-11892.502, BIC=-11881.100, Time=0.295 seconds
Fit ARIMA(1,1,0)x(0,0,0,0) [intercept=True]; AIC=-11894.013, BIC=-11876.910, Time=0.192 seconds
Fit ARIMA(0,1,1)x(0,0,0,0) [intercept=True]; AIC=-11894.060, BIC=-11876.956, Time=0.472 seconds
Fit ARIMA(0,1,0)x(0,0,0,0) [intercept=False]; AIC=-11892.525, BIC=-11886.824, Time=0.096 seconds
Fit ARIMA(1,1,1)x(0,0,0,0) [intercept=True]; AIC=-11892.042, BIC=-11869.237, Time=1.141 seconds
Fit ARIMA(0,1,2)x(0,0,0,0) [intercept=True]; AIC=-11892.099, BIC=-11869.294, Time=0.523 seconds
Fit ARIMA(1,1,2)x(0,0,0,0) [intercept=True]; AIC=-11891.298, BIC=-11862.792, Time=0.336 seconds
Total fit time: 3.067 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 2212
Model:               SARIMAX(0, 1, 1)   Log Likelihood                5950.030
Date:                Wed, 20 May 2020   AIC                         -11894.060
Time:                        04:01:43   BIC                         -11876.956
Sample:                             0   HQIC                        -11887.811
                                - 2212
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0005      0.000      1.347      0.178      -0.000       0.001
ma.L1          0.0391      0.018      2.209      0.027       0.004       0.074
sigma2         0.0003   5.93e-06     45.360      0.000       0.000       0.000
===================================================================================
Ljung-Box (Q):                       55.32   Jarque-Bera (JB):               278.59
Prob(Q):                              0.05   Prob(JB):                         0.00
Heteroskedasticity (H):               0.82   Skew:                             0.02
Prob(H) (two-sided):                  0.01   Kurtosis:                         4.74
===================================================================================

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

So p=0, d=1, q=1. p implies Auto Regressive so when p=0 implies AR is not used for calculation. d implies Integrated so when d=1, it implies we are going to calculate differencing with lag=1. Differencing is done in order to make data stationary. q implies Moving Average so when q=1 it implies we are going to calculate MA(1).

In [130]:
```
model_autoARIMA.plot_diagnostics(figsize=(15,8))
plt.show()
```

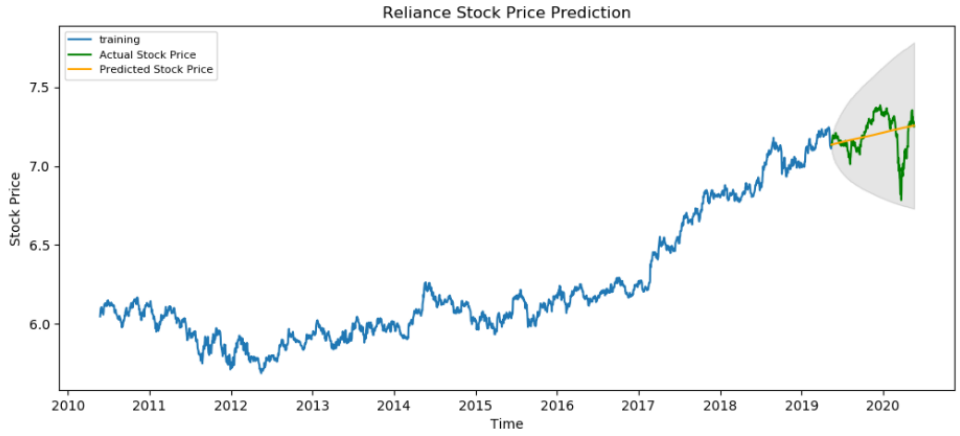


ARIMA Plotting

In [131]:
```
model = ARIMA(train_data, order=(0, 1, 1))
fitted = model.fit(disp=-1)
print(fitted.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:           D.Adj Close   No. Observations:                 2211
Model:                 ARIMA(0, 1, 1)   Log Likelihood                5950.032
Method:                       css-mle   S.D. of innovations              0.016
Date:                Wed, 20 May 2020   AIC                         -11894.063
Time:                        04:01:57   BIC                         -11876.960
Sample:                             1   HQIC                        -11887.815

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0005      0.000      1.355      0.175      -0.000       0.001
ma.L1.D.Adj Close     0.0404      0.021      1.892      0.058      -0.001       0.082
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
MA.1          -24.7801           +0.0000j           24.7801            0.5000
-----------------------------------------------------------------------------
```

In [132]:
```
# Forecasting values
fc, se, conf = fitted.forecast(247, alpha=0.05)  # 95% confidence value
fc_series = pd.Series(fc, index=test_data.index)
lower_series = pd.Series(conf[:, 0], index=test_data.index)
upper_series = pd.Series(conf[:, 1], index=test_data.index)
plt.figure(figsize=(12,5), dpi=100)
plt.plot(train_data, label='training')
plt.plot(test_data, color = 'green', label='Actual Stock Price')
plt.plot(fc_series, color = 'orange',label='Predicted Stock Price')
plt.fill_between(lower_series.index, lower_series, upper_series,
                 color='k', alpha=.10)
plt.title('Reliance Stock Price Prediction')
plt.xlabel('Time')
plt.ylabel('Stock Price')
plt.legend(loc='upper left', fontsize=8)

plt.show()
```



In [133]:
```
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [153]:
```
#Performance:
mse = mean_squared_error(test_data, fc)
print('MSE: '+str(mse))
mae = mean_absolute_error(test_data, fc)
print('MAE: '+str(mae))
rmse = math.sqrt(mean_squared_error(test_data, fc))
print('RMSE: '+str(rmse))
mape = np.mean(np.abs(fc - test_data)/np.abs(test_data))
print('MAPE: '+str(mape))
```

```
MSE: 0.013511206582547987
MAE: 0.0852704149709506
RMSE: 0.11623771583504205
MAPE: 0.011900728052480233
```

Monte Carlo Simulation

We are going to plot 100 Possible Simulations of Reliance Closing price for one Business year i.e. 252 days.

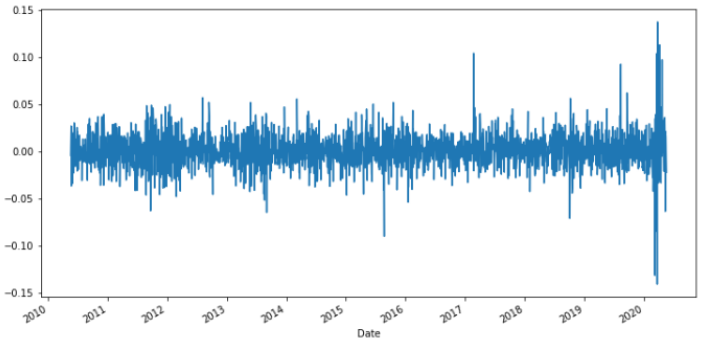
In [63]:
```
log_ret = np.log(1 + rel['Adj Close'].pct_change())
```

In [64]:
```
log_ret.tail()
```

Out[64]:
```
Date
2020-05-13    0.020995
2020-05-14   -0.041269
2020-05-15    0.016199
2020-05-18   -0.012862
2020-05-19   -0.022355
Name: Adj Close, dtype: float64
```

In [65]:
```
log_ret.plot(figsize=(12,6))
```

Out[65]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x120d4903408>
```



In [66]:
```
mu = log_ret.mean()
mu
```

Out[66]:
```
0.0004828074423122051
```

In [67]:
```
var = log_ret.var()
var
```

Out[67]:
```
0.00032357463073752013
```

drift = mu - 0.5*var

In [68]:
```
drift = mu - 0.5*var
drift
```

Out[68]:
```
0.00032102012694344505
```

In [69]:
```
stdev = log_ret.std()
stdev
```

Out[69]:
```
0.017988180306454573
```

In [70]:
```
t_interval=252
iterations=100
```

daily_returns = exp(r)

r = drift + stdev * z

In [192]:
```
daily_returns = np.exp(np.array([drift]) + np.array([stdev])*norm.ppf(np.random.rand(t_interval,iterations)))
```

In [193]:
```
daily_returns
```

Out[193]:
```
array([[0.98276929, 1.04391189, 1.02270986, ..., 0.96927518, 0.98895606,
        0.97090623],
       [1.0069067 , 0.98497045, 1.01757359, ..., 0.97678309, 0.99625032,
        0.99892231],
       [1.0239323 , 0.97379264, 0.97674584, ..., 0.97698332, 0.98543131,
        0.98459223],
       ...,
       [0.99291861, 0.99222616, 0.9866105 , ..., 1.03393752, 0.99794162,
        1.0065714 ],
       [0.99229644, 1.01235924, 1.00137277, ..., 0.99416939, 0.9841357 ,
        1.02074438],
       [0.99248792, 1.030004  , 0.97786622, ..., 1.00374938, 0.9804528 ,
        0.98893106]])
```

In [194]:
```
P0 = rel['Adj Close'].iloc[-1]
P0
```

Out[194]:
```
1408.9000244140625
```

In [195]:
```
Future_Price = np.zeros_like(daily_returns)
Future_Price
```

Out[195]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [196]:
```
Future_Price[0] = P0
Future_Price
```

Out[196]:
```
array([[1408.90002441, 1408.90002441, 1408.90002441, ..., 1408.90002441,
        1408.90002441, 1408.90002441],
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ],
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ],
       ...,
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ],
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ],
       [   0.        ,    0.        ,    0.        , ...,    0.        ,
           0.        ,    0.        ]])
```

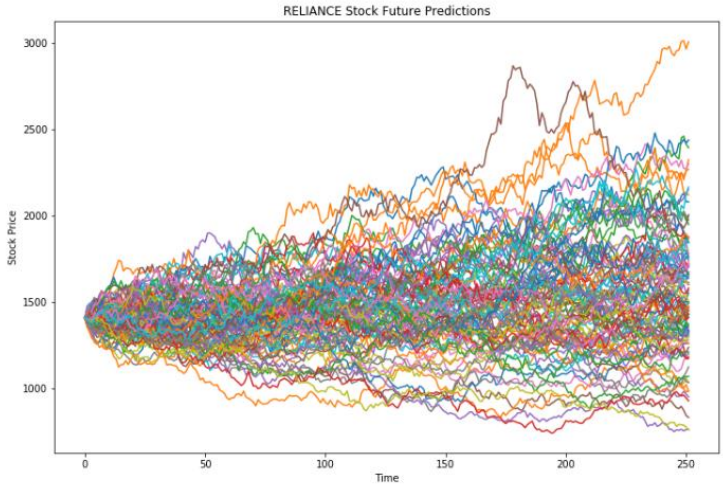
In [197]:
```
for x in range(1,t_interval):
    Future_Price[x] = Future_Price[x-1] * daily_returns[x]
```

In [198]:
```
Future_Price
```

Out[198]:
```
array([[1408.90002441, 1408.90002441, 1408.90002441, ..., 1408.90002441,
        1408.90002441, 1408.90002441],
       [1418.63087275, 1387.72488834, 1433.65945528, ..., 1376.18971797,
        1403.61710328, 1407.38166159],
       [1452.58197577, 1351.35628153, 1400.32090717, ..., 1344.51439577,
        1383.16823953, 1385.69704639],
       ...,
       [1692.79349458, 2228.27102414, 1795.60774259, ..., 1452.53074547,
         787.72766368, 1825.8792293 ],
       [1679.75296552, 2255.81076652, 1798.07270021, ..., 1444.06160581,
         775.230915  , 1863.7559676 ],
       [1667.13452756, 2323.49412073, 1758.27456315, ..., 1449.47594665,
         760.07731881, 1843.12615819]])
```

In [200]:
```
plt.figure(figsize=(12,8))
plt.plot(Future_Price)
plt.title('RELIANCE Stock Future Predictions')
plt.xlabel('Time')
plt.ylabel('Stock Price')
```

Out[200]:
```
Text(0, 0.5, 'Stock Price')
```



Monte Carlo model gave us 100 iterations of possible stock prices.

Forecasting with Prophet

Type *Markdown* and LaTeX: α^2

In [176]:
```
reliance = pd.DataFrame(rel['Adj Close'].reset_index())
reliance.columns = ['ds','y']
reliance
```

Out[176]:

	ds	y
0	2010-05-20	429.394226
1	2010-05-21	427.504791
2	2010-05-24	439.141998
3	2010-05-25	423.253540
4	2010-05-26	432.872467
...	...	...
2457	2020-05-13	1496.449951
2458	2020-05-14	1435.949951
2459	2020-05-15	1459.400024
2460	2020-05-18	1440.750000
2461	2020-05-19	1408.900024

2462 rows × 2 columns

In [177]:
```
import fbprophet
```

In [178]:
```
pred = fbprophet.Prophet()
```

In [179]:
```
pred.fit(reliance)
```

```
INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.
```

Out[179]:
```
<fbprophet.forecaster.Prophet at 0x120ed883988>
```

In [180]:
```
future = pred.make_future_dataframe(periods=365)
```

In [181]:
```
future.tail()
```

Out[181]:

	ds
2822	2021-05-15
2823	2021-05-16
2824	2021-05-17
2825	2021-05-18
2826	2021-05-19

In [182]:
```
forecast_val = pred.predict(future)
forecast_val.tail()
```

Out[182]:

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	weekly	weekly_
2822	2021-05-15	1680.764069	1570.685428	1752.845634	1637.470130	1720.760750	-19.603131	-19.603131	-19.603131	-12.571496	-12.5
2823	2021-05-16	1681.361131	1575.520333	1747.577928	1637.833017	1721.532013	-20.196967	-20.196967	-20.196967	-12.571496	-12.5
2824	2021-05-17	1681.958192	1583.445218	1762.099685	1638.195904	1722.303276	-3.737881	-3.737881	-3.737881	4.454131	4.4
2825	2021-05-18	1682.555253	1583.663069	1773.547616	1638.558791	1723.052044	-3.167507	-3.167507	-3.167507	5.553798	5.5
2826	2021-05-19	1683.152314	1587.581639	1772.343713	1638.921678	1723.794760	-4.724209	-4.724209	-4.724209	4.479699	4.4

In [183]:
```
forecast_val.describe()
```

Out[183]:

	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	weekly	weekly_lower
count	2827.000000	2827.000000	2827.000000	2827.000000	2827.000000	2827.000000	2827.000000	2827.000000	2827.000000	2827.000000
mean	750.099097	676.321540	832.537711	748.055960	752.026382	4.430171	4.430171	4.430171	4.380261	4.380261
std	451.463229	449.223557	451.054880	447.532620	455.242390	13.843471	13.843471	13.843471	3.345004	3.345004
min	331.202101	216.633323	372.420702	331.202101	331.202101	-52.740070	-52.740070	-52.740070	-12.571496	-12.571496
25%	407.859322	333.474294	488.499707	407.859322	407.859322	-1.457748	-1.457748	-1.457748	4.479699	4.479699
50%	460.756353	390.178615	545.901917	460.756353	460.756353	7.306391	7.306391	7.306391	5.315110	5.315110
75%	1156.842997	1094.466750	1251.049710	1156.842997	1156.842997	13.989069	13.989069	13.989069	5.340253	5.340253
max	1683.152314	1591.212343	1773.547616	1638.921678	1723.794760	25.147239	25.147239	25.147239	5.553798	5.553798

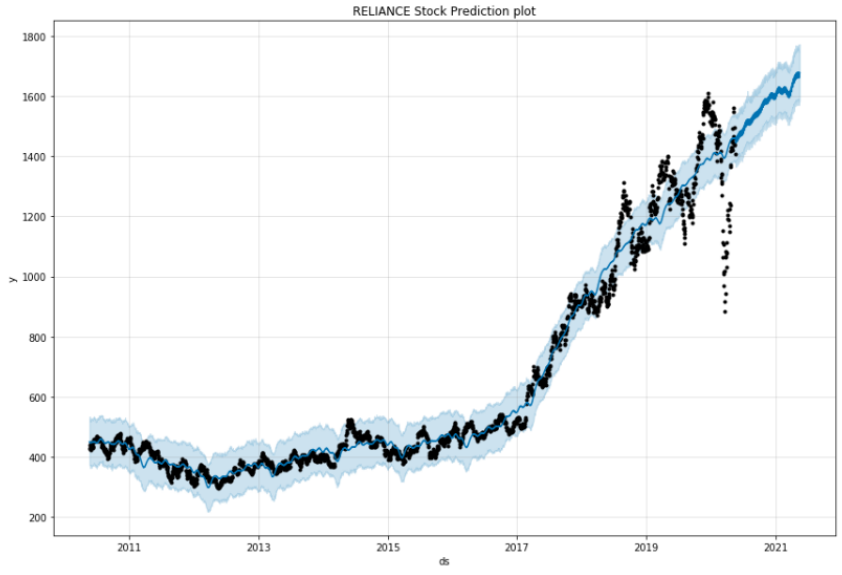
In [184]:
```
reliance.describe()
```

Out[184]:

	y
count	2462.000000
mean	632.970021
std	348.758006
min	295.313965
25%	395.762108
50%	460.514999
75%	887.197876
max	1609.949951

In [201]:
```
#Daily predictions
fig = pred.plot(forecast_val, figsize=(12,8))
plt.title('RELIANCE Stock Prediction plot')
```

Out[201]:
```
Text(0.5, 1, 'RELIANCE Stock Prediction plot')
```



In [186]:
```
fig2 = pred.plot_components(forecast_val)
plt.show()
```