A

Project Report

ON

“Time Series Analysis and Forecasting of Stock Prices using R”

UNDERTAKEN AT

“MIT School of Distance Education”

IN PARTIAL FULFILMENT OF

“PGCM - Business Analytics”

MIT SCHOOL OF DISTANCE EDUCATION, PUNE.

GUIDED BY

“**Prof. Dr. Dipti Kalkotwar**”

SUBMITTED BY

“**APURVA CHAURASIA**”

STUDENT REGISTRATION NO.: MIT2022N01201

MIT SCHOOL OF DISTANCE EDUCATION PUNE - 411 038

YEAR 2022-23

MIT | School of Distance Education

EXEMPT CERTIFICATE

To

The Director,

MIT School of Distance Education,

Respected Sir,

This is to request you to kindly exempt me from submitting the certificate for Project Work due to the reason mentioned below:

Tick the right option

1. As per the Rules of the Organisation
2. Self Employed
3. Working in Public Sector
4. [✓ Full-time Student]

Thanking you in anticipation of your approval to my request.

Kind Regards,

Student Sign: - [illegible]

Student Name:- Apurva Chaurasia

Student ID - MIT2022N01201

DECLARATION

I hereby declare that this project report entitled "**Time Series Analysis and Forecasting of Stock Prices using R**" is a bonafide record of the project work carried out by me during the academic year 2022-2023, in fulfillment of the requirements for the award of "PGCM - Business Analytics" of MIT School of Distance Education.

This work has not been undertaken or submitted elsewhere in connection with any other academic course.

Sign:- [illegible]

Name:- Apurva Chaurasia

Student ID: MIT2022N01201

ACKNOWLEDGEMENT

I would like to take this opportunity to express my sincere thanks and gratitude to Prof. Dr. Dipti Kalkotwar, Faculty of MIT School of Distance Education, for allowing me to do my project work under her valuable guidance. It has been a great learning and fruitful experience.

I would like to sincerely express my deep sense of gratitude and profound thanks to all staff members of MIT School of Distance Education for their kind support and cooperation which helped me in gaining knowledge. skills and experience to do my project work successfully.

I am also thankful to my colleagues at Tata Consultancy Services, friends at university and my parents for their moral support, endurance and encouragement during the project period when I could work in conflicting time zones due to their blessings.

Sign:- [illegible]

Name:- Apurva Chaurasia

Student ID: MIT2022N01201

ABSTRACT

This project works titled as "**Time Series Analysis and Forecasting of Stock Prices using R**" deals with Time Series analysis and forecasting methodologies using R programming language. The project has been carried out taking two stocks, i.e., HDFC Bank and Nifty 50 for demonstration and application purpose.

The analysis covers in depth descriptive statistics, daily, weekly and monthly trends charts, Candlestick chart and deployment of normality tests in statistical as well as graphical formats. The financial compounded returns have been calculated for HDFC Bank and Nifty 50 and subsequent forecasting exercise has been undertaken for 'Returns' as well as 'Prices' of these two stocks. Nonparametric correlation coefficients have been calculated to check the relationships between these returns if any.

This project work focuses upon the practical applications of AUTO ARIMA, fitting of best distributions and probabilistic random part range estimation using Monte Carlo simulations for HDFC Bank and Nifty returns as well as prices. Since ARIMA requires stationarity in data, autocorrelation and partial auto correlation tests have been carried out to check the same. If the need arises to convert the non-stationary data into stationary data, the same is accomplished automatically by AUTO ARIMA and appropriate values of p, d, q are derived. Subsequently, model building is carried out for prediction. RMSE / Predicted values ratio has been used to check the model effectiveness. In our case, these ratios were found to be quite high prompting us to delve into residuals analysis using AUTO ARIMA. Residuals also showed presence of stationarity and therefore ARIMA-GARCH family of models were not deployed.

It was logical to assume that model in-effectiveness was partly on account of non-normality of data. To address this, power transformations were used in AUTO ARIMA, and model building was repeated. Then, BoxCox transformations were tried but RMSE was surprisingly found lower for non-transformed data.

For prediction of random component of time series data of stocks, four distributions, namely., normal, logistic, Cauchy, and t distributions were fitted. Based on Loglikelihood, AIC and BIC parameters, best fitting distribution (t distribution in our case) was identified. This conclusion about best distribution was also validated graphically using QQ plots and in one more statistical way by using goodness of fit tests.

Next step was to predict the range of random part using Monte Carlo simulations. These simulations were also carried out using above four distributions and predicted values were arrived with 1% and 5% probabilities.

Machine learning models, namely, Random Forest and Neural Networks were deployed as alternative models to augment forecasting accuracy. The prediction accuracy here was significantly better and Random Forest proved out to be the best model for our data sets.

Looking forward, the integration of AI-driven techniques and algorithmic trading is poised to revolutionize time series analysis and forecasting. AI algorithms, particularly deep learning

models, exhibit promise in capturing intricate temporal dependencies and nonlinear patterns within time series data. Furthermore, algorithmic trading, driven by sophisticated predictive models, offers the potential for automated decision-making in financial markets, paving the way for more efficient and adaptive trading strategies.

In conclusion, while acknowledging the limitations of available models, this project report underscores the application of algorithmic trading as the future of time series analysis and forecasting, opening avenues for enhanced predictive accuracy and automated decision-making in various domains.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

In the dynamic and ever-evolving realm of the stock market, where fortunes are made and lost in the blink of an eye, the ability to harness the predictive power of time series data analysis has become paramount. Time series data, representing a sequence of observations collected over successive intervals, serves as a rich foundation that encapsulates the historical performance of stocks.

The importance of time series data analysis in the context of stock market forecasting lies in its capacity to unveil patterns, trends, and dependencies within historical stock prices. By scrutinizing past performance, analysts can discern valuable insights that inform future investment decisions. This analytical approach is not limited to mere hindsight; rather, it empowers stakeholders to anticipate market trends, identify potential risks, and optimize investment strategies.

In the intricate landscape of stock markets, where external factors, market sentiment, and global events influence stock prices, the need for accurate forecasting has never been more pronounced. Time series analysis provides a robust framework for modeling and understanding the inherent dynamics of stock prices, enabling practitioners to make informed predictions about future market movements. From day trading to long-term investments, the ability to anticipate price fluctuations is a critical asset in navigating the complexities of financial markets.

This exploration into the importance of time series data analysis and forecasting for stock prices is a strategic advantage for stock market stakeholders. As we delve into the intricacies of historical data, statistical models, and cutting-edge forecasting techniques, we unlock the potential to make proactive decisions, mitigate risks, and ultimately enhance the prospects of success in the ever-volatile world of stock trading.

The utilization of AUTO ARIMA represents a fundamental step towards automating the process of identifying optimal parameters for time series forecasting models. This approach streamlines the modeling process and enhances the predictive capabilities by efficiently capturing temporal dependencies and patterns inherent in sequential data. Furthermore, the exploration of machine learning models like Random Forest and Neural Networks extends the analytical horizon beyond traditional statistical methods, enabling the recognition and interpretation of complex nonlinear relationships within time series data.

Moreover, Monte Carlo simulations stand as a robust technique for generating a multitude of potential future scenarios. By integrating four probability distributions—normal, logistic, Cauchy, and t distributions—these simulations offer a probabilistic view of outcomes, enabling comprehensive risk evaluation and improved decision-making in uncertain environments.

This report focuses upon the practical application of these methodologies within the context of time series analysis and forecasting. By harnessing the capabilities of R language, this project endeavors to provide a detailed understanding of effective strategies for analyzing time series data and forecasting future trends.

History and Evolution of Time Series Forecasting

This project report deals with analysis and forecasting for HDFC Bank and Nifty 50 – Indian market index. The history of forecasting is deeply rooted in statistical methods, evolving over time with advancements in technology and data analysis techniques.

Time series analysis began with the work of Francis Galton and Louis Bachelier, who explored randomness in financial markets in late 19th Century. Pioneers like Norbert Wiener and John W. Tukey laid the groundwork for statistical analysis and time series techniques in early to mid-20th Century.

The use of autoregressive models (AR), moving average models (MA), and autoregressive integrated moving average models (ARIMA) gained popularity for analyzing and predicting stock price movements 1960s - 1980s. Late 20th Century saw introduction of GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models to capture volatility clustering and ARCH (Autoregressive Conditional Heteroskedasticity) models to model financial time series with changing variance.

In our times. the advent of computational power and big data revolutionized time series analysis. Machine learning techniques like neural networks, random forest and many other deep learning methods have been applied for stock price forecasting recently.

Current Status and Future Directions

Today, time series analysis for stock price forecasting involves a mix of statistical methods, machine learning, and deep learning techniques. High-frequency trading and the availability of vast amounts of financial data have made real-time analysis crucial. We were lucky to get the data as a free resource on https://finance.yahoo.com/quote.

Future directions reveal following possibilities.

Improved Deep Learning Models: Development of more sophisticated neural networks to handle complex stock market dynamics.

AI-Driven Predictive Analytics: Integration of AI to enhance forecasting accuracy by considering a wider range of data sources.

Explainable AI: Focus on making models interpretable to understand the rationale behind forecasts.

Quantum Computing: Potential utilization of quantum computing for faster and highly complex analysis of financial markets data.

In future, the field is likely to witness a convergence of various disciplines viz. finance, statistics, and computer science, leading to more robust and accurate stock price forecasting methods.

CHAPTER 2: ORGANIZATIONAL PROFILE

Organization:

National Stock Exchange of India (NSE) was established in 1992 as the first electronic exchange in India. It has headquarters in Mumbai and is a leading stock exchange in our country, providing a platform for trading in equities, derivatives, debt, and currency derivatives.

NSE operates on a fully automated screen-based electronic trading system, offering transparency, efficiency, and reliability in trading activities.

Timeline:

1992: NSE began operations, introducing electronic trading in India.

1994: **Nifty 50, the benchmark index**, was launched to track the performance of the top 50 stocks listed on the exchange.

2000: NSE introduced the first exchange-traded fund (ETF) in India, called Nifty BeES (Benchmark Exchange Traded Scheme).

2008: NSE launched currency futures trading.

2016: NSE started trading in interest rate futures (IRFs).

2020: NSE introduced the NSE IFSC Limited in the GIFT City (Gujarat International Finance Tec-City), facilitating global investors' access to Indian markets.

Management Team:

The NSE is governed by a Board of Directors comprising individuals with expertise in finance, law, economics, and related fields. The management team is led by a CEO and Managing Director, overseeing various operational and strategic aspects of the exchange. The leadership ensures compliance with regulatory frameworks, promotes innovation, and drives the exchange's growth and development. The current Board of Directors consists of following dignitaries.

Mr. Ashishkumar Chauhan	- Managing Director & CEO
Mr. Yatrik Vin	- Group Chief Financial Officer & Head - Corporate Affairs
Mr. Sriram Krishnan	- Chief Business Development Officer
Mr. Somasundaram K S	- Chief Enterprise Risk Officer
Mr. Shharad Dhakkate	- Chief Human Resources Officer
Mr. Piyush Chourasia	- Chief Regulatory Officer
Mr. Mayur Sindhwad	- Chief Technology Officer- Operations
Mr. Viral Mody	- Chief Technology Officer- Applications & Development

Product/Service Profile:

Equities and Derivatives: NSE facilitates trading in a wide range of equities and derivatives, offering various products such as futures, options, and indices.

Debt Market: Provides a platform for trading in government and corporate bonds, promoting liquidity and transparency in debt securities.

Currency and Interest Rate Derivatives: Offers currency futures and interest rate futures, allowing participants to hedge against currency and interest rate risks.

Indices and Investment Products: Nifty indices track market movements and are used as benchmarks for various investment products like ETFs and index funds.

NSE vis-a-vis Stock Price Prediction:

Technological Advancements: NSE's adoption of electronic trading revolutionized stock market operations, enabling faster transactions and data availability for analysis.

Data Availability: Comprehensive data archives offer valuable historical information, facilitating the development and testing of stock price prediction models and algorithms.

Platform for Derivatives: Introduction of derivatives trading, including futures and options, has expanded the scope of predictive analytics by providing additional market indicators and tools for forecasting.

Research and Education Initiatives: NSE has undertaken initiatives to educate market participants, researchers, and analysts about market dynamics, encouraging the development of advanced prediction methodologies.

National Stock Exchange (NSE) stands as a pivotal institution in India's financial landscape. Its robust infrastructure, diverse product offerings, and commitment to technological innovation have significantly contributed to the evolution of stock price prediction approaches. By fostering transparency, liquidity, and accessibility, NSE continues to shape the development of financial markets and predictive analytics in the stock market domain.

CHAPTER 3: PROJECT OBJECTIVES AND SCOPE

Objectives of Project:

In today's highly competitive business environment, companies need more from Finance than accurate financial statements and reports. Forward-looking, predictive insights are key that can help shape tomorrow's business strategy and improve day-to-day decision-making in real time. Since the 2008 financial crisis, market practitioners are realizing that reliance on models which are mathematically pure but fundamentally inaccurate is no longer acceptable. A more practical approach is needed.

Data Science is providing computational intelligence to businesses in ways many had never envisioned. Analytics is a practical and pragmatic approach where statistical rules and discrete structures are automated on the datasets as outcomes are observed in the laboratory and in the business world.

In today's world, businesses as well as consumers are affected by fluctuations in consumer prices, industrial production, interest rates, and the price of natural gas. Financial analytics involves applying classic statistical models and computerized algorithms to the financial market data and investment portfolios. It addresses relationships that occur in practice every day in time-critical fashion as investors, speculators, and producers across the country and the globe.

The basic objective is to determine a model that describes pattern of time series & to forecast future values of the series. Towards this end, we have various objectives as described below.

Long-Term Trend Analysis: To analyze and forecast long-term trends in the Indian stock market.

Data Analysis and Interpretation: To analyze data to extract meaningful insights. It involves cleaning, processing, and analyzing data to uncover hidden relationships among variables.

Pattern Identification: To analyze historical stock data to identify recurring patterns, trends, and seasonality in the Indian market.

Model Evaluation and Selection: To compare different time series models (like ARIMA etc.) to determine which performs best in forecasting Indian stock prices and returns.

Prediction Accuracy: To evaluate the accuracy of various forecasting models in predicting stock prices in the Indian market.

Machine Learning Integration: To explore integration of machine learning techniques for stock price prediction and assess their effectiveness compared to statistical methods like AUTO ARIMA.

These objectives have been set to delve into specific aspects of time series analysis and forecasting within the Indian stock market context by taking one large cap stock, namely HDFC bank and market index, Nifty 50. The project work is aimed at throwing various insights and valuable guidance for market stakeholders -investors, analysts, students, faculty members and uninitiated individuals in stock market investments.

Steps of Project Research Work:

Key Steps in conducting the research for this project work are as follows.

1. Defining Objectives and Scope - First step is to define the research objectives & outline the boundaries of the study, including the time frame, data sources, and specific methodologies to be employed.

2. Data Collection and Preprocessing - This requires gathering historical stock price data from reliable sources like https://finance.yahoo.com/quote/. Then we check collected data for errors, missing values, and inconsistencies. WE Convert raw data into a usable format, data frame or a time series structure suitable for analysis.

3. Exploratory Data Analysis (EDA)- Here, we conduct statistical summaries, visualizations, and exploratory graphs to understand patterns, trends, and correlations in the data.

4. Model Selection and Development - Appropriate models (e.g., ARIMA, GARCH, ML) need to be selected based on the characteristics observed during EDA and the project objectives. Then, we fit the chosen models to historical data, optimizing model parameters to achieve the best fit and forecasting accuracy.

5. Model Validation and Testing - Out-of-Sample Testing is carried out to evaluate model performance to assess forecasting accuracy. Data us split into training and test data. Model is built using training data and its accuracy is testes based on test data.

6. Forecasting and Analysis - Future stock price predictions are made for the desired time horizon. Performance evaluation is done by compare forecasted prices with actual values to assess the accuracy and effectiveness of the models.

7. Interpretation and Conclusion - It is important to analyze and interpret the findings, discussing the implications of the forecasted trends and patterns. Summary of outcomes provide insights for all stakeholders.

8. Documentation and Reporting-Final step is to document all steps, methodologies, findings, and limitations encountered during the project work. Compilation of the project work into a coherent report like this, including methodologies, results, conclusions, and recommendations becomes a reference for further work.

Need of the Project:

The Need for a project work on Time Series Analysis and Forecasting of Stock Prices arises due to following factors.

Decision-Making Support- Investors, traders, and financial analysts require reliable insights into future stock price movements to make informed decisions regarding buying, selling, or holding securities. Forecasting models aid in strategic decision-making.

Portfolio Optimization - Investors seek to optimize their portfolios for maximum returns with controlled risk. Forecasting models aid in selecting stocks or assets that align with an individual's risk tolerance and investment goals.

Real-World Applications - Time series analysis and forecasting have practical applications in various domains beyond finance. Industries like insurance, retail, and manufacturing use similar techniques to predict demand, sales, and other vital parameters.

Economic and Financial Policy Implications- Accurate stock price forecasts have implications for economic policies and financial regulations. Understanding market behavior aids policymakers in making informed decisions to maintain market stability.

Technological Advancements - With the advancement of computational techniques and big data analytics, there's an opportunity to innovate and refine predictive models, contributing to the development of cutting-edge methodologies.

Continuous Market Monitoring - Regular time series analysis enables continuous monitoring of market trends, empowering market participants to adapt to changing conditions and seize emerging opportunities.

Education and Research Advancement-Conducting such studies contributes to the collective knowledge base, advancing research in finance and analytics, and fostering continuous improvement in forecasting methodologies.

In summary, a project study like this addresses the pressing needs of stakeholders facilitating better decision-making, and overall market efficiency in the dynamic landscape of financial markets.

Scope of Project:

The scope of the project report on "**Time Series Analysis & Forecasting of Stock Prices using R**" encompasses following aspects.

Statistical/Graphical aspects	Applicability
Plots of Financial Data (Visualizations)	HDFC Bank & Nifty daily, weekly and monthly data
Financial Returns	HDFC Bank & Nifty daily data
EDA	HDFC Bank & Nifty compounded returns
Normality tests, Auto correlation and partial auto correlation, Box test for Stationary Time Series	HDFC Bank & Nifty compounded returns as well as daily prices
Model Building using ARIMA, residual analysis, Power Transformations, BoxCox transformations	HDFC Bank & Nifty compounded returns as well as daily prices
Fitting and evaluating distributions (Normal, Logistic, Cauchy and t), QQ plots, goodness of fit tests	HDFC Bank compounded returns as well as daily prices only
Building and evaluating ML models – Random forest and neural network	HDFC Bank & Nifty daily prices only

Type of Project Research:

This study is completely PRACTICAL in nature. Only necessary theoretical aspects have been included and all analytical aspects have been dealt with by using real time data.

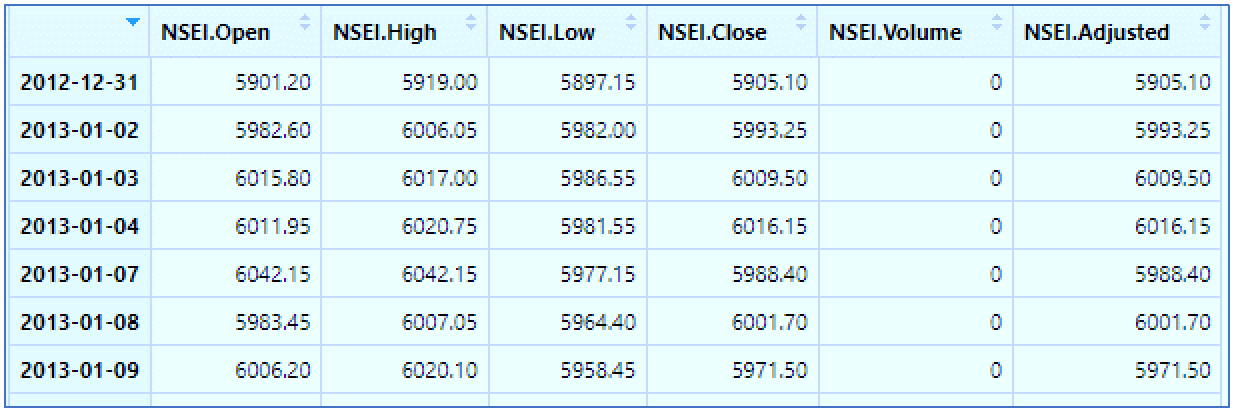
Data Collection Method:

A time series is a sequence where a metric is recorded over regular time intervals. Depending on the frequency, a time series can be of yearly (ex: annual budget), quarterly (ex: expenses), monthly (ex: air traffic), weekly (ex: sales qty), daily (ex: stock price).

Primary Data: It is not applicable to this project work.

Secondary Data: Timeseries **data source https://finance.yahoo.com/quote**. It has been extracted using R function. **HDFC bank** has been considered as the sample stock for analysis along-with market index **Nifty 50**. The data period has been considered from 31-12-2012 to 10-11-2023. The data is available in following format (Table 3.1) for daily frequency. For analysis purpose, data has been transformed into weekly as well as monthly frequency also and has been included as part of analysis in next chapter.

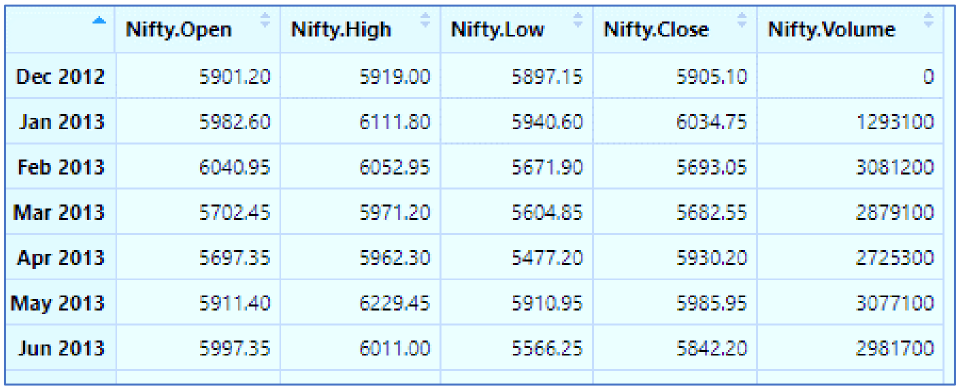
Table 3.1



	NSEI.Open	NSEI.High	NSEI.Low	NSEI.Close	NSEI.Volume	NSEI.Adjusted
2012-12-31	5901.20	5919.00	5897.15	5905.10	0	5905.10
2013-01-02	5982.60	6006.05	5982.00	5993.25	0	5993.25
2013-01-03	6015.80	6017.00	5986.55	6009.50	0	6009.50
2013-01-04	6011.95	6020.75	5981.55	6016.15	0	6016.15
2013-01-07	6042.15	6042.15	5977.15	5988.40	0	5988.40
2013-01-08	5983.45	6007.05	5964.40	6001.70	0	6001.70
2013-01-09	6006.20	6020.10	5958.45	5971.50	0	5971.50

If there is a problem with data quality in terms of missing values, they are appropriately treated. Next step is to explore data to understand the data and find scenarios for performing the analysis. In our case, we need to transform our data into OHLC chart format in Table 3.2.

Table 3.2



	Nifty.Open	Nifty.High	Nifty.Low	Nifty.Close	Nifty.Volume
Dec 2012	5901.20	5919.00	5897.15	5905.10	0
Jan 2013	5982.60	6111.80	5940.60	6034.75	1293100
Feb 2013	6040.95	6052.95	5671.90	5693.05	3081200
Mar 2013	5702.45	5971.20	5604.85	5682.55	2879100
Apr 2013	5697.35	5962.30	5477.20	5930.20	2725300
May 2013	5911.40	6229.45	5910.95	5985.95	3077100
Jun 2013	5997.35	6011.00	5566.25	5842.20	2981700

Clearly the last column has been omitted compared to earlier format.

Missing data in Time Series

Missing data can arise for many reasons, and it is important to consider if the missingness will induce bias in the analysis. When missing values lead to errors, there are two ways to handle the problem. First, we could just omit missing value, assuming there is a long list of observations like time series. Alternatively, we could deal with the missing values by using estimates.

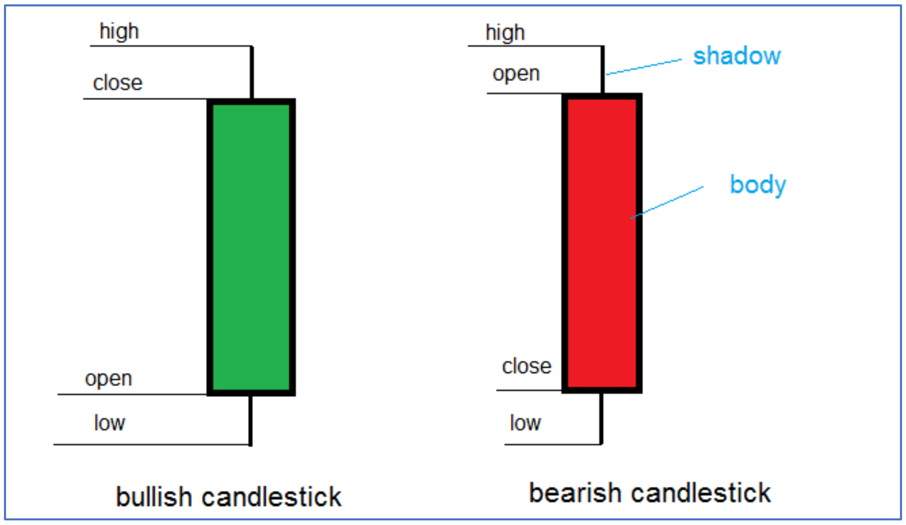
Sample Size: 10 years data has been used in this study starting from 31-12-2012 to 10-11-2023.

TOOLS & TECHNIQUES:

We deploy several tools and techniques for analysis and forecasting as summarized below.

Data Visualization:

Data trend is plotted to see daily, weekly and monthly trend. OHLC chart is plotted to facilitate the plotting of candle stick chart. An OHLC chart is a type of bar chart that shows open, high, low, and closing prices for each period. OHLC charts are useful since they show the four major data points over a period, with the closing price being considered the most important by many traders.



The chart type is useful because it can show increasing or decreasing momentum. When the open and close are far apart it shows strong momentum, and when the open and close are close together it shows indecision or weak momentum. The high and low show the full price range of the period, useful in assessing volatility.

Financial Returns

Returns are crucial for comparability among stocks. There are arithmetic returns and geometric returns. Arithmetic returns, also known as simple returns, measure the average periodic rate of return over a specific time period. It is calculated by taking the difference between the final and initial values of an investment, considering only the cash income and ignoring compounding effects. Geometric returns, also known as compound returns, account for the effect of compounding over time. Geometric returns are calculated by taking the nth root of the total return over n periods, subtracting 1 to express the result as a percentage.

Exploratory Data Analysis:

Exploratory Data Analysis (EDA) serves as a crucial preliminary step in unraveling the intricacies of time series data, offering insights into patterns, trends, and anomalies that lay hidden within temporal dynamics. Here are key aspects of EDA for time series data:

Descriptive Statistics:

Basic descriptive statistics, including measures like mean, median, variance, skewness and kurtosis provide a snapshot of central tendencies and distributional properties. These statistics illuminate the overall behavior and variability of the time series.

If skewness and kurtosis are close to zero, the data is said to be symmetric and similar to normal data. Generally, we consider data with skewness and kurtosis less than 0.5 as normal for analysis. If skewness is more than 0.5 data is positively skewed and if it is less than -0.5, data is negatively skewed.

Kurtosis represents flatness of tail. If it is more than 0.5 data is flat tailed and if it is less than 0.5, data is not possessing any tail. Normally there are very few instances of negative kurtosis. Uniform distribution is one such example.

Such data is said to be non-normal and needs transformation for further analysis.

Outlier Detection:

Identification and treatment of outliers are crucial in EDA. Extreme values can distort analysis and forecasting. Visualization tools like box plots may be employed to detect and handle outliers.

Correlation:

In the context of time series data, correlation provides insights into the linear relationship between two time-dependent variables. A high positive/negative correlation between two time series indicates a strong tendency for the variables to move in the same/opposite direction. Analyzing the correlation structure in time series data aids in forecasting.

Spearman's rank correlation coefficient and Kendall's tau are non-parametric measures of association that assess the strength and direction of monotonic relationships between variables. In the context of time series forecasting, these correlation coefficients offer alternatives to Pearson correlation, particularly when dealing with non-linear relationships or ordinal data.

Spearman's rank correlation coefficient assesses the strength and direction of the monotonic relationship between two variables by comparing their ranks. It is calculated by converting the original data into ranks and then applying Pearson correlation to the ranked data.

Kendall's tau measures the strength and direction of the ordinal association between two variables. It is based on the count of concordant and discordant pairs of observations.

Kendall's tau is less sensitive to outliers than Pearson correlation, making it a robust measure in the presence of extreme values often encountered in time series data.

These coefficients are particularly useful when dealing with time series data that exhibit trends, seasonality, or irregular patterns, where the assumption of linearity may not hold.

Normality Testing

The Jarque-Bera Normality Test is a statistical test used to assess whether a given set of data follows a normal distribution. In the context of time series data analysis, checking for normality is important because many statistical methods and models assume that the data is normally distributed. The test is named after its developers, Carlos Jarque and Anil K. Bera. The Jarque-Bera test statistic is calculated based on skewness and kurtosis, two statistical measures that describe the shape of the data distribution. The test statistic follows a chi-square distribution under the null hypothesis of normality. By comparing the calculated test statistic to the critical values from the chi-square distribution, one can determine whether to reject the null hypothesis of normality. Alternatively, if p-value is less than 0.05, null hypothesis that data is normal is rejected.

It's important to note that the Jarque-Bera test is sensitive to sample size, and a large sample size may lead to a higher likelihood of rejecting the null hypothesis even for small deviations from normality.

The Kolmogorov-Smirnov (KS) test is another non-parametric statistical test used to assess whether a sample comes from a specific distribution, such as a normal distribution. The KS test compares the cumulative distribution function (CDF) of the sample with the expected CDF of the theoretical distribution. The test statistic, denoted as D, represents the maximum absolute difference between the two cumulative distribution functions. The null hypothesis of the KS test is that the sample is drawn from the specified distribution. if p-value is less than 0.05, null hypothesis that data is normal is rejected. For large sample sizes, the test may be overly sensitive, detecting minor discrepancies from the assumed distribution that may not be practically significant. By leveraging this non-parametric test, analysts can gain insights into the suitability of statistical models and make more informed decisions in the process of time series forecasting and analysis.

The Shapiro-Wilk test is a statistical test used to assess whether a sample follows a normal distribution. Named after its developers, Samuel S. Shapiro and Martin Wilk, this test is widely used for its sensitivity, especially in the case of smaller sample sizes. The Shapiro-Wilk test is based on the comparison of sample statistics and expected values under the assumption of normality. The test statistic, denoted as W, measures the degree of departure from normality. The null hypothesis of the test is that the data follows a normal distribution. The null hypothesis of the Shapiro-Wilk test is that the sample comes from a normally distributed population. If the p-value associated with the test is below a chosen significance level (commonly 0.05), the null hypothesis is rejected, indicating evidence against normality. The Shapiro-Wilk test is more powerful for smaller sample sizes compared to other normality tests like the Kolmogorov-Smirnov test.

Autocorrelation and Partial Autocorrelation

Autocorrelation & partial autocorrelation functions reveal temporal dependencies within the data.

Autocorrelation (ACF) measures the correlation between a time series and its lagged values at different time points. The ACF plot displays these correlations for various lags. ACF is crucial for identifying temporal patterns, seasonality, and cyclical behavior in time series data. Peaks or troughs in the ACF plot at specific lags indicate the presence of significant correlations, helping analysts discern the periodicity of the series. ACF aids in the selection of appropriate models, such as autoregressive (AR) models. The decay pattern in the ACF plot guides the determination of the order of the autoregressive component in ARIMA (AutoRegressive Integrated Moving Average) models. The ACF plot can be used to assess the stationarity of a time series. If the autocorrelation values decrease slowly, it may indicate non-stationarity, prompting the need for differencing.

Partial Autocorrelation (PACF) measures the correlation between a time series and its lagged values, excluding the influence of intermediate lags. The PACF plot displays these partial correlations for different lags. PACF is particularly useful in specifying the order of the autoregressive component in time series models. It helps identify the direct relationships between a variable and its past values without the confounding effects of intermediate lags. PACF assists in distinguishing between autoregressive (AR) and moving average (MA) components in an ARIMA model. A sharp drop in partial autocorrelation after a certain lag suggests a potential cut-off for the AR component. PACF aids in diagnosing model adequacy. Outlying partial autocorrelation values may indicate the need for additional model refinement.

ACF and PACF are instrumental in selecting appropriate time series models, which, in turn, enhances the accuracy of forecasting. By capturing the temporal dependencies inherent in the data, these plots guide the construction of models that better represent the underlying patterns.

Autocorrelation is considered statistically significant in time series data analysis when the calculated autocorrelation values are outside the range expected under the assumption of no correlation (i.e., when the autocorrelation values deviate from zero). The significance of autocorrelation is crucial for determining the presence of temporal dependencies and guiding the selection and refinement of time series models. Visual inspection of ACF and PACF plots can help identify statistically significant autocorrelation. Peaks or troughs in the ACF or PACF plot that extend beyond the confidence intervals indicate significant correlations at specific lags.

Likewise, Partial Autocorrelation Function (PACF) values are considered statistically significant in time series data analysis when they fall outside the confidence intervals, indicating a significant correlation between the observed variable and its lagged values, while accounting for the influence of intermediate lags. Significance assessment is crucial for determining the appropriate order of autoregressive (AR) terms in a time series model. Statistically significant PACF values guide the specification of autoregressive terms in models like the Autoregressive Integrated Moving Average

(ARIMA) model. The PACF plot aids in identifying the direct relationships between the variable and its past values without the confounding effects of intermediate lags.

Alternatively, statistically significant autocorrelation can be identified by applying Ljung-Box test in time series data analysis to assess the overall significance of autocorrelations at multiple lags. Named after statisticians Greta M. Ljung and George E.P. Box, this test is particularly valuable for detecting any significant departure from randomness in a time series, aiding in the evaluation of model adequacy and the identification of temporal dependencies.

The Ljung-Box test is based on the calculation of the test statistic Q, which is the sum of squared autocorrelations of the time series up to a specified lag. The test statistic follows a chi-square distribution under the null hypothesis of no autocorrelation. The null hypothesis of the Ljung-Box test is that there is no autocorrelation in the time series up to the specified lag. In other words, the autocorrelations at those lags are not statistically significant. If the p-value associated with the test is below 0.05, the null hypothesis is rejected, indicating auto correlation is present.

Stationarity Analysis

A time series whose statistical properties, such as mean, variance, etc., remain constant over time, are called a stationary time series. In other words, a stationary time series is a series whose statistical properties are independent of the point in time at which they are observed. A stationary time series has a constant variance, and it always returns to the long-run mean.

Stationarity, a critical assumption for many time series models, is explored during EDA. Augmented Dickey-Fuller (ADF) tests and visual inspections help assess stationarity. Differencing or transformation may be employed to achieve stationarity if necessary.

In essence, EDA serves as a compass in the exploration of time series data, guiding analysts to discern meaningful structures and relationships. By combining statistical techniques with visualizations, EDA facilitates a holistic understanding of temporal patterns, setting the stage for informed modeling and forecasting efforts in diverse fields including stock price prediction.

ARIMA – Forecasting

ARIMA, short for 'Auto Regressive Integrated Moving Average' is a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values. Any 'non-seasonal' time series that exhibits patterns can be modeled with ARIMA models.

Understanding p, d and q in ARIMA model:

An ARIMA model is characterized by 3 terms: p, d, q.

Where, p is the order of the AR term. q is the order of the MA term and d is the number of differencing required to make the time series stationary.

The first step to build an ARIMA model is to make the time series stationary. Because, term 'Auto Regressive' in ARIMA means it is a linear regression model that uses its own lags as predictors. Linear regression models, as we know, work best when the predictors are not correlated and are independent of each other. So, to make a series stationary, the most common approach is to difference it. That is, subtract the previous value from the current value. Sometimes, depending on the complexity of the series, more than one differencing may be needed. The value of d, therefore, is the minimum number of differencing needed to make the series stationary. And if the time series is already stationary, then d = 0.

'p' is the order of the 'Auto Regressive' (AR) term. It refers to the number of lags of Y to be used as predictors. And 'q' is the order of the 'Moving Average' (MA) term. It refers to the number of lagged forecast errors that should go into the ARIMA Model.

ARIMA Prediction Formula:

Unlike a multiple linear regression, in a time series model, we do not have separate dependent and independent variables. Here dependent and independent variables are same with time lag. For this reason, they are called auto regressive.

Then, Formula for ARIMA model is:

Predicted Yt = Constant + Linear combination Lags of Y (up to p lags) + Linear Combination of Lagged forecast errors (up to q lags)

Determining the right order of differencing is important. The right order of differencing is the minimum differencing required to get a near-stationary series which roams around a defined mean and the ACF plot reaches to zero quickly. If the autocorrelations are positive for many numbers of lags (10 or more), then the series needs further differencing. On the other hand, if the lag 1 autocorrelation itself is too negative, then the series is probably over-differenced.

The next step is to identify if the model needs any AR terms. We can find out the required number of AR terms by inspecting the Partial Autocorrelation (PACF) plot.

Similarly, we can look at the ACF plot for the number of MA terms. An MA term is technically, the error of the lagged forecast. The ACF tells how many MA terms are required to remove any autocorrelation in the stationary series.

Stationarity Tests

ARIMA works well on stationary data only. If data is not stationary, it needs to be made stationary using AR or MA. First, we test original data for stationarity. If data lacks stationarity, it means trend is present in the data. In such a case difference of data is taken and stationarity test is performed. If first difference is found stationary, it is taken as input for AR and MA in modelling. In case if difference is still not stationery, difference of differences is taken, and entire cycle is repeated. If data is not becoming stationary even after 2-3 differences, we need to explore other models than ARIMA.

The Augmented Dickey-Fuller (ADF) test is a statistical test used in time series analysis to assess the presence of a unit root in a univariate time series. The test helps determine whether a time series is stationary or exhibits a unit root, which can impact the choice of appropriate models for forecasting and analysis.

The null hypothesis of the ADF test is that a unit root is present in the time series, indicating that the series is non-stationary. The alternative hypothesis is that the series is stationary after differencing. If p is less than 0.05, H1 is accepted i.e. series is stationary.

Model Insights

ARIMA (0,0,0) would mean that data itself is stationary. There is no dependence on previous values or on past mistakes. There is no pattern in data. Prediction is coming out as constant plus random error. Thus, such a model is not adding any value to prediction process.

ARIMA (0,1,0) means there is no AR or MA term. First difference in data is being used for model building. Predicted values are having a constant difference. Such data behavior is called random walk.

Overfitting

Overfitting occurs when a model is too complex and captures random fluctuations in the training data, rather than the underlying trends. It can lead to poor out-of-sample performance and inaccurate forecasts.

Overfitting can occur if the differencing order is too high. Excessive differencing may introduce artificial patterns or trends that do not reflect the true behavior of the data. If orders of p and/or q are too high, the model may fit the noise in the data, capturing random fluctuations that do not represent genuine patterns.

An overfitted ARIMA model is likely to perform well on the training data but may fail to generalize to new, unseen data. The model becomes too tailored to the idiosyncrasies of the training set, leading to poor out-of-sample forecasting accuracy. We should strike a balance between capturing important features and avoiding the inclusion of unnecessary complexity. Regular model evaluation and validation techniques are crucial to addressing overfitting issues, ensuring that the ARIMA model provides accurate and reliable forecasts on new data.

Power Transformations

Power transformations are employed in time series analysis to stabilize variance, make the data more symmetric, and facilitate the fitting of ARIMA models. Common power transformations include the Box-Cox transformation and the logarithmic transformation.

The Box-Cox transformation is a family of power transformations that includes both logarithmic transformation (λ=0) and square root transformation (λ=0.5) as special cases. The parameter

λ is chosen to maximize the log-likelihood of the transformed data. This transformation is effective in stabilizing variance and addressing non-constant volatility. The logarithmic transformation is particularly useful when the time series exhibits exponential growth or decay. Log-transformed data often reduces the impact of extreme values and makes the series more amenable to linear modeling.

Once the time series data is transformed, ARIMA models can be fitted to the transformed series. The transformed model is then used to make predictions on the original scale by back transforming the forecasts.

Out of Sample Forecasting

Out-of-sample forecasting involves applying a trained time series model to a set of data not used during the model's development. This external dataset serves as a test, gauging the model's true predictive capabilities on unseen observations.

Models that perform exceptionally well on the training data may succumb to overfitting, capturing noise instead of true patterns. Out-of-sample testing acts as a safeguard, revealing whether a model generalizes well to new, unseen data.

Testing a model on out-of-sample data aids in evaluating its performance. Discrepancies between predicted and actual values signal areas for improvement, guiding model refinement for increased accuracy.

We divide dataset into training and testing subsets. The training set is used to build the model, while the testing set assesses its predictive performance.

Root Mean Squared Error (RMSE) is utilized to quantify the accuracy of out-of-sample forecasts and facilitate comparisons between models. The model with least error is better.

Out of Sample Forecasting Error Measures

Accuracy metrics to judge forecasts on out of sample data are:

Root Mean Squared Error (RMSE)

Mean Absolute Error (MAE)

Mean Absolute Percentage Error (MAPE)

Mean Squared Error (MSE)

Adjusted R^2 is not useful in isolation. Least AIC is the best model.

Fitting Distributions

There are many patterns in data. Different distributions capture different patterns. The idea is to map given data to a shape whose mathematical characteristics are known and can be used for generalization. The same can be used for simulation. Therefore, identification of a distribution is very important.

For a distribution to be applicable, data needs to be independent. Since time series data is dependent, we remove the dependency by performing ARIMA and residuals are taken for fitting a distribution.

Most common distributions in Finance are symmetric and right skewed distributions. The symmetric distributions include Normal, t, Cauchy, Logistic etc. distributions. In these distributions, skewness is zero, but kurtosis keeps varying. Right skewed distributions include exponential, Gamma, Lognormal, Weibull, Burr etc, distributions.

The normal distribution is characterized by its bell-shaped curve, emphasizing a central peak that represents the mean and a gradual tapering on either side. This symmetrical design succinctly captures the concept of central tendency, highlighting where data is most likely to cluster. The standard deviation, a measure of spread, delineates the extent of variability around the mean. Numerous algorithms and statistical models, such as linear regression and analysis of variance, assume a normal distribution of errors. This assumption simplifies computations and enhances the efficiency of these techniques, making them more applicable and interpretable.

The t-distribution shines brightest when sample sizes are small. Unlike the normal distribution, which assumes knowledge of the population standard deviation, the t-distribution employs the sample standard deviation, providing a more realistic estimate for uncertainty in small samples. The t-distribution's shape is governed by degrees of freedom, a parameter influenced by sample size. As sample size grows, the t-distribution converges toward the normal distribution. This dynamic quality reflects the distribution's adaptability to varying data scenarios.

In scenarios where outliers may exert undue influence, the t-distribution demonstrates resilience. Its heavier tails compared to the normal distribution render it less sensitive to extreme values, enhancing its reliability in the face of potential outliers.

Real-world data often deviates from the idealized assumptions of normality. The t-distribution's ability to accommodate uncertainty and handle deviations from normality makes it a robust choice for scenarios where data distribution may be ambiguous or skewed.

The hallmark of the Cauchy distribution lies in its heavy tails, extending infinitely in both directions. This peculiarity implies that extreme events, while rare, are more probable than anticipated by other distributions with finite variance. The Cauchy distribution is grounded in the Cauchy principle, emphasizing symmetry and a pivotal location parameter. This principle distinguishes it from normal distributions, challenging our intuition and guiding us to reconsider notions of centrality.

At the heart of the logistic distribution lies its characteristic S-curve, signifying gradual growth or decline. This curve elegantly mirrors real-world processes, capturing the essence of phenomena that exhibit slow initial progress, accelerated growth, and eventual stabilization. The probability density function of the logistic distribution unveils the likelihood of observing specific values. This function's flexibility makes it adept at characterizing a diverse array of situations, from estimating resource availability to predicting the likelihood of success in business ventures.

Extreme value modeling, essential in fields like finance, benefits from the logistic distribution's reliability. Its ability to characterize tail behavior makes it an asset when studying rare but impactful events. he logistic distribution's flexibility extends beyond its role in specific models. Its adaptability to diverse datasets and scenarios showcases its versatility as a distribution capable of embracing the complexities inherent in real-world data.

Our job is to find the distribution which most appropriately fits to our data of residuals.

Choosing the Best Model: Log Likelihood, AIC, and BIC in Harmony

In the landscape of statistical modeling, selecting the most fitting model is a critical endeavor. Log likelihood, Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC) emerge as guiding metrics, each offering a unique lens to evaluate models and strike a balance between goodness of fit and model complexity.

1. Log Likelihood:

At the core of model evaluation lies the log likelihood, quantifying the likelihood of observing the given data under the model. The higher the log likelihood, the better the model captures the observed data.

2. Akaike Information Criterion (AIC):

AIC adds a layer of sophistication by incorporating a penalty for model complexity. It aims to strike a balance between goodness of fit and the number of parameters. A lower AIC indicates a model that explains the data well while avoiding unnecessary complexity. AIC encourages parsimony, favoring models that offer a good fit with fewer parameters.

3. Bayesian Information Criterion (BIC):

BIC, akin to AIC, introduces a penalty for model complexity. However, BIC's penalty term is more stringent, especially for larger sample sizes. This emphasis on simplicity aligns with the Bayesian philosophy of preferring simpler models unless the data compellingly supports complexity. A lower BIC signifies a model that not only fits well but is also parsimonious.

For both AIC and BIC, a lower value indicates a better model. These metrics prioritize models that offer a good fit with minimal complexity.

While each metric provides valuable insights, a holistic approach often involves considering all three: maximizing log likelihood for goodness of fit, minimizing AIC for simplicity with good fit, and minimizing BIC for a more stringent emphasis on simplicity.

Graphical Selection:

In the intricate art of fitting distributions to data, histograms and Quantile-Quantile (QQ) plots emerge as indispensable tools, casting light on the underlying patterns and aiding statisticians in their quest for the most suitable models.

Histograms provide a visual snapshot of the data's distribution by binning values into intervals and depicting the frequency of observations in each bin. The resulting bar chart offers an immediate sense of the data's shape, center, and spread.

QQ plots compare the quantiles of the observed data against the quantiles of a theoretical distribution. If the points on the QQ plot align closely with a straight line, it suggests that the data follows the chosen distribution.

Together, histograms and QQ plots form a symbiotic relationship in distribution fitting. While histograms offer an initial visual impression, QQ plots provide a rigorous diagnostic tool for assessing the compatibility of the data with assumed distributions.

Goodness of fit

In the journey of statistical modeling, assessing the goodness of fit is paramount to ensure that chosen models align harmoniously with observed data. Three —Kolmogorov-Smirnov (KS), Cramér-von Mises (CVM), and Anderson-Darling (AD) offer distinct perspectives on the fidelity of models to empirical data.

The KS test quantifies the maximum vertical distance between the cumulative distribution functions (CDFs) of the observed and expected distributions. Larger KS statistics indicate greater divergence, prompting a critical assessment of the goodness of fit.

CVM builds upon the KS test by considering the integral of the squared differences between the observed and expected CDFs. This nuanced approach provides a comprehensive assessment of fit, placing more emphasis on the tails of the distribution.

AD test elevates the scrutiny of tails by assigning greater weight to deviations in these regions. This emphasis on the tails makes it particularly effective for assessing fit in the presence of outliers.

Interpreting the results involves comparing test statistics to critical values at specific significance levels. Lowest value is the best fit.

Fixed part of forecast comes from predictable portion by ARIMA and random portion by distribution. Having identified best distribution, next step is to understand how to use this for simulation and how to use simulation for possible range of error.

Monte Carlo Simulation

At the heart of Monte Carlo simulation lies the art of generating random samples according to assumed distributions. By simulating thousands or even millions of iterations, analysts create several scenarios, each representing a plausible realization of the underlying distribution.

Monte Carlo simulation facilitates parameter estimation by comparing simulated datasets with the observed data. Iteratively adjusting parameters to minimize the disparity between simulated and observed outcomes refines the distribution's fit, enabling a more accurate representation of reality.

Beyond fitting a distribution, Monte Carlo simulation offers a unique lens for quantifying uncertainty. Confidence intervals derived from simulated data shed light on the range of plausible outcomes, empowering analysts with a nuanced understanding of the model's reliability.

Monte Carlo simulation excels in exploring tail behavior, crucial in scenarios where extreme events carry significant consequences. By simulating a vast array of scenarios, analysts can uncover rare but impactful events that might be challenging to capture through deterministic approaches.

Modelling using Machine Learning Model

Machine learning (ML) models typically perform better than statistical models. Also, ensemble ML models produce superior performance than single ML models. In this project, we have deployed two ML models: Random Forest (RF) and Neural Network (NN) models in forecasting the stock price movement.

Random forest involves constructing many decision trees from bootstrap samples from the training dataset, like bagging.

Unlike bagging, random forest also involves selecting a subset of input features (columns or variables) at each split point in the construction of the trees. Typically, constructing a decision tree involves evaluating the value for each input variable in the data in order to select a split point. By reducing the features to a random subset that may be considered at each split point, it forces each decision tree in the ensemble to be more different.

The effect is that the predictions, and in turn, prediction errors, made by each tree in the ensemble are less correlated. When the predictions from these less correlated trees are averaged to make a prediction, it often results in better performance than bagged decision trees.

Neural networks can capture nonlinear and dynamic relationships between variables, which are common in financial markets. Furthermore, they are well-suited for dealing with high-dimensional and noisy data, typical of stock data. Additionally, they can be adapted to changing market conditions by updating their weights with new data.

To use neural networks for stock prediction, we need to have a dataset of historical stock prices and other relevant features, such as volume etc. We then need to train neural network on dataset, using an optimization algorithm to minimize the loss function. Finally, we test neural network on test data, and evaluate its accuracy and performance.

Final Forecast

Time series data forecast consist of two parts.

Part 1 is predictable and given by one out of many models viz. ARIMA, ARIMA+GARCH, Econometric Non-Linear Model, Stochastic Model or Machine Learning models.

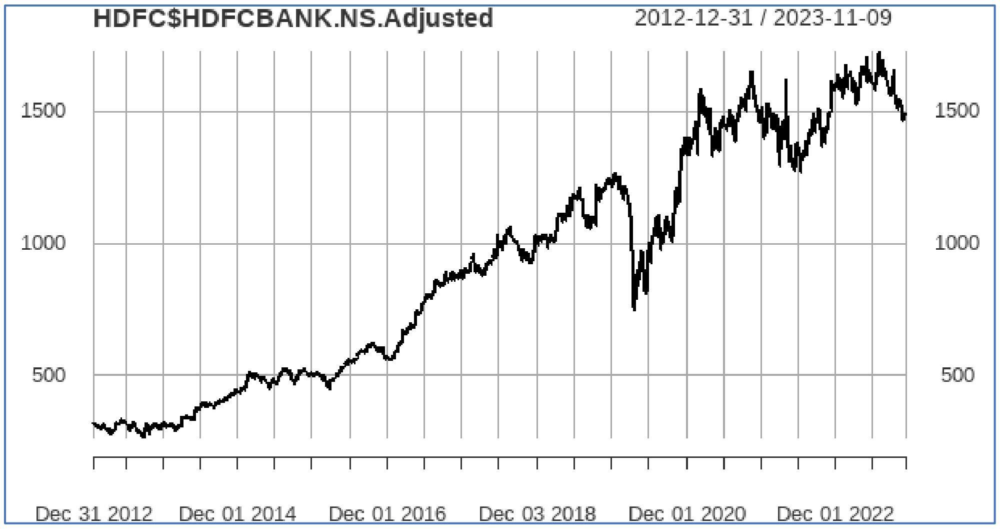
Part 2 is random, and its range is provided with probabilities using Monte Carlo Simulation for the best fitting distribution.

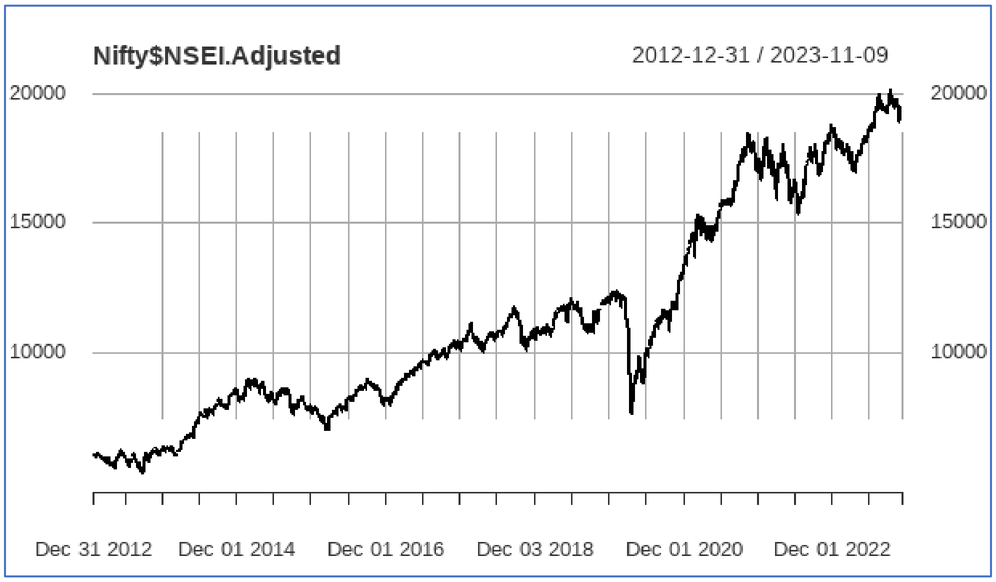
CHAPTER 4: DATA ANALYSIS AND INTERPRETATION

This project has been implemented by comprehensively using R language. The **R code has been provided in Annexure 1**. The comments provided in code are the brief explanations of the steps which have not been reproduced in this chapter to avoid duplication and save number of pages. The following explanations may require the reference of R code for quick understanding.

Visualization of Data

As explained in last chapter, 10 years data has been downloaded from yahoofinance.com (from 31-12-2012 to 10-11-2023) using R. The first step is to observe the trend of daily data as below.





As a part of data cleaning, we need to check if there are any missing values in data. R output of HDFC bank and Nifty is given below.

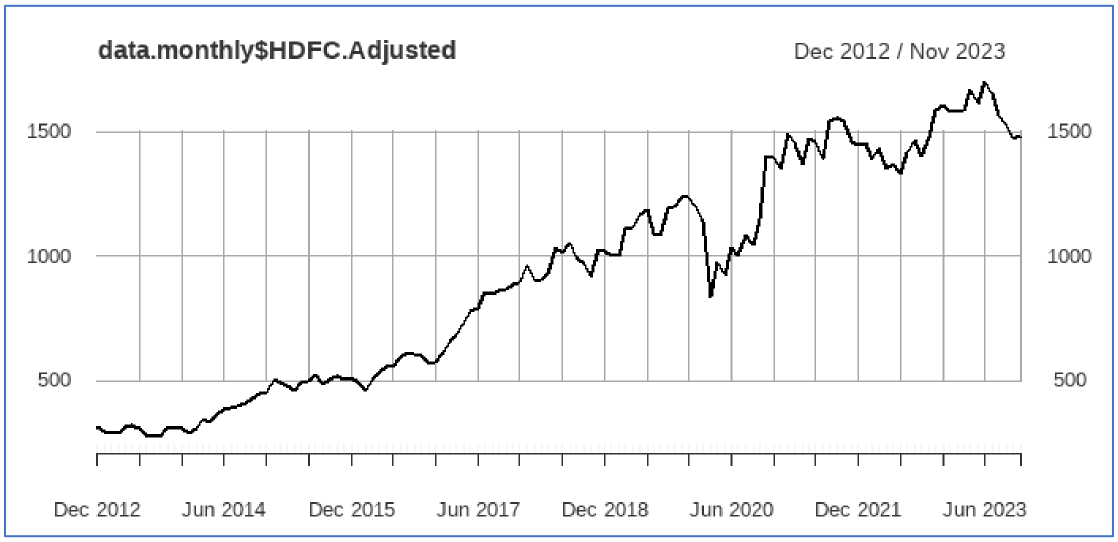
HDFC Bank Summary:

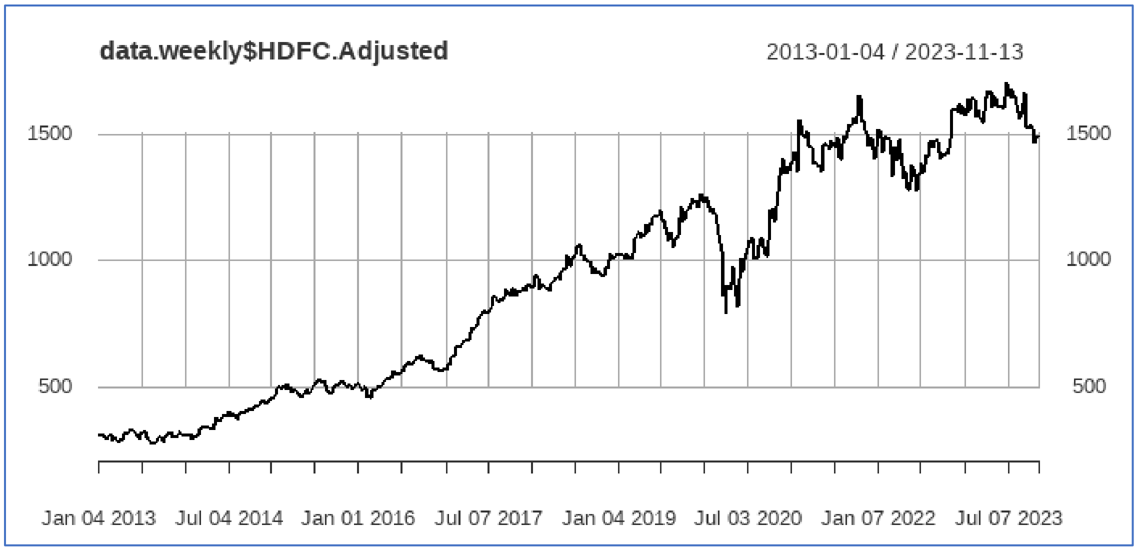
```
    Index              HDFCBANK.NS.Open  HDFCBANK.NS.High  HDFCBANK.NS.Low
 Min.   :2012-12-31   Min.   : 277.9    Min.   : 286.0    Min.   : 264.0
 1st Qu.:2015-09-15   1st Qu.: 529.0    1st Qu.: 532.4    1st Qu.: 523.3
 Median :2018-06-07   Median : 962.5    Median : 971.5    Median : 952.9
 Mean   :2018-06-08   Mean   : 949.4    Mean   : 957.9    Mean   : 940.1
 3rd Qu.:2021-02-25   3rd Qu.:1380.8    3rd Qu.:1398.7    3rd Qu.:1365.1
 Max.   :2023-11-13   Max.   :1723.5    Max.   :1757.5    Max.   :1713.8
                      NA's   :2         NA's   :2         NA's   :2
 HDFCBANK.NS.Close HDFCBANK.NS.Volume  HDFCBANK.NS.Adjusted
 Min.   : 280.9    Min.   :        0   Min.   : 261.2
 1st Qu.: 528.5    1st Qu.:  2920613   1st Qu.: 498.2
 Median : 962.9    Median :  4981200   Median : 926.0
 Mean   : 949.2    Mean   :  6942002   Mean   : 920.0
 3rd Qu.:1381.3    3rd Qu.:  8465556   3rd Qu.:1351.6
 Max.   :1728.2    Max.   :201129980   Max.   :1728.2
 NA's   :2         NA's   :2           NA's   :2
```

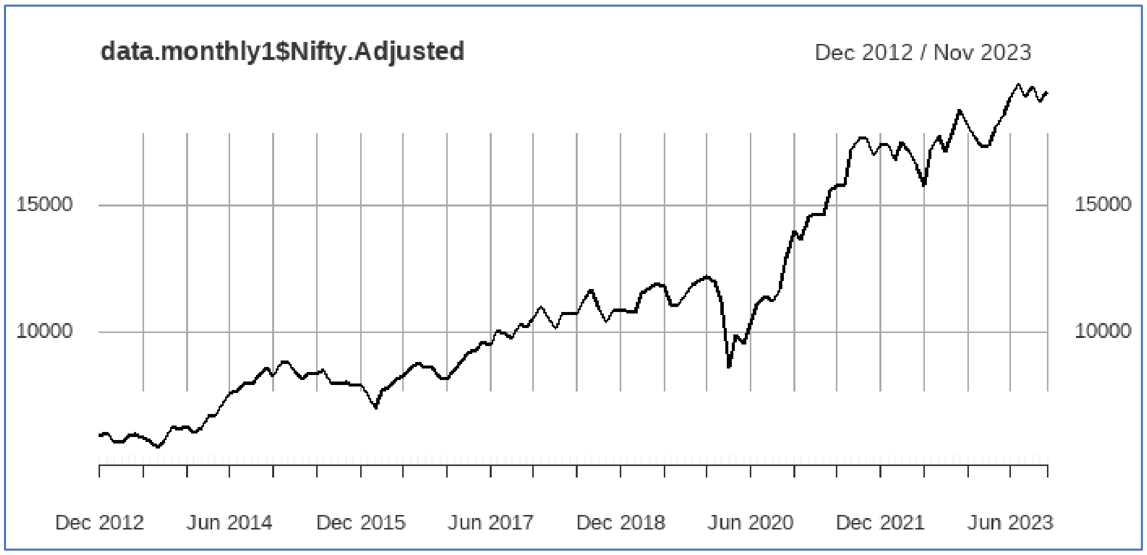
Nifty Summary:

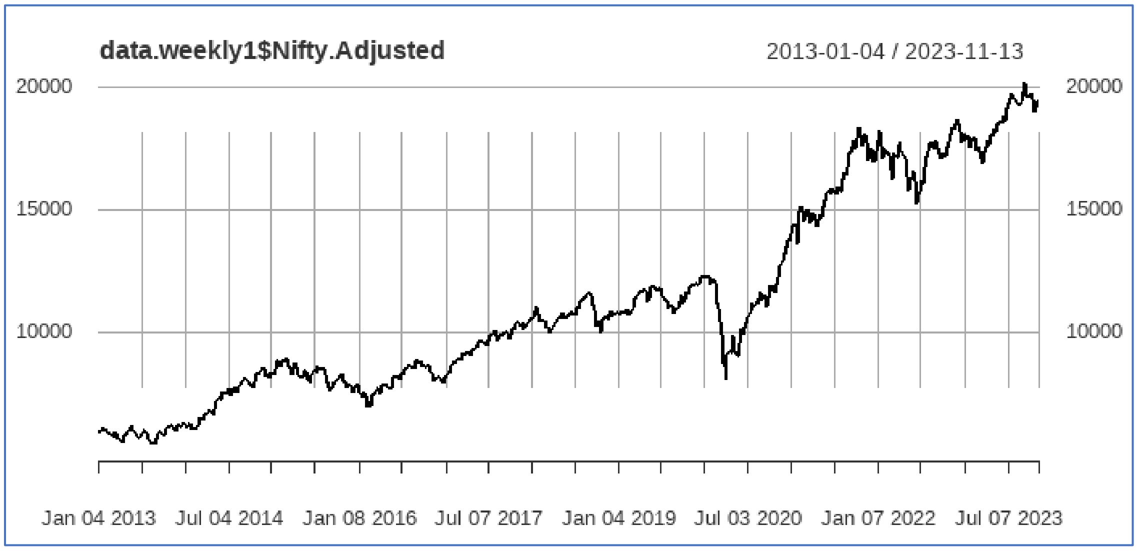
```
    Index               NSEI.Open        NSEI.High         NSEI.Low
 Min.   :2012-12-31   Min.   : 5233   Min.   : 5318   Min.   : 5119
 1st Qu.:2015-09-15   1st Qu.: 8156   1st Qu.: 8197   1st Qu.: 8103
 Median :2018-06-07   Median :10428   Median :10488   Median :10380
 Mean   :2018-06-08   Mean   :11255   Mean   :11308   Mean   :11182
 3rd Qu.:2021-02-25   3rd Qu.:14715   3rd Qu.:14786   3rd Qu.:14553
 Max.   :2023-11-13   Max.   :20156   Max.   :20222   Max.   :20130
                      NA's   :15      NA's   :15      NA's   :15
   NSEI.Close      NSEI.Volume       NSEI.Adjusted
 Min.   : 5285   Min.   :      0   Min.   : 5285
 1st Qu.: 8153   1st Qu.: 166250   1st Qu.: 8153
 Median :10444   Median : 230350   Median :10444
 Mean   :11246   Mean   : 293585   Mean   :11246
 3rd Qu.:14670   3rd Qu.: 346350   3rd Qu.:14670
 Max.   :20192   Max.   :1811000   Max.   :20192
 NA's   :15      NA's   :15        NA's   :15
```

Missing values have been highlighted in red font above. We need to remove them for proceeding further by using R. Having done that, we are converting daily data into weekly and monthly data and plotting their trend. As is evident, weekly and monthly trends being smoother than daily trend.

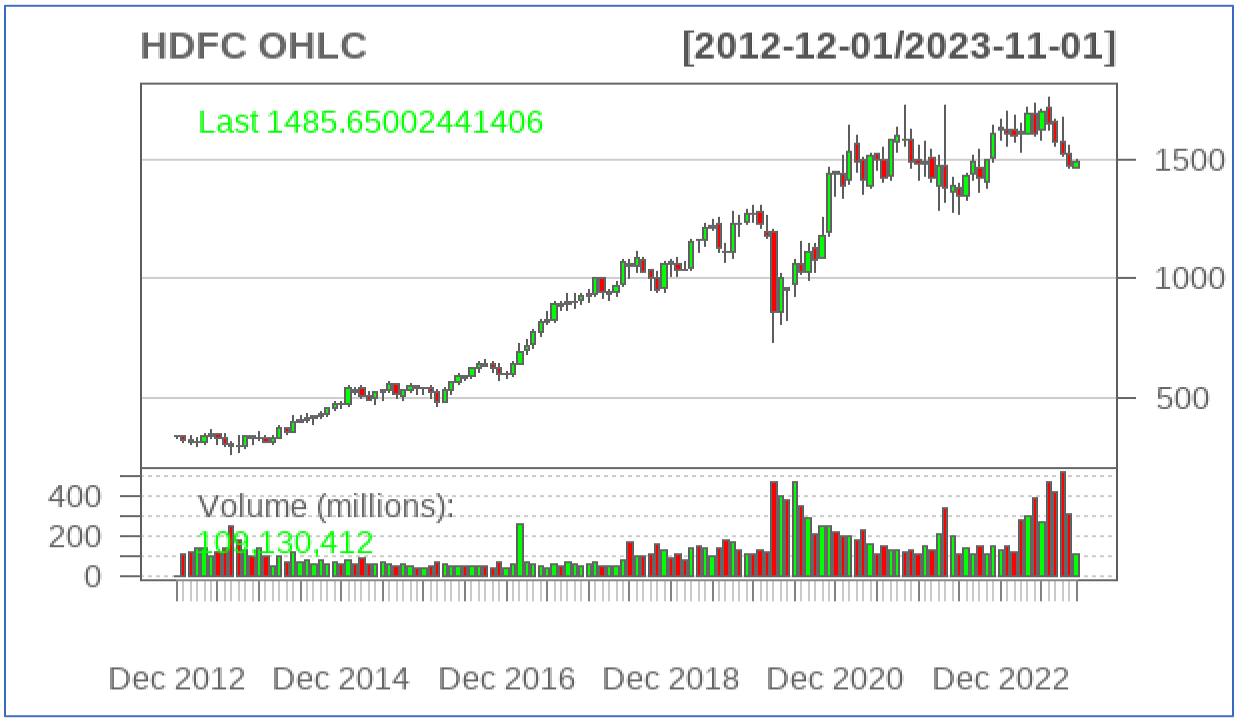



data.weekly$HDFC.Adjusted
2013-01-04 / 2023-11-13
1500
1000
500
Jan 04 2013
Jul 04 2014
Jan 01 2016
Jul 07 2017
Jan 04 2019
Jul 03 2020
Jan 07 2022
Jul 07 2023


data.monthly1$Nifty.Adjusted
Dec 2012 / Nov 2023
15000
10000
Dec 2012
Jun 2014
Dec 2015
Jun 2017
Dec 2018
Jun 2020
Dec 2021
Jun 2023


data.weekly1$Nifty.Adjusted
2013-01-04 / 2023-11-13
20000
15000
10000
Jan 04 2013
Jul 04 2014
Jan 08 2016
Jul 07 2017
Jan 04 2019
Jul 03 2020
Jan 07 2022
Jul 07 2023

The weekly and monthly trends are smoother than the daily chart seen earlier. Let us also see the candle stick chart of HDFC bank along-with volume to see the OHLC patterns.



Financial Returns:

The compounding returns have been calculated for HDFCBANK and Nifty using R code. Top and Tail six returns are given below for HDFC Bank.

```
HDFCBANK.NS.Adjusted
2013-01-01         0.008657076
2013-01-02         0.004154954
2013-01-03        -0.005836621
2013-01-04        -0.005870692
2013-01-07        -0.016548724
```

```
HDFCBANK.NS.Adjusted
2023-11-02         0.001524778
2023-11-03         0.004728940
2023-11-06         0.007219036
2023-11-07        -0.004862926
2023-11-08         0.002887103
```

Similarly, Nifty returns can be plotted. Both can be merged using R to see the returns of both stocks in a tabular format for further analysis.

```
           HDFCBANK.NS.Adjusted NSEI.Adjusted
2013-01-01         0.008657076           NA
2013-01-02         0.004154954  0.014817435
2013-01-03        -0.005836621  0.002707714
2013-01-04        -0.005870692  0.001105953
2013-01-07        -0.016548724 -0.004623255
```

Exploratory Data Analysis (EDA)

EDA is a phenomenon under data analysis used for gaining a better understanding of data aspects. We shall look at various exploratory data analysis methods like:

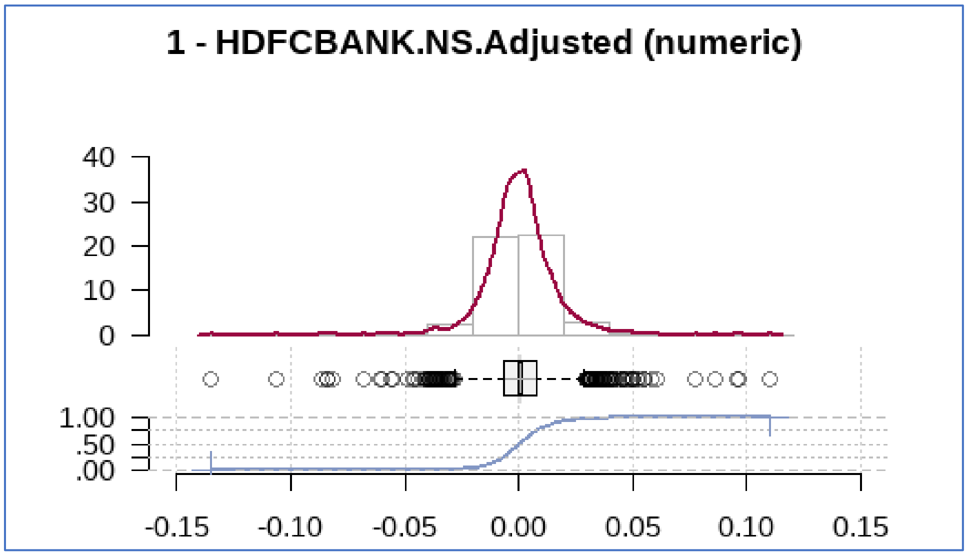
Descriptive Statistics

First is the descriptive statistics of HDFC bank stock returns. Number of data points, quantile values, mean, standard deviation, coefficient of variance, median absolute deviation, IQR, Skewness, Kurtosis, lowest and highest returns etc. have been enumerated below.

1 - HDFCBANK.NS.Adjusted (numeric)

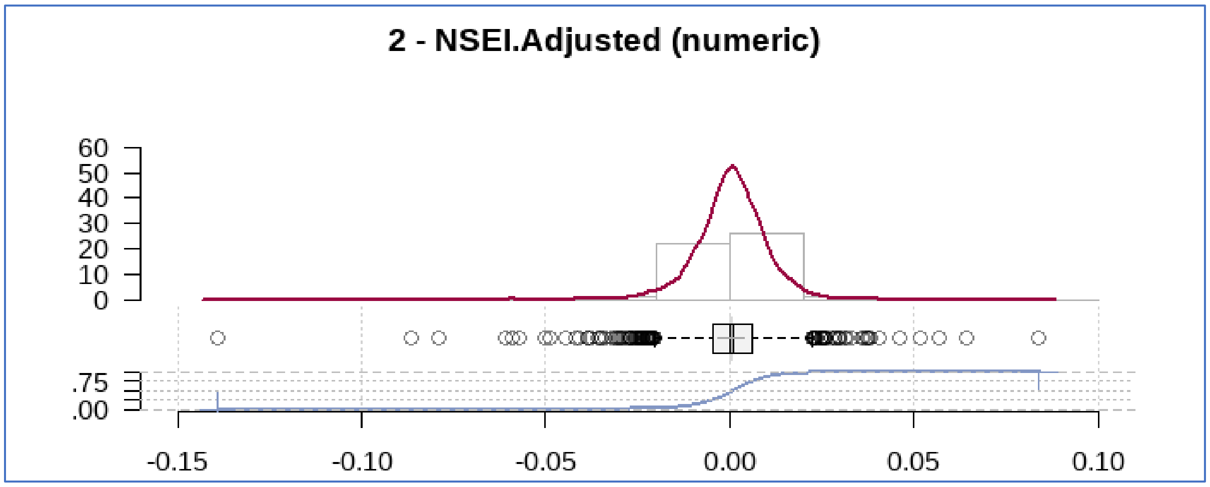
```
        length              n            NAs        unique'
         2'682          2'682              0          2'677
                       100.0%           0.0%

           .05            .10            .25         median
 -0.0201244987  -0.0143924139  -0.0067270505   0.0003623824

         range             sd          vcoef            mad
  0.2445010251   0.0146666752  25.2164584189   0.0106640513

           0s           mean         meanCI
            6   0.0005816310   0.0000263070
         0.2%                  0.0011369551

          .75            .90            .95
 0.0075861333   0.0161253405   0.0231468007

          IQR           skew           kurt
 0.0143131838  -0.1438043110   9.1603756197

lowest : -0.1347538917, -0.1061470680, -0.0867012341, -0.0842640666, -0.08391
highest: 0.0774959966, 0.085723832, 0.0954309087, 0.0962759994, 0.1097471334

' 95%-CI (classic)
```

PDF, box plot and cumulative density function of HDFC Bank returns is given below.



The same statistic and plots for Nifty returns are as given below.

```
2 - NSEI.Adjusted (numeric)

       length              n            NAs         unique             0s'
        2'682          2'669             13          2'667               3
                       99.5%           0.5%                           0.1%

          .05            .10            .25         median             .75
 -0.015436616   -0.010606621   -0.004551338    0.000682437    0.006144586

        range             sd          vcoef            mad             IQR
  0.223040517    0.010609598   23.761885300    0.008013271    0.010695924

         mean         meanCI
  0.000446496    0.000043807
                 0.000849185

          .90            .95
  0.011392760    0.015468858

         skew           kurt
 -1.225024532   18.061066373

lowest : -0.139037565, -0.086668950, -0.079174169, -0.060972600, -0.059160835
highest: 0.04633335, 0.051824726, 0.056691388, 0.064145446, 0.084002952

' 95%-CI (classic)
```

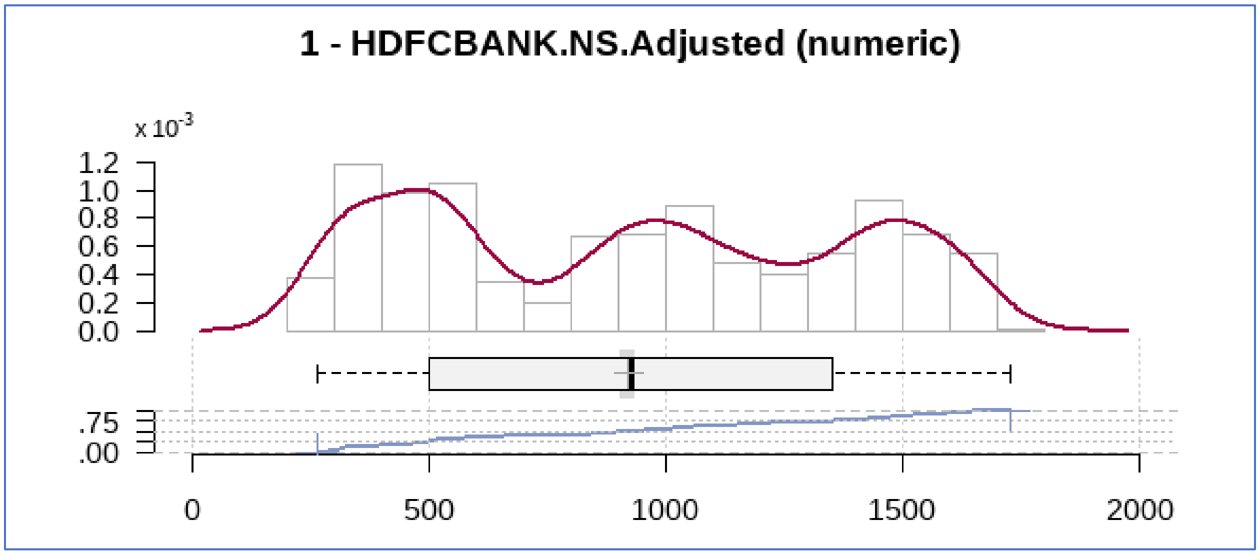


As is evident from above data and graphical representation, skewness for HDFC bank is -0.1438 indicating data is symmetric which is also validated with symmetric shape of data in PDF plot. But kurtosis is 9.16 which is indicating presence of extreme values and **absence of normality of data**. The box plot above also shows presence of many outliers confirming the same.

The same conclusions hold good for Nifty returns except that Nifty is less symmetric and having higher kurtosis than HDFC Bank.

Now we are moving into EDA of stock prices. Let us have description of data for HDFC stock price and Nifty price respectively.

```
1 - HDFCBANK.NS.Adjusted (numeric)

      length          n        NAs     unique         0s        mean'
       2'683      2'683          0      2'613          0    920.0260
                 100.0%       0.0%                  0.0%


    .05        .10        .25     median        .75         .90
   304.1818   319.3028   498.2279   926.0388   1'351.5505   1'535.7500

      range         sd      vcoef        mad         IQR        skew
  1'466.9699   443.8763     0.4825   633.7545    853.3226      0.1139

     meanCI
   903.2227
   936.8294

        .95
  1'607.4719

       kurt
    -1.3608


lowest : 261.2301, 261.2534, 261.5323, 262.2296, 265.9489
highest: 1'696.60, 1'701.40, 1'708.21, 1'719.80, 1'728.20
' 95%-CI (classic)
```

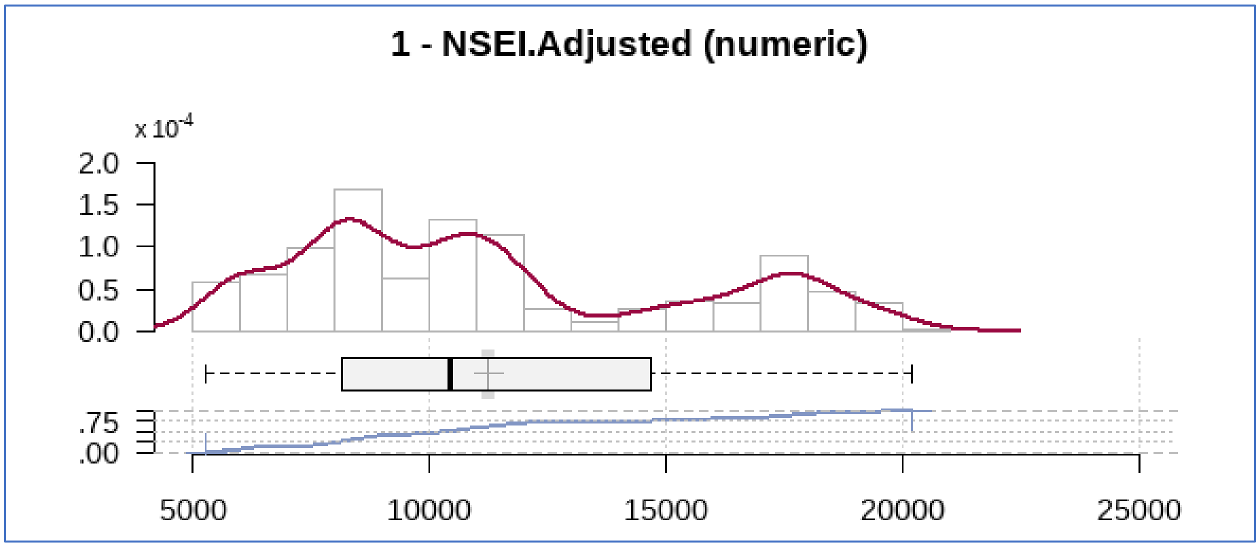


It is evident from above data and plots that skewness is less than 0.5 indicating symmetricity, and kurtosis is also significantly lesser than HDFC returns data.

When we do the same exercise for Nifty price, we get skewness and kurtosis both significantly improved towards normality vis-à-vis their returns data.

Thus, it can be said that absolute prices are less away from normality of data as compared to their compounded returns data.

```
1 - NSEI.Adjusted (numeric)

      length           n         NAs      unique          0s        mean'
       2'670       2'670           0       2'647           0  11'246.272
                  100.0%        0.0%                    0.0%

         .05         .10         .25      median         .75         .90
   5'948.220   6'261.475   8'152.912  10'443.700  14'670.038  17'787.156

       range          sd       vcoef         mad         IQR        skew
  14'907.350   4'057.989       0.361   3'619.546   6'517.125       0.614

      meanCI
  11'092.279
  11'400.265

         .95
  18'419.685

        kurt
      -0.835

lowest : 5'285.0, 5'287.450, 5'302.550, 5'341.450, 5'401.450
highest: 19'996.350, 20'070.0, 20'103.100, 20'133.301, 20'192.350

' 95%-CI (classic)
```



Correlation

The correlation coefficient between HDFC Bank and Nifty returns is given below. Spearman coefficient is 0.663 as shown below.

	var1	var2	coef_corr
	<fct>	<fct>	<dbl>
1	NSEI.Adjusted	HDFCBANK.NS.Adjusted	0.663
2	HDFCBANK.NS.Adjusted	NSEI.Adjusted	0.663

Kendall coefficient is only 0.4886 as seen below.

```
     var1               var2               coef_corr
  <fct>              <fct>              <dbl>
1 NSEI.Adjusted      HDFCBANK.NS.Adjusted  0.486
2 HDFCBANK.NS.Adjusted NSEI.Adjusted       0.486
```

Thus, both coefficients confirm absence of strong correlation between HDFC and Nifty returns.

Normality Tests

Multiple tests have been used to validate if the returns data of HDFC Bank and Nifty is normal.

Jarque - Bera Normality Test:
Test Results:
STATISTIC: X-squared: 9269.2776,
P VALUE: Asymptotic p Value: < 2.2e-16

Here p value is much lesser than 0.05 hence Null hypothesis that data is normal is rejected.

Kolmogorov-Smirnov test
Test Results:
STATISTIC: D: 0.4777
P VALUE:
Alternative Two-Sided: < 2.2e-16
Alternative Less: < 2.2e-16
Alternative Greater: < 2.2e-16

Here p value is much lesser than 0.05 hence Null hypothesis that data is normal is rejected.

Shapiro - Wilk Normality Test
Test Results:
STATISTIC: W: 0.9181
P VALUE: < 2.2e-16

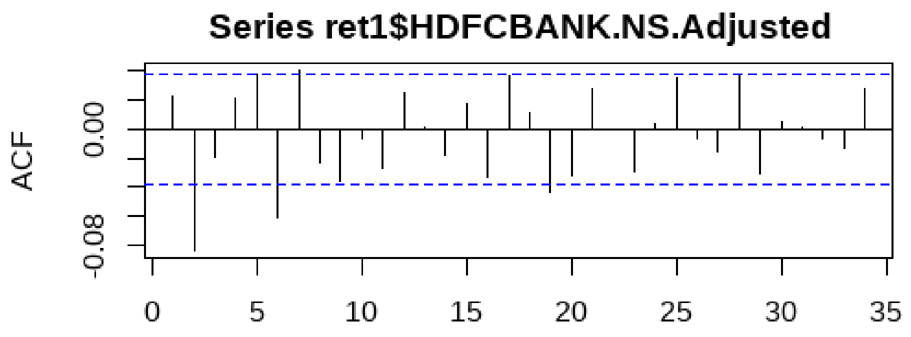
Here p value is much lesser than 0.05 hence Null hypothesis that data is normal is rejected.

Thus, all tests suggest that returns data is not following normal distribution. We conducted the same tests for Nifty also (included in code) and that also suggest the similar lack of normality in Nifty returns data. Likewise, we performed the same normality tests for HDFC Bank and Nifty prices and that also gave the same result. Thus, all the data sets, namely HDFC Bank and Nifty returns and prices considered in this project do not exhibit normality of data.

Autocorrelation – HDFC Bank Returns

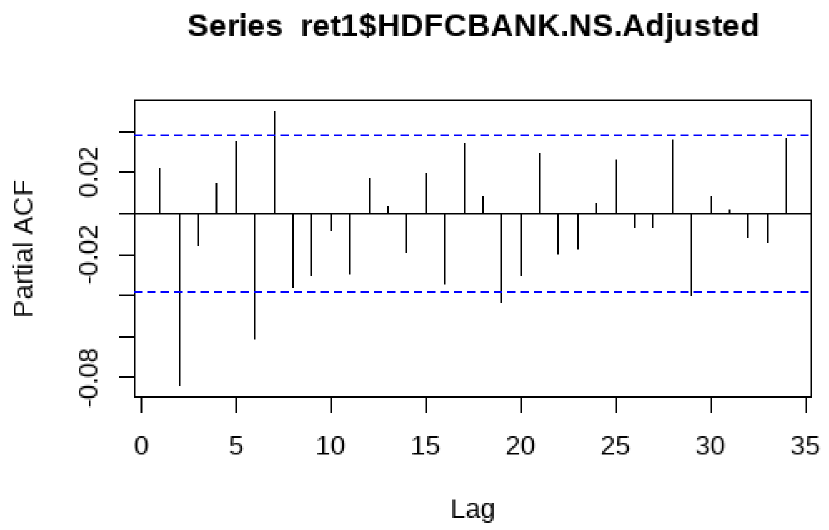
Autocorrelation plays a pivotal role in ARIMA modeling by aiding in model identification, validation, and enhancing the accuracy of time series forecasting. It allows us to capture the temporal dependencies within the data, enabling better insights and predictions.

ACF plot for HDFC Bank is given below. As seen in the plot, correlation is statistically significant at lags 2, 6, 7, 19 where values are crossing the blue lines.



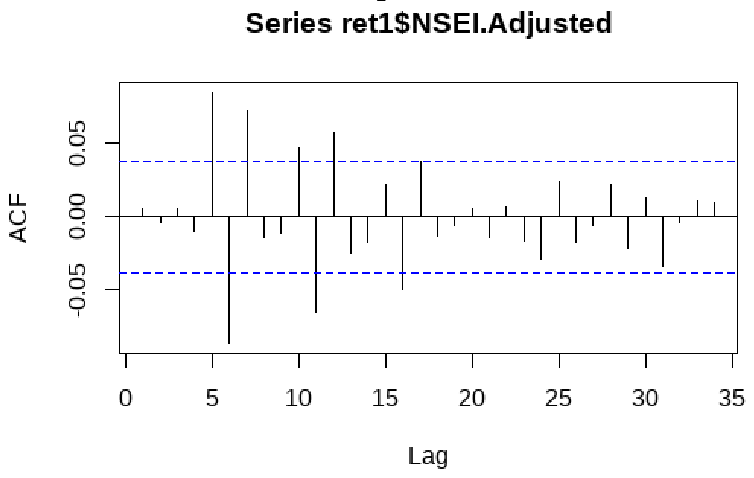
Partial Autocorrelation -HDFC Bank Returns

Here we are trying to identify if these relationships are mutually exclusive. As seen in HDFC PACF plot, partial correlation is statistically significant at lags 2, 6,7.



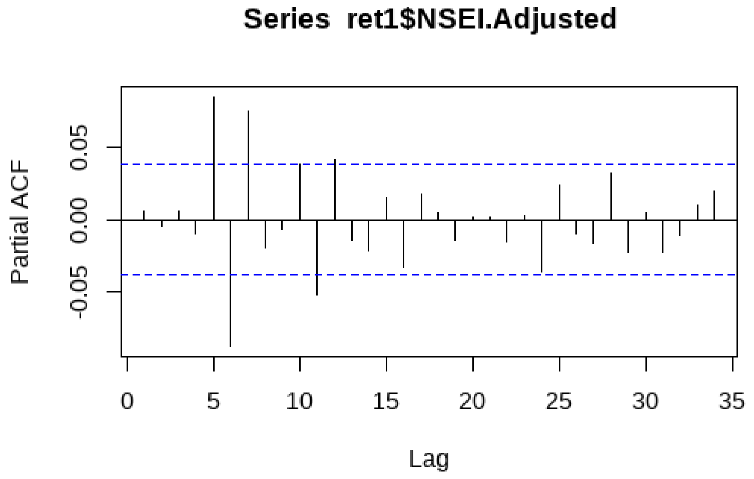
Autocorrelation – Nifty Returns

ACF plot for Nifty is given below. As seen in the plot, correlation is statistically significant at lags 5,6,7,10,11,12,16 where values are crossing the blue lines.



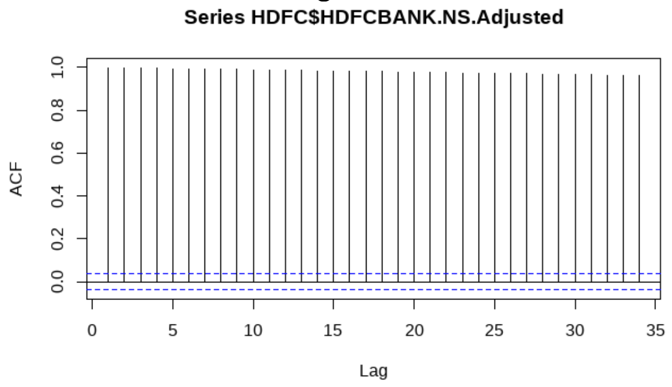
Partial Autocorrelation -Nifty Returns

As seen in HDFC PACF plot, partial correlation is statistically significant at lags 5, 6,7, 11,12



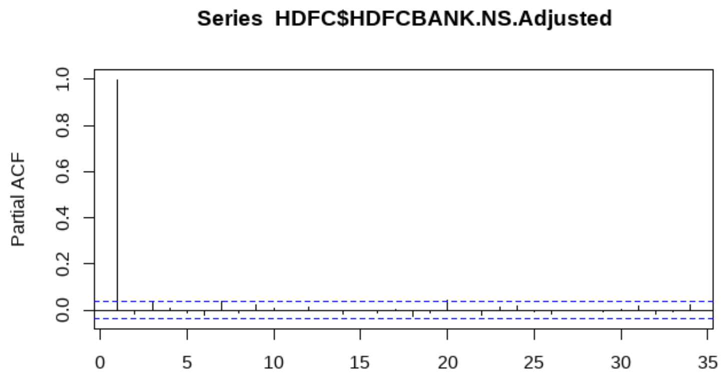
Autocorrelation – HDFC Bank Stock Price

ACF plot for HDFC Bank Stock price is given below. As seen in the plot, correlation is statistically significant at all lags where values are crossing the blue lines.



Partial Autocorrelation -HDFC Bank Stock Price

As seen in HDFC PACF plot, partial correlation is statistically significant at lags 1.



Thus, correlation as well as partial correlation are present in returns data of HDFC Bank and Nifty returns. Similarly, we have verified the correlation for prices of HDFC Bank and Nifty and they also exhibit the correlation at different lags. Therefore, we are good to go for modelling using ARIMA. Alternatively, same is being verified using statistical tests.

Ljung-Box Test for Auto Correlation

Box-Ljung test for HDFC Bank Return:
```
data:  ret1$HDFCBANK.NS.Adjusted
X-squared = 79.269, df = 30, p-value = 2.515e-06
```
Box-Ljung test for Nifty Return:
```
data:  ret1$NSEI.Adjusted
X-squared = 104.67, df = 30, p-value = 3.357e-10
```
Box-Ljung test for HDFC Bank Price:
```
data:  HDFC$HDFCBANK.NS.Adjusted
X-squared = 78182, df = 30, p-value < 2.2e-16
```
Box-Ljung test for Nifty Price:
```
data:  Nifty$NSEI.Adjusted
X-squared = 76867, df = 30, p-value < 2.2e-16
```

For both, HDFC Bank & Nifty, p value is less than 0.05 meaning alternative hypothesis is accepted i.e., auto correlation is present. We also ran the code for HDFC and Nifty prices and found that the same conclusion holds good. This is also the validation of ACF and PACF plots above.

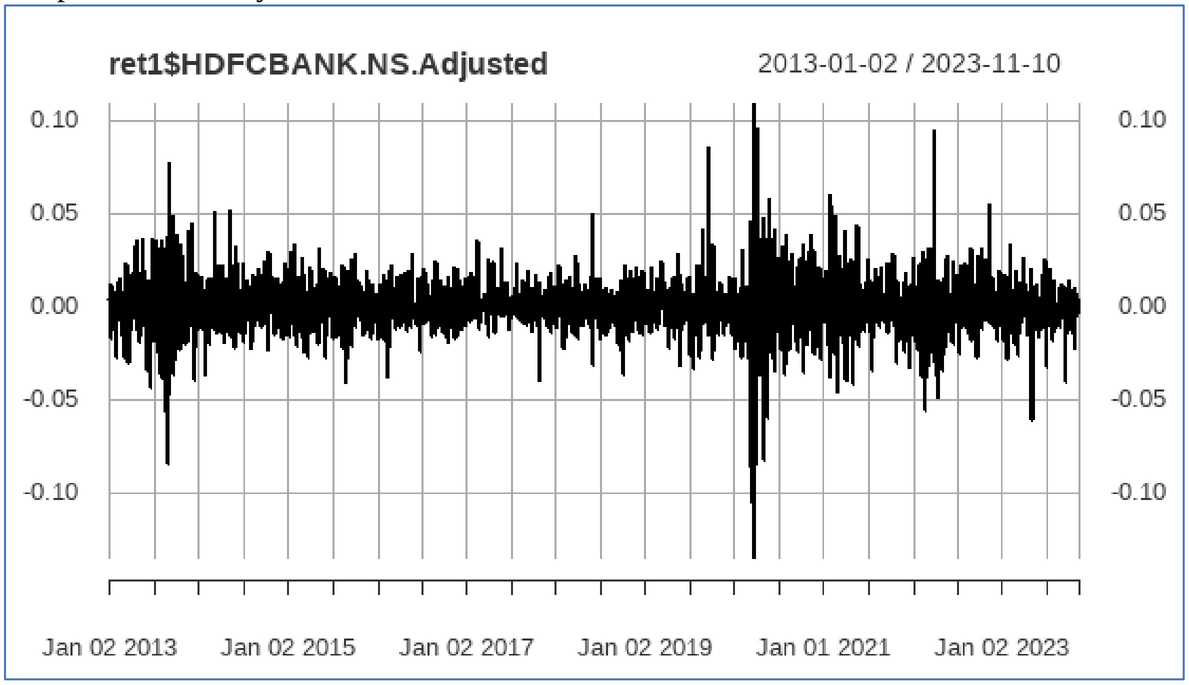
Model Building – AUTO ARIMA:

Using auto ARIMA model for HDFC Bank returns as below, we get best model: ARIMA (2,0,2)

```
Fitting models using approximations to speed things up...
 ARIMA (2,0,2) with non-zero mean: -14957.38
 ARIMA (0,0,0) with non-zero mean: -14944.18
 ARIMA (1,0,0) with non-zero mean: -14942.51
 ARIMA (0,0,1) with non-zero mean: -14943.7
 ARIMA (0,0,0) with zero mean     : -14942.16
 ARIMA (1,0,2) with non-zero mean: -14956.83
 ARIMA (2,0,1) with non-zero mean: -14956.95
 ARIMA (3,0,2) with non-zero mean: -14955.02
 ARIMA (2,0,3) with non-zero mean: -14956.53
 ARIMA (1,0,1) with non-zero mean: -14954.94
 ARIMA (1,0,3) with non-zero mean: -14955
 ARIMA (3,0,1) with non-zero mean: -14954.25
 ARIMA (3,0,3) with non-zero mean: Inf
 ARIMA (2,0,2) with zero mean     : -14955.07
 Now re-fitting the best model(s) without approximations...
 ARIMA (2,0,2) with non-zero mean: -14959.49
```

Best model: ARIMA (2,0,2) with non-zero mean

Let us plot HDFC daily returns data as below.



Similarly, using auto ARIMA model for Nifty, we are getting Best model: ARIMA (0,0,0)

Fitting models using approximations to speed things up...

```
ARIMA(2,0,2) with non-zero mean : -16673.95
ARIMA(0,0,0) with non-zero mean : -16681.98
ARIMA(1,0,0) with non-zero mean : -16680.89
ARIMA(0,0,1) with non-zero mean : -16680.05
ARIMA(0,0,0) with zero mean     : -16679.26
ARIMA(1,0,1) with non-zero mean : -16678.88

Now re-fitting the best model(s) without approximations...

ARIMA(0,0,0) with non-zero mean : -16681.98
```

Best model: ARIMA (0,0,0) with non-zero mean

For HDFC Bank price:

Best model is ARIMA (2,1,2) with drift.

For Nifty price:

Best model is ARIMA (0,1,0) with drift.

Model Summary

Now we will check fitted model details using summary function.

Series: ret1$HDFCBANK.NS.Adjusted ARIMA(2,0,2) with non-zero mean

Coefficients:

```
         ar1      ar2      ma1     ma2    mean

      0.0305  -0.4913  -0.0042  0.4138   6e-04
s.e.  0.2313   0.2120   0.2402  0.2266   3e-04

sigma^2 = 0.0002144:  log likelihood = 7485.76

AIC=-14959.52   AICc=-14959.49   BIC=-14924.19
```

Training set error measures:

```
                        ME       RMSE        MAE MPE MAPE      MASE      ACF1

Training set -3.298691e-06 0.01462972 0.01009825 NaN  Inf 0.6926625 -0.003528
```

RMSE in isolation does not convey much. Therefore, let us calculate the ration of RMSE/Predicted values to understand the error in prediction and usefulness of model. That requires predicted values for calculation. 20 predicted values for HDFC Bank returns using our model are as given below.

$pred

```
Time Series:
Start = 2669
End = 2688
Frequency = 1

[1] 0.0007279137 0.0003392462 0.0004912450 0.0006868359 0.0006181145 0.000519
[7] 0.0005506959 0.0005998768 0.0005862558 0.0005616773 0.0005676207 0.000579
[13] 0.0005773310 0.0005712313 0.0005722967 0.0005753261 0.0005748949 0.00057
[19] 0.0005735595 0.0005743023
```

RMSE/Predicted value for HDFC returns is 0.01462972/0.0002736028=53.5.

Similarly,

RMSE/Predicted value for Nifty returns is 0.01060959/0.0004463128=23.8
RMSE/Predicted value for HDFC price is 14.68772/1492.464 =0.009841256
RMSE/Predicted value for Nifty price is 116.2637/19430.42= 0.005983592

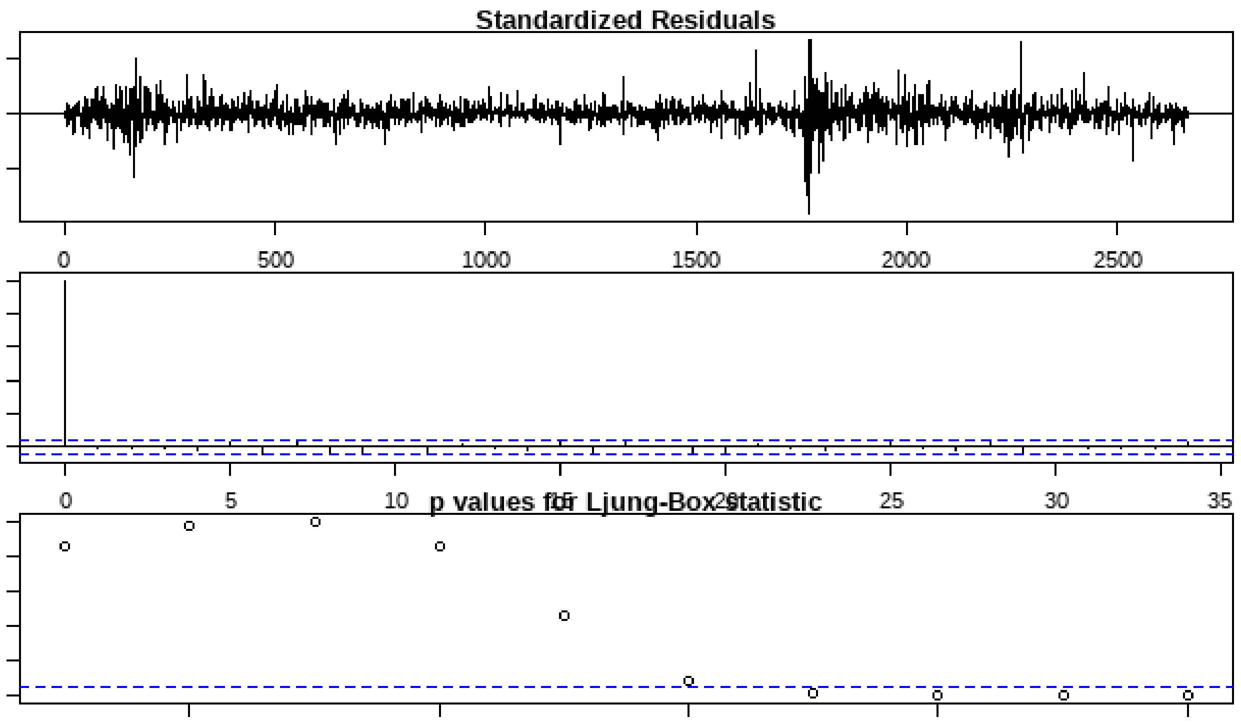
For HDFC and Nifty reruns, error is very high implying that model accuracy is very poor, and the predictions are not reliable.

The error ratio for absolute prices of HDFC stock and Nifty 50 are reasonable.

Residuals and Squared Residuals Analysis

After fitting ARIMA, error is also called residual. We shall check if there are serial correlations i n residuals or squared residuals. In such case we can still progress with forecasting using ARIMA -GARCH family of models.

The residual plot, ACF function and p values corresponding to auto correlation function are given below. Since p values are below blue lines on right hand side, it means p values are less than 0.05 and there is serial correlation present in the residuals.



Let us also test correlation statistically for residuals & squared residuals.

Box-Ljung test for residuals:

data: model1$residuals
X-squared = 43.853, df = 20, p-value = 0.001575

Box-Ljung test for squared residuals:

data: model1$residuals^2
X-squared = 1805.5, df = 20, p-value < 2.2e-16

We get p value less than 0.05 for residuals and squared residuals as above. The same is the case f or all other models also.

P value < 0.05 above is confirming that there is correlation in residuals (H1 is accepted) as well as in squared residuals. Thus, there is scope for exploration of ARIMA.

But our ARIMA model has not been able to capture this in above model. The reason is obviously non normality of data as we have seen earlier. This calls for Transformation of data.

Power Transformations:

Auto ARIMA takes lambda as input and calculates the optimum value for conversion of data into Normal. After using lambda=auto in auto ARIMA. We have got following results.

HDFC Bank returns

Best model: ARIMA (2,0,2) with non-zero mean

Model Summary:

Series: ret1$HDFCBANK.NS.Adjusted
ARIMA (2,0,2) with non-zero mean

Box Cox transformation: **lambda= 1**

Coefficients:

```
         ar1      ar2      ma1     ma2     mean
      0.0305  -0.4913  -0.0042  0.4138  -0.9994
s.e.  0.2313   0.2120   0.2402  0.2266   0.0003

sigma^2 = 0.0002144:  log likelihood = 7485.76
AIC=-14959.53   AICc=-14959.5   BIC=-14924.19
```

Training set error measures:

```
                        ME       RMSE         MAE  MPE  MAPE      MASE       ACF1
Training set -3.298551e-06 0.01462972 0.01009825  NaN   Inf 0.6926625 -0.003528
```

Predicted values are as given below.

Point	Forecast	Lo 80	Hi 80	Lo 95	Hi 95
2670	**0.0002736233**	-0.01848929	0.01903653	-0.02842177	0.02896902
2671	0.0006877980	-0.01808152	0.01945712	-0.02801740	0.02939299
2672	0.0007263979	-0.01809799	0.01955078	-0.02806301	0.02951581
2673	0.0005231370	-0.01830340	0.01934967	-0.02826956	0.02931583
2674	0.0004980784	-0.01834153	0.01933768	-0.02831461	0.02931076
2675	0.0005976502	-0.01824264	0.01943794	-0.02821609	0.02941139
2676	0.0006129607	-0.01823043	0.01945635	-0.02820551	0.02943143
2677	0.0005642730	-0.01827933	0.01940787	-0.02825453	0.02938307
2678	0.0005552777	-0.01828905	0.01939961	-0.02826464	0.02937519
2679	0.0005790399	-0.01826536	0.01942344	-0.02824098	0.02939905

Hence RMSE/Predicted value for HDFC= 0.01462706/0.0002736233=53.45

Nifty Returns:

Best model: ARIMA (1,0,0) with non-zero mean
Model Summary:
Series: ret1$NSEI.Adjusted
ARIMA (1,0,0) with non-zero mean
Box Cox transformation: **lambda= 0.8584132**

Coefficients:
```
         ar1     mean
      0.0328  -1.1638
s.e.  0.0194   0.0004

sigma^2 = 0.0004611:  log likelihood = 6462.8
AIC=-12919.61   AICc=-12919.6   BIC=-12901.94
```

Training set error measures:
```
                       ME       RMSE         MAE  MPE MAPE      MASE      ACF1
Training set 0.0001297906 0.01060887 0.007239456 -Inf  Inf 0.7229826 -0.01512
```

Predicted values are:

```
Point   Forecast           Lo 80          Hi 80      Lo 95          Hi 95
2670    0.0003080859 -0.01197246 0.01330405 -0.02023830 0.02168254
2671    0.0002877257 -0.01201905 0.01327652 -0.02029458 0.02165843
2672    0.0002869940 -0.01202039 0.01327518 -0.02029605 0.02165698
2673    0.0002869675 -0.01202044 0.01327513 -0.02029610 0.02165693
2674    0.0002869665 -0.01202044 0.01327513 -0.02029611 0.02165693
2675    0.0002869665 -0.01202044 0.01327513 -0.02029611 0.02165693
2676    0.0002869665 -0.01202044 0.01327513 -0.02029611 0.02165693
2677    0.0002869665 -0.01202044 0.01327513 -0.02029611 0.02165693
2678    0.0002869665 -0.01202044 0.01327513 -0.02029611 0.02165693
2679    0.0002869665 -0.01202044 0.01327513 -0.02029611 0.02165693
```

Hence RMSE/Predicted value for HDFC = 0.010608870.0003080859 = 34.43

HDFC Bank Price:

Best model: ARIMA (2,1,2) with drift

Model Summary:
```
Series: HDFC$HDFCBANK.NS.Adjusted
ARIMA(2,1,2) with drift
Box Cox transformation: lambda= 0.004546487
```

Coefficients:
```
         ar1      ar2      ma1     ma2  drift
      0.0536  -0.4855  -0.0266  0.4073  6e-04
s.e.  0.2418   0.2224   0.2507  0.2374  3e-04

sigma^2 = 0.0002269:  log likelihood = 7449.15
AIC=-14886.3   AICc=-14886.26   BIC=-14850.93
```

Training set error measures:
```
                      ME     RMSE      MAE          MPE     MAPE      MASE
Training set -0.08864405 14.76288 9.216258 -0.009728495 1.007229 0.9964984
                   ACF1
Training set 0.01383202
```

Predicted Values:

```
Point   Forecast     Lo 80    Hi 80    Lo 95    Hi 95
2684       1489.190 1461.637 1517.260 1447.257 1532.332
2685       1490.194 1450.831 1530.620 1430.414 1552.460
2686       1491.281 1444.144 1539.949 1419.795 1566.350
2687       1492.075 1438.536 1547.597 1410.973 1577.816
2688       1492.814 1433.160 1554.939 1402.548 1588.861
```

```
2689         1493.692 1428.398 1561.957 1394.991 1599.343
2690         1494.606 1424.306 1568.360 1388.434 1608.857
2691         1495.455 1420.552 1574.288 1382.425 1617.680
2692         1496.283 1416.986 1579.996 1376.717 1626.182
2693         1497.143 1413.662 1585.529 1371.363 1634.402
```

RMSE/Predicted value = 14.76288/1489.190 = 0.009913362

Nifty Price:

Best model: ARIMA (0,1,0) with drift

Model Summary:

```
Series: Nifty$NSEI.Adjusted
ARIMA(0,1,0) with drift
Box Cox transformation: lambda= 0.3930239

Coefficients:
       drift
      0.0173
s.e.  0.0077

sigma^2 = 0.1597:  log likelihood = -1338.58
AIC=2681.15   AICc=2681.15   BIC=2692.93
```

Training set error measures:

```
                    ME     RMSE     MAE          MPE      MAPE      MASE
Training set 0.1803678 116.2459 79.3977 -0.006669508 0.7253278 0.9966362
                   ACF1
Training set 0.02336036
```

Predicted Values:

```
Point        Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2671         19450.48 19245.71 19656.58 19137.84 19766.21
2672         19457.42 19168.14 19749.33 19016.07 19904.92
2673         19464.35 19110.35 19822.31 18924.55 20013.40
2674         19471.29 19062.79 19885.06 18848.66 20106.24
2675         19478.23 19021.76 19941.29 18782.77 20189.10
2676         19485.17 18985.36 19992.88 18723.96 20264.86
2677         19492.11 18952.48 20040.96 18670.53 20335.26
2678         19499.05 18922.38 20086.27 18621.35 20401.41
2679         19506.00 18894.55 20129.31 18575.64 20464.10
2680         19512.94 18868.61 20170.45 18532.83 20523.89
```

RMSE / Predicted value = 116.2459/19450.48 = 0.0059765

Thus, we see that lambda values have changed but errors have not improved significantly.

Clearly, ARIMA model has not improved the results for above models.

Out of sample Forecasting

We are dividing our data into training data and test data. We are now using two ARIMA models i.e. one with BoxCox transformation and another without it.

ARIMA with BoX Cox:

RMSE=0.9986191

ARIMA without BoX Cox:

RMSE = 0.009903939

This is a strange result, where error is less without data transformation for non-normal data.

Fitting Distributions:

We are now working with residuals of **HDFC Bank reruns** for fitting of best distribution.

Fitting models using approximations to speed things up...

```
 ARIMA(2,0,2) with non-zero mean : -14963.88
ARIMA(0,0,0) with non-zero mean : -14950.66
ARIMA(1,0,0) with non-zero mean : -14948.99
ARIMA(0,0,1) with non-zero mean : -14950.19
ARIMA(0,0,0) with zero mean     : -14948.61
ARIMA(1,0,2) with non-zero mean : -14963.34
ARIMA(2,0,1) with non-zero mean : -14963.45
ARIMA(3,0,2) with non-zero mean : -14961.51
ARIMA(2,0,3) with non-zero mean : -14963.03
ARIMA(1,0,1) with non-zero mean : -14961.46
ARIMA(1,0,3) with non-zero mean : -14961.51
ARIMA(3,0,1) with non-zero mean : -14960.75
ARIMA(3,0,3) with non-zero mean : Inf
ARIMA(2,0,2) with zero mean     : -14961.54
```

Now re-fitting the best model(s) without approximations...

```
ARIMA(2,0,2) with non-zero mean : -14965.99
```

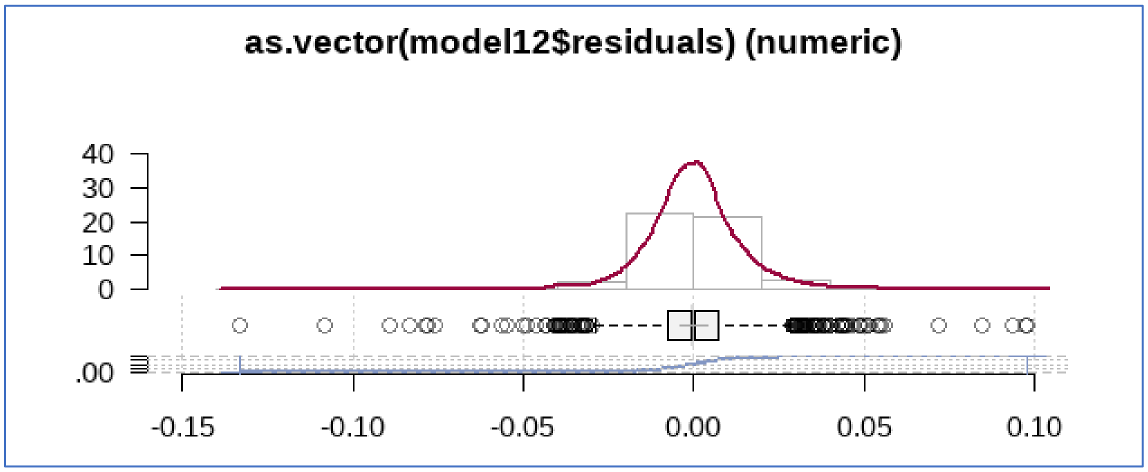
Best model: ARIMA (2,0,2) with non-zero mean

The descriptive statistics of residuals is given below.

```
as.vector(model14$residuals) (numeric)

      length          n           NAs      unique        0s'
      2'668       2'668             0         = n          0
                 100.0%          0.0%                    0.0%
```

```
              .05             .10                  .25          median          .75
   -0.0208136184   -0.0153359166        -0.0074771010   -0.0002293406 0.0069693125

           range              sd                vcoef             mad          IQR
    0.2308836743    0.0146324639    -4'462.7024903760    0.0106800734 0.0144464134

            mean          meanCI
   -0.0000032788   -0.0005587606
                    0.0005522029

             .90             .95
    0.0155448702    0.0227973480

            skew            kurt
   -0.2011449575    8.6283918291

lowest: -0.1329970272, -0.1084111568, -0.0893172487, -0.0833657663, -0.078461
highest: 0.0715686607, 0.0842763449, 0.093529397, 0.0973988082, 0.0978866471

' 95%-CI (classic)
```

Below are also given the pdf, cum pdf which are validating the same conclusions.



Skewness is close to zero, but kurtosis is very high. Hence normal distribution may not be appro priate fit. Therefore, we shall fit 4 different distributions here to select the best fit.
The summary of normal, logistic, Cauchy and t distributions.is given below.

summary(normaldist)
Fitting of the distribution ' norm ' by maximum likelihood
Parameters:

	estimate	Std. Error
mean	-5.533844e-07	0.0002831327
sd	1.462730e-02	0.0001960104

Loglikelihood: 7489.019 AIC: -14974.04 BIC: -14962.26
Correlation matrix:

	mean	sd
mean	1	0
sd	0	1

summary(logisdist)

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
              estimate   Std. Error
location -0.0001203605 0.0002423075
scale     0.0073383519 0.0001134377
```
Loglikelihood: 7716.864 AIC: -15429.73 BIC: -15417.95
```
Correlation matrix:
            location       scale
location 1.000000000 0.009069143
scale    0.009069143 1.000000000
```

summary(cauchydist)

```
Fitting of the distribution ' cauchy ' by maximum likelihood
Parameters :
              estimate   Std. Error
location -0.0003201685 0.0002046201
scale     0.0067499918 0.0001639312
```
Loglikelihood: 7534.696 AIC: -15065.39 BIC: -15053.61

```
Correlation matrix:
            location       scale
location 1.000000000 0.004006873
scale    0.004006873 1.000000000
```

summary(tdist)

```
Fitting of the distribution ' t.scaled ' by maximum likelihood
Parameters :
          estimate   Std. Error
df    3.4170956047 0.2458140333
mean -0.0002139282 0.0002252561
sd    0.0096409246 0.0002345408
```
Loglikelihood: 7774.942 AIC: -15543.88 BIC: -15526.22

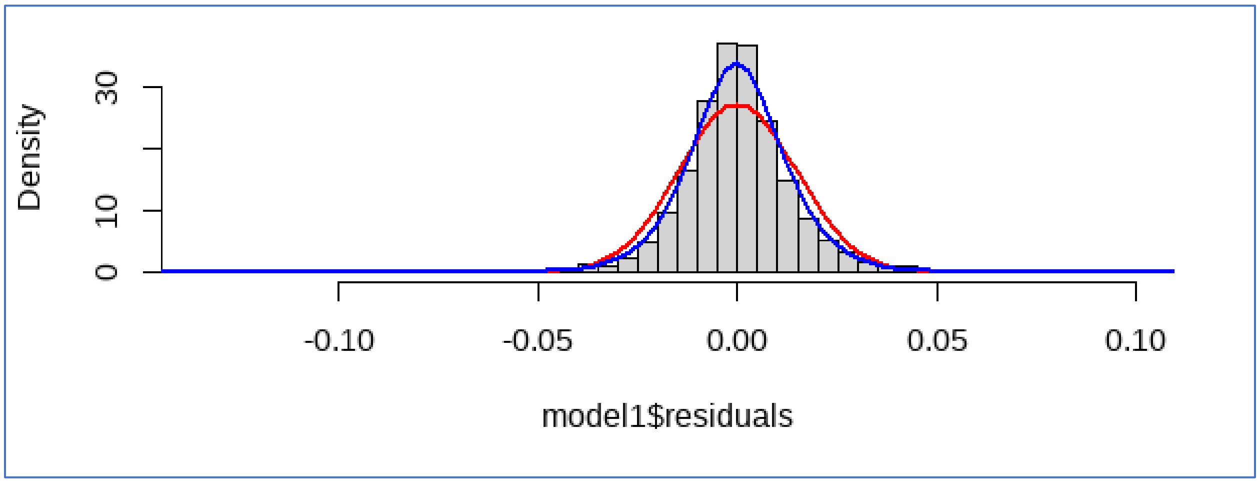
```
Correlation matrix:
             df       mean         sd
df   1.00000000 0.04149317 0.65784214
mean 0.04149317 1.00000000 0.03840503
sd   0.65784214 0.03840503 1.00000000
```

Distribution	**Loglikelihood:**	**AIC:**	**BIC:**
Normal	7489.019	-14974.04	-14962.26
Logistic	7716.864	-15429.73	-15417.95
Cauchy	7534.696	-15065.39	-15053.61
t	**7774.942**	**-15543.88**	**-15526.22**

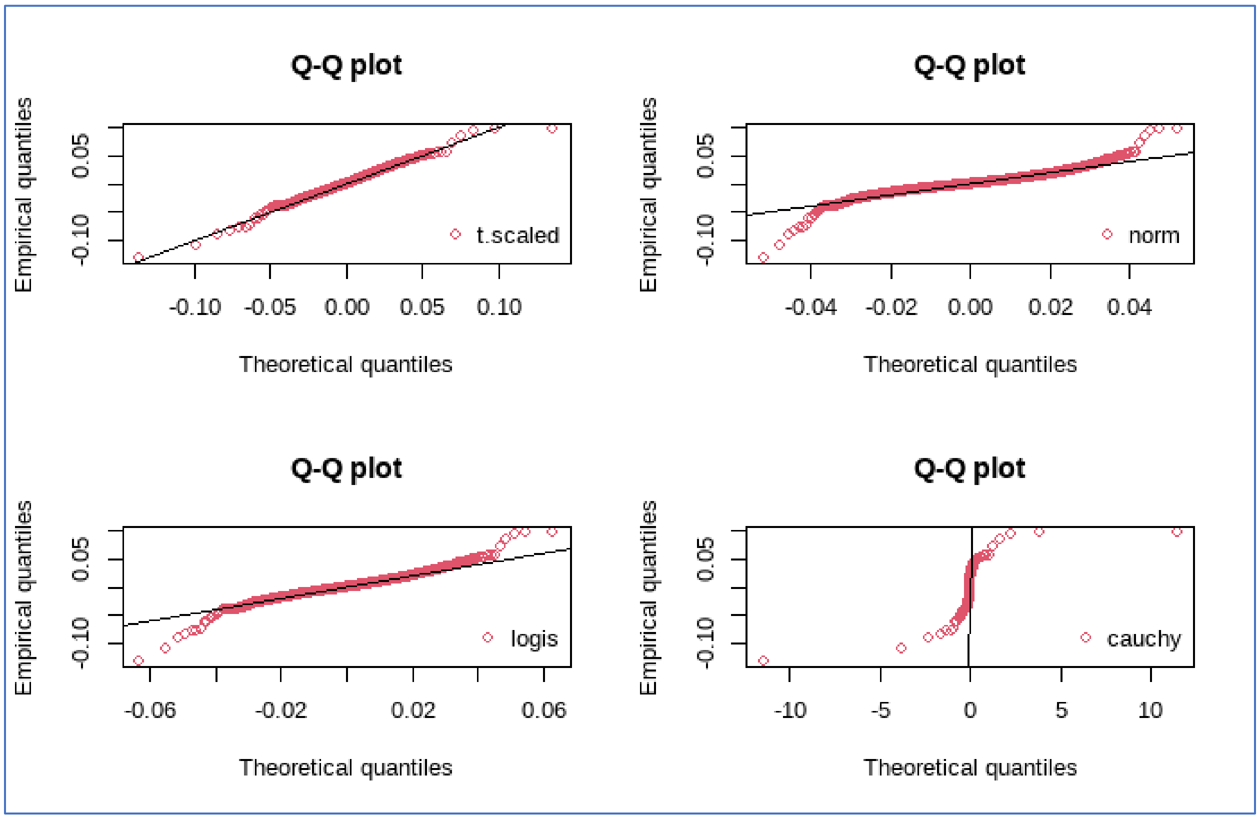
Clearly t distribution is the best fit with highest LL and lowest AIC and BIC.

Graphical Fitting of Distributions

Let us also check graphically the fitting behavior of these distributions.



Now we are using QQ plot to check which distribution is better. As is evident, for ‘t’ distribution the data points are lying on a nearly 45-degree slope line.



Another way to check the distribution of fit is by checking goodness of fit.

Goodness-of-fit statistics

```
                             Normal Distribution Logistic Distribution
Kolmogorov-Smirnov statistic          0.07056107            0.02921331
Cramer-von Mises statistic            5.33238112            0.67038670
Anderson-Darling statistic           31.75475507            4.43838716
                             Cauchy Distribution t Distribution
Kolmogorov-Smirnov statistic          0.05444291     0.01081562
Cramer-von Mises statistic            1.54765364     0.05157075
Anderson-Darling statistic           20.00844149     0.52245688

Goodness-of-fit criteria
                               Normal Distribution Logistic Distribution
Akaike's Information Criterion           -14967.54             -15422.89
Bayesian Information Criterion           -14955.77             -15411.11
                               Cauchy Distribution t Distribution
Akaike's Information Criterion           -15058.62      -15536.91
Bayesian Information Criterion           -15046.84      -15519.24
```

As seen above, t is having the least value as per KS, CVM and AD. Having decided the best fittin g distribution, next step is to calculate the range of values for prediction.

Monte Carlo Simulation

Generating data from a distribution, we get following results.

Distribution:	**1%**	**5%**
Normal	-0.03384368	-0.02395165
Logistic	-0.03334452	-0.02138896
Cauchy	-0.22323164	-0.04554845
t	**-2.698576**	**-1.428698**

The calculation for t distribution has been done using following formula.

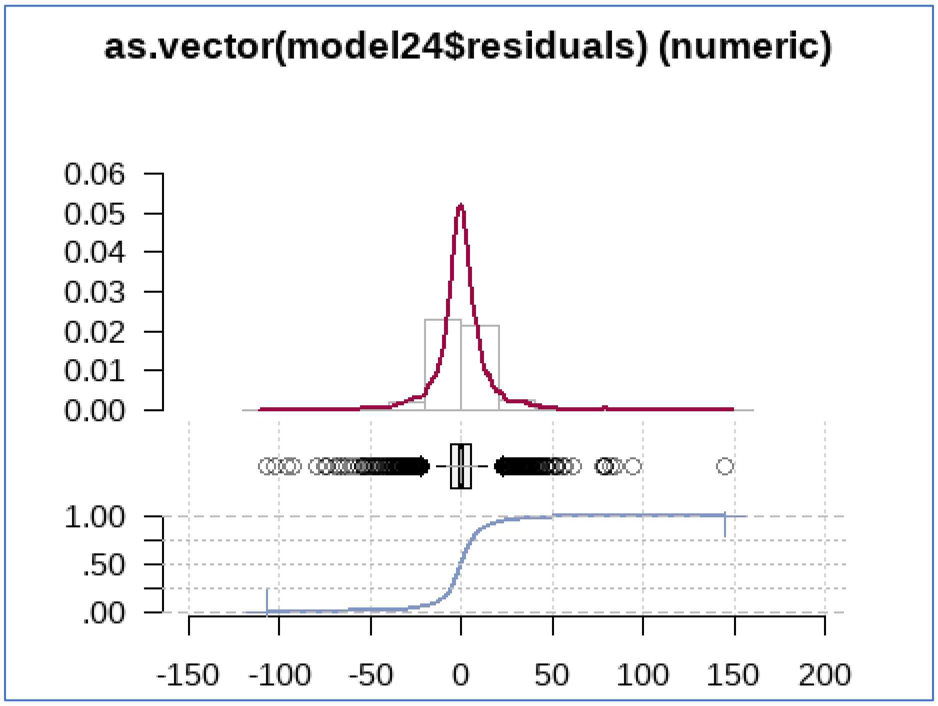
t value = mean + quantile * standard deviation for the t distribution
For 1%: 0.0421728 + 0.6574522*-4.168743 = -2.698576
For 5%: 0.0421728 + 0.6574522*-2.237229 = -1.428698

Distribution of HDFC Bank price

We are working with residuals of HDFC Bank price now.

```
as.vector(model24$residuals) (numeric)

length           n           NAs     unique        0s      mean      meanCI'
2'683        2'683             0        = n         0  -0.007285   -0.565748
             100.0%         0.0%                    0.0%               0.551178

.05         .10        .25     median        .75       .90     .95
-21.719  -13.698    -5.5917  -0.194575   5.512314  14.026693  21.91

     range          sd           vcoef         mad         IQR      skew       kurt
 251.570018   14.752337  -2'025.092303   8.243124   11.104060   0.081168   11.2

lowest : -106.972998, -102.971582, -95.741997, -92.319410, -79.725657
highest: 78.844636, 82.328344, 84.449085, 94.510084, 144.59702
```

Below are also given the pdf, cum pdf which are validating the same conclusions.



Skewness is close to zero, but kurtosis is very high confirming thick tails. Hence normal distribut ion may not be appropriate fit. Therefore, we shall fit 4 different distributions here to select the b est fit.
The summary of normal, logistic, Cauchy and t distributions.is given below.

> summary(normaldist1)

```
Fitting of the distribution ' norm ' by maximum likelihood
Parameters :
        estimate        Std. Error
mean    -0.007284773    0.2847538
sd      14.749587318    0.2013513
```

Loglikelihood: -11027.54   AIC: 22059.08   BIC: 22070.87

```
Correlation matrix:
        mean    sd
mean    1       0
sd      0       1
```

> summary(logisdist1)

```
Fitting of the distribution ' logis ' by maximum likelihood
Parameters :
            estimate Std. Error
location -0.02590957  0.2208425
scale     6.84564113  0.1145690
```

Loglikelihood: -10664.42   AIC: 21332.84   BIC: 21344.63

```
Correlation matrix:
            location       scale
location 1.000000000 0.003592805
scale    0.003592805 1.000000000
```

```
> summary(cauchydist1)
```

Fitting of the distribution ' cauchy ' by maximum likelihood

```
Parameters :
           estimate Std. Error
location -0.1747337  0.1517586
scale     5.2957935  0.1383946
```

Loglikelihood: -10568.35 AIC: 21140.71 BIC: 21152.49

```
Correlation matrix:
           location      scale
location 1.00000000 0.01910102
scale    0.01910102 1.00000000
```

```
> summary(tdist1)
```

Fitting of the distribution ' t.scaled ' by maximum likelihood

```
Parameters :
       estimate Std. Error
df    2.0060532  0.1057377
mean -0.1032074  0.1713809
sd    6.9525745  0.2022568
```

Loglikelihood: -10458.46 AIC: 20922.91 BIC: 20940.6

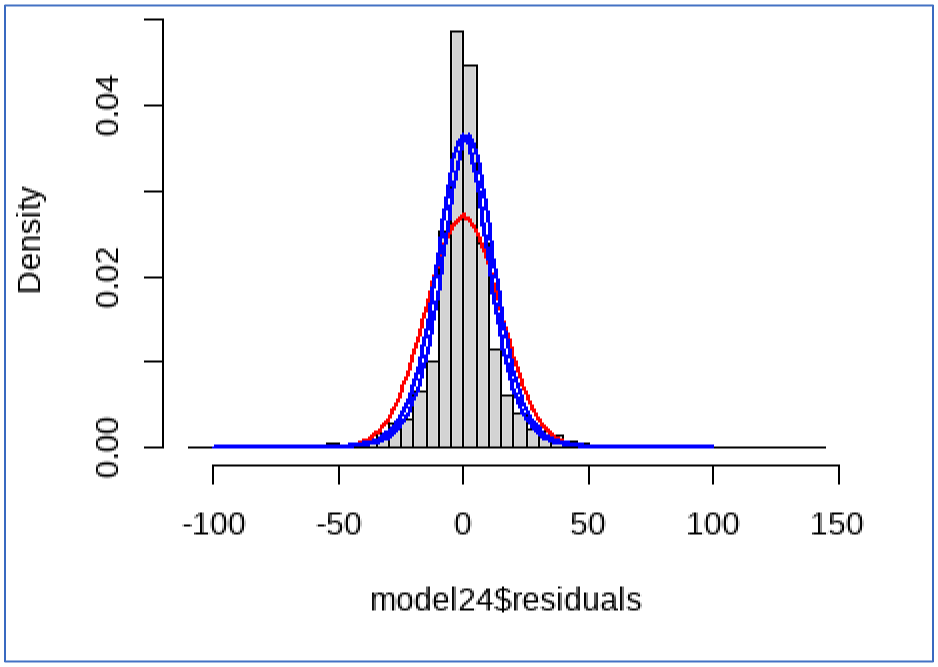
```
Correlation matrix:
             df       mean         sd
df   1.00000000 0.02949488 0.65652396
mean 0.02949488 1.00000000 0.03016227
sd   0.65652396 0.03016227 1.00000000
```

Let us summarize the four distributions based on selection criteria. We select a distribution wher e log likelihood is highest, and AIC and BIS are lowest.

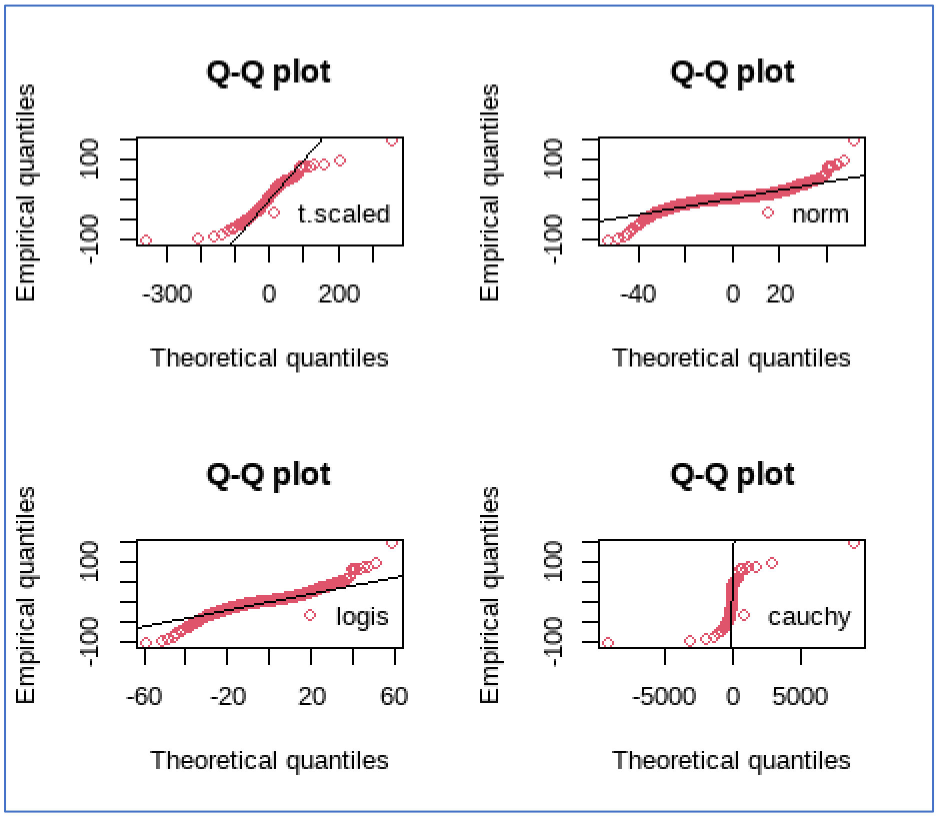
Distribution	**Loglikelihood:**	**AIC:**	**BIC:**
Normal	-11027.54	AIC: 22059.08	BIC: 22070.87
Logistic	-10664.42	AIC: 21332.84	BIC: 21344.63
Cauchy	-10568.35	AIC: 21140.71	BIC: 21152.49
t	**-10458.46**	**AIC: 20922.91**	**BIC: 20940.6**

Clearly t distribution is the best fit with highest LL and lowest AIC and BIC.

Let us also check graphically the fitting behavior of these distributions.



Now we are using QQ plot to check which distribution is better. As is evident, for ‘t’ distribution the data points are lying on a nearly 45-degree slope line.



Another way to check the distribution is by checking goodness of fit.

Goodness-of-fit statistics:

```
                             Normal Distribution Logistic Distribution Cauchy

Kolmogorov-Smirnov statistic   0.1179173       0.06450038            0.03191278
Cramer-von Mises statistic     15.2829413      4.19321512            0.52605112
Anderson-Darling statistic      Inf           24.71287667            8.62842251
                             t Distribution
Kolmogorov-Smirnov statistic     0.01289411
Cramer-von Mises statistic       0.11736091
Anderson-Darling statistic       0.94586122

Goodness-of-fit criteria
                             Normal Distribution Logistic Distribution Cauchy

Akaike's Information Criterion 22059.08        21332.84              21140.71
Bayesian Information Criterion 22070.87        21344.63              21152.49
                               t Distribution
Akaike's Information Criterion       20922.91
Bayesian Information Criterion       20940.60
```

't' distribution is having the least value as per KS, CVM and AD. Having decided the best fitting distribution, next step is to calculate the range of values for prediction.

Monte Carlo Simulation

Generating data from a distribution we get following results from R output.

Distribution:	**1%**	**5%**
Normal	-34.13334	-24.15862
Logistic	-31.01934	-19.86649
Cauchy	-175.06282	-35.65917
t	-49.83974	-20.61883

The calculation for t distribution above has been done using following formula.

t value = mean + quantile * standard deviation for the t distribution

For 1%: -0.1032074 + 6.9525745*-7.153685=-49.83974
For 5%: -0.1032074 + 6.9525745*-2.950795=-20.61883

Exploring ML models

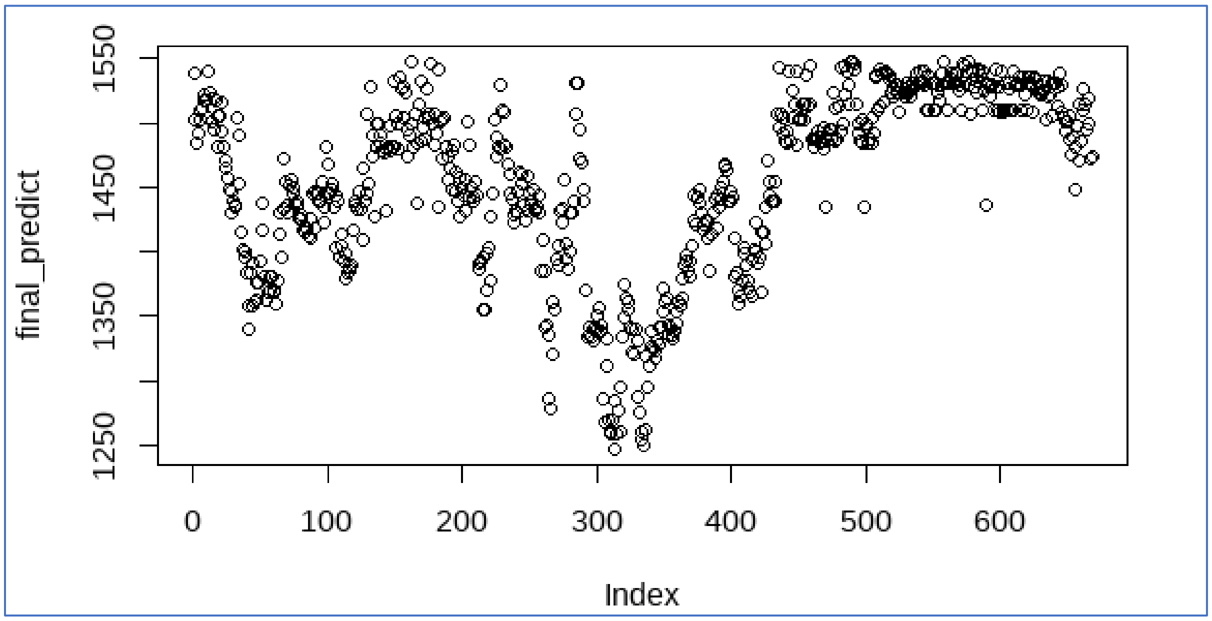
With multiple factors involved in predicting stock prices, it is challenging to predict stock prices with high accuracy, and this is where machine learning plays a vital role.

We split the data into test and train as per requirement. We also split test data into dependent and independent variable combinations. This will be required for comparison with predicted values a nd checking the model accuracy.

Forecasting using Random Forest ML model: HDFC Bank Price

```
Call:
 randomForest(formula = HDFCBANK.NS.Adjusted ~ ., data = train_data1)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1
          Mean of squared residuals: 32.33461
                    % Var explained: 99.97
```

The plot of predicted values is shown below.



The error value using two different calculation methods is as below.

Method 1: "RMSE: 0.0437841502367694"
Method 2: "RMSE: 0.0437841502367694"

Now we are predicting the values using the fitted model and comparing with actual test values. A s seen below, error is hovering around 1.3 to 3.24 % for given values which is similar to the error seen in RMSE value above.

	Predicted Value	**Actual Value**	**% Error**
2023-10-13	1514.167	1535.75	1.405380
2023-10-16	1495.252	1529.60	2.245574
2023-10-17	1519.340	1541.20	1.418393
2023-10-18	1498.991	1519.75	1.365953
2023-10-19	1471.714	1514.95	2.853975
2023-10-20	1473.362	1522.80	3.246520

Forecasting using Neural Network ML model: HDFC Bank Price

Series: train_data1$HDFCBANK.NS.Adjusted

Model: NNAR (1,3)
Call: nnetar (y = train_data1$HDFCBANK.NS.Adjusted, size = 3, lambda = L)
Average of 20 networks, each of which is a 1-3-1 network with 10 weights.
options were - linear output units.
sigma^2 estimated as 0.005831
The error value for this model is as below.
"Error Neural Network: 26.5710488628941%"

Now we are predicting the values using the fitted model and comparing with actual test values.

Point.	Forecast	Actual Value
2663	1915.262	1535.75
2664	1915.262	1529.60
2665	1915.262	1541.20
2666	1915.262	1519.75
2667	1915.262	1514.95
2668	1915.262	1522.80

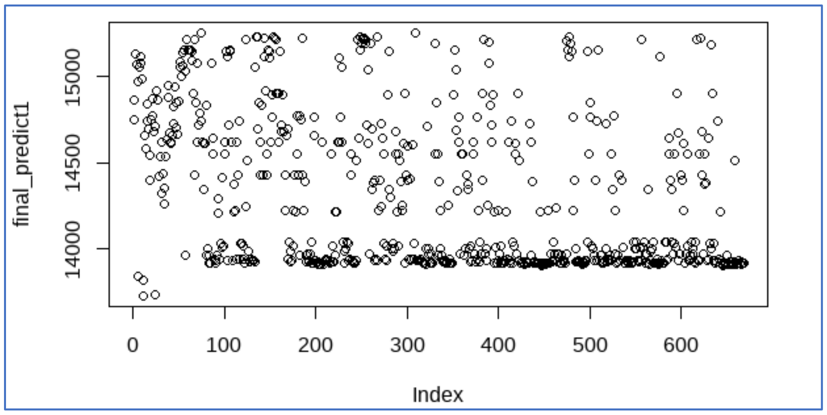
In this case, error around 26% is quite high compared to 4% of Random Forest Model. Thus, we conclude that Random Forest is the best fitting model for above data set.

Forecasting using Random Forest ML model: Nifty Price

Call:

```
 randomForest(formula = NSEI.Adjusted ~ ., data = train_data2)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1
          Mean of squared residuals: 1723.6
                    % Var explained: 99.96
```

The plot of predicted values is shown below.



The error value using two different calculation methods is as below.

```
Method 1: "RMSE: 0.214413957940767"
Method 2: "RMSE: 0.214413957940767"
```

Next, we are predicting the values using the fitted model and comparing with actual test values. As seen below, error is hovering around 27 to 28 % for given values which is more than estimate using RMSE above.

	Predicted Value	**Actual Value**	**% Error**
2023-11-02	13936.97	19133.25	27.15838
2023-11-03	13917.45	19230.60	27.62863
2023-11-06	13915.42	19411.75	28.31444
2023-11-07	13928.78	19406.70	28.22694
2023-11-08	13918.71	19443.50	28.41458
2023-11-09	13922.83	19395.30	28.21544

Forecasting using Neural Network ML model: Nifty Price

```
Series: train_data2$NSEI.Adjusted
Model:  NNAR(2,3)
Call:   nnetar(y = train_data2$NSEI.Adjusted, size = 3, lambda = L)

Average of 20 networks, each of which is
a 2-3-1 network with 13 weights
options were - linear output units
sigma^2 estimated as 2.706
```

The error value for this model is as below.

```
"Error Neural Network: 15.4611270357076%"
```

Now we are predicting the values using the fitted model and comparing with actual test values.

Point.	**Forecast**	**Actual Value**
2663	14623.86	19133.25
2664	14623.86	19230.60
2665	14623.86	19411.75
2666	14623.86	19406.70
2667	14623.86	19443.50
2668	14623.86	19395.30

In this case, error is around 25%.

Analysis & Forecasting based on 3 years Data:

When we reduce time horizon of data from 10 to 3 years i.e., if we do model for data starting from 31-12-2020, we see positive changes in model performance at some places. The code for the same has been attached in the annexure.

CHAPTER 5: CONCLUSION AND FINDINGS

Conclusions including Limitations

Following conclusions can be drawn from this project work.

1. Weekly and Monthly stock prices have a smoother trend compared to daily stock price trend.
2. Data taken from https://finance.yahoo.com/quote is in timeseries format and needs to be converted to data frame format for analysis.
3. Stock returns data is symmetric with negative skewness implying fall is drastic.
4. Stock returns data has high Kurtosis, i.e., is thick tailed with presence of extreme vales.
5. Absolute prices are less non normal as compared to their compounded returns data.
6. All the data sets, namely HDFC Bank and Nifty returns and prices considered in this project do not exhibit normality of data.
7. The correlation between HDFC and Nifty returns is not strong.
8. Correlation and partial correlation are present in returns and prices data of HDFC Bank and Nifty. Therefore, ARIMA is an appropriate model for forecasting.
9. RMSE/predicted value ratio is very high for HDFC and Nifty reruns implying that model acc uracy is very poor. The same for prices of HDFC stock and Nifty 50 is reasonable.
10. ARIMA model has not been able to capture correlation and partial correlation in data due to l ack of normal behavior of data. This calls for Transformation of data ARIMA.
11. There is correlation and partial correlation in residuals and squared residuals hence GARCH f amily of ARIMA models may not be worth applying for.
12. On power transformation, lambda values have changed but errors have not improved significa ntly. Clearly, ARIMA model has not improved the results even after data transformation.
13. On out of sample modeling with BoxCox transformation, results have not improved.
14. t is the best fitting distribution for random part of data.
15. Random Forest is the best fitting model for the data set considered in this project.
16. For 3 years data, better prediction accuracy has been achieved. ARIMA results have significa ntly improved. t distribution continues to remain the best fitting distribution for random part p rediction in time series data. Machine learning models, Random Forest as well as Neural netw orks give significantly better prediction for both HDFC stock price as well as Nifty index. Ou t of these two, Random Forest is still the best fitting model of all.

Findings

Following findings can be enumerated for this project work.

1. HDFC and Nifty Stock data have missing values.
2. Nifty returns are less symmetric and having higher kurtosis than HDFC Bank returns.
3. Stock prices have skewness and kurtosis significantly improved towards normality vis-à-vis t heir returns data.
4. Normality tests, viz. Jarque - Bera Normality Test, Kolmogorov-Smirnov test and Shapiro - Wilk Normality Tests result into p value significantly lesser than 0.05.

5. HDFC Bank return correlation is statistically significant at lags 2, 6, 7, 19 & its partial correla tion is statistically significant at lags 2, 6,7. Nifty returns correlation is statistically significant at lags 5,6,7,10,11,12,16 and partial correlation is statistically significant at lags 5, 6,7, 11,12. HDFC Bank & Nifty Stock prices correlation are statistically significant at all lags and partial correlation are statistically significant at lags 1.
6. Box-Ljung test gives p value less than 0.05 for both, HDFC Bank & Nifty returns and prices, meaning alternative hypothesis is accepted i.e., auto correlation is present.
7. AUTO ARIMA provides best models for HDFC return, Nifty return, HDFC price and Nifty p rice as ARIMA (2,0,2) with non-zero mean, ARIMA (0,0,0) with non-zero mean, ARIMA (2,1 ,2) with drift and ARIMA (0,1,0) with drift respectively.
8. RMSE/Predicted value for HDFC return, Nifty return, HDFC price and Nifty price are 53.5, 23.8, 0.00984 and 0.005983592 respectively.
9. p value is less than 0.05 for residuals and squared residuals confirming that there is correlatio n in residuals as well as in squared residuals.
10. On Power transformation, for HDFC return, Nifty return, HDFC price and Nifty price best m odels are ARIMA (2,0,2) with non-zero mean, ARIMA (1,0,0) with non-zero mean and lamb da= 0.8584132, ARIMA (2,1,2) with drift and lambda= 0.004546487 and ARIMA (0,1,0) wit h drift and lambda= 0.3930239.
11. On BoX Cox transformation of non-normal stock returns data, we get RMSE less for non-tran sformed data which is strange.
12. For random part prediction of HDFC Bank returns, t distribution gives the highest Loglikelih ood and lowest AIC/ BIC at 7774.942, -15543.88 and -15526.22 respectively out of four distr ibutions i.e., normal, logistic, Cauchy and t distributions. This is also validated by QQ plots a nd goodness of fit tests i.e., Kolmogorov-Smirnov statistic, Cramer-von Mises statistic and A nderson-Darling statistic which give least value for t distribution.
13. For random part prediction of HDFC Bank price, t distribution gives the highest Loglikelihoo d and lowest AIC/ BIC at -10458.46, 20922.91 and 20940.6 respectively out of four distributi ons i.e., normal, logistic, Cauchy and t distributions. This is also validated by QQ plots where data points are lying on 45-degree slope line and goodness of fit tests i.e., Kolmogorov-Smirn ov statistic, Cramer-von Mises statistic and Anderson-Darling statistic which give least value for t distribution.
14. Random forest model for HDFC price gives “RMSE: 0.0437841502367694" and Error Neura l Network is 26.5710488628941%".
15. Random forest model for Nifty price gives "RMSE: 0.214413957940767" and Error Neural N etwork: 15.4611270357076%"
16. For 3 years data, ARIMA model error is lesser at 1% and 0.7% for HDFC and Nifty Index pri ces. But ARIMA is not able to work well for their returns data in this case also. The error in p rediction of Random Forest is 2% against earlier 4 % for HDFC prices and the same is 5% ag ainst earlier 28% for Nifty index. As regards Neural network prediction, error here is 3.8% ag ainst earlier 26% for HDFC price and it is 7.7% as against earlier 15.5% for Nifty index.

CHAPTER 6: SUGGESTIONS AND RECOMMENDATIONS

It is suggested and recommended to:

1. Share the analysis with peer community and seek critique for improvement.
2. Implement real-time data feeds for continuous model updates.
3. Continue exploring with more advanced time series models beyond ARIMA, such as deep learning architectures (LSTM, GRU), ensemble methods, or hybrid models for improved forecasting accuracy.
4. Fine-tune model parameters regularly by automating optimization processes to adapt to evolving market situations.
5. Enhance predictive models by integrating alternative data sources like social media sentiment, macroeconomic indicators, news sentiment, or industry-specific data to capture comprehensive market influences.
6. Investigate methods to make models more resilient to extreme market events by incorporating outlier detection strategies.
7. Develop strategies for dynamic model adaptation to structural changes in market behavior.
8. Provide a dashboard with visual components, adding interactive features, trend analysis, and scenario testing for enhanced user experience.
9. Incorporate features allowing users to customize parameters and set alerts for specific stock price movements or market conditions.
10. Develop tools to assess portfolio risk and suggest optimal allocations based on forecasted stock prices and risk.
11. Introduce simulations for testing portfolios under different market scenarios derived from forecasting models.
12. Establish a feedback loop for users to provide insights on the accuracy of predictions, aiding in model improvement.
13. Maintain detailed documentation of model updates, methodologies, and findings, ensuring future research.
14. Share insights from the project study through presentations, publications, or workshops to benefit the wider student community.
15. Engage in collaborations with industry partners for continued research and access to diverse datasets or methodologies.

Implementing these recommendations will foster continuous improvement in stock price forecasting models, support informed decision-making, and contribute to the advancement of predictive analytics in the dynamic domain of stock price prediction and wealth building.

These recommendations aim to provide actionable steps for enhancing the effectiveness and applicability of stock price forecasting models developed using R.

CHAPTER 7: ANNEXURE – R CODE

R Code considering 10 years period data for analysis

```
# Installing & loading all the required packages for analysis in R
install.packages(c('quantmod','qrmtools','MASS','PerformanceAnalytics','TSA'),dependencies =
T)
install.packages(c('aTSA','forecast','fBasics','urca','dlookr'),dependencies = T)
install.packages(c('SmartEDA','nortest','DescTools','TTR','tidyverse'),dependencies = T)
install.packages(c('neuralnet','keras','tensorflow','lattice','randomForest'),dependencies = T)
install.packages(c('fitdistrplus','metRology'),dependencies = T)

library(quantmod)# Extract data from Yahoo Finance
library(qrmtools)
library(MASS)
library(PerformanceAnalytics)# Calculations that are performed on a stock/portfolio
library(TSA)
library(aTSA)
library(forecast)# Forecasting the time series
library(fBasics)# Basic Statistical Calculations
library(urca)# Stationarity Analysis
library(dlookr)
library("SmartEDA")# Exploratory Data Analysis
library(nortest)
library(DescTools)
library(TTR)
library(tidyverse)
library(neuralnet)
library('lattice')
library('randomForest')
library(fitdistrplus)
library(metRology)#Compare the performance of different distributions

# Download data from Yahoo Finance
HDFC<-getSymbols("HDFCBANK.NS",from="2012-12-31", To= "2023-11-10",
auto.assign=FALSE)
Nifty<-getSymbols("^NSEI",from="2012-12-31",To="2023-11-10",auto.assign=FALSE)

# Plotting the data downloaded from Yahoo Finance (Time Series Data)
class(HDFC)# To know structure of data
plot(HDFC$HDFCBANK.NS.Adjusted)
plot(Nifty$NSEI.Adjusted)
summary(HDFC)# To understand missing values in any columns
summary(Nifty)
```

```
HDFC<-na.omit(HDFC)# Removing the rows containing the missing data
Nifty<-na.omit(Nifty)
data.monthly<-to.monthly(HDFC)
data.weekly<-to.weekly(HDFC)
data.monthly1<-to.monthly(Nifty)
data.weekly1<-to.weekly(Nifty)
plot(data.monthly$HDFC.Adjusted)
plot(data.weekly$HDFC.Adjusted)
plot(data.monthly1$Nifty.Adjusted)
plot(data.weekly1$Nifty.Adjusted)

#plotting Candle stick chart
OHLC<-data.monthly[,-6]
OHLC1<-data.monthly1[,-6]
HDFC.ohlc<-
as.quantmod.OHLC(OHLC,col.names=c("Open","High","Low","Close","Volume"))
chartSeries(HDFC.ohlc,theme="white.mono",name="HDFC OHLC", up.col =
"green",dn.col="red")

#Calculating returns of data
returns<-diff(log(HDFC$HDFCBANK.NS.Adjusted))
returns<-na.omit(returns)
head(returns)
tail(returns)

Nifty_Returns<-diff(log(Nifty$NSEI.Adjusted))
head(Nifty_Returns)
Nifty_Returns<-na.omit(Nifty_Returns)
ret1<-merge(returns,Nifty_Returns)
head(ret1)
ret1<-na.omit(ret1)
ret2<-as.data.frame(ret1)#converting data into data frame so as to use Desc function
#write.csv(ret2,"ret2.csv",row.names=FALSE) to output returns into a file
head(ret2)

#EDA - Calculating descriptive statistics for returns and price data
Desc(ret2)
HDFC1<-as.data.frame(HDFC$HDFCBANK.NS.Adjusted)
Desc(HDFC1)
Nifty1<-as.data.frame(Nifty$NSEI.Adjusted)
Desc(Nifty1)

#EDA of bivariate data-Calculation of correlation coefficient
```

```
correlate(ret2,method="spearman")#Nonparametric correlation coefficient uses ranking data
correlate(ret2,method="kendall")#Nonparametric correlation coefficient uses ranking data

#Test of Normality for HDFC & Nifty returns and prices
jarqueberaTest(ret1$HDFCBANK.NS.Adjusted)#Jarque - Bera Normality Test
ksnormTest(ret1$HDFCBANK.NS.Adjusted)# Kolmogorov Smirnov Test
shapiroTest(ret1$HDFCBANK.NS.Adjusted)#Shapiro - Wilk Normality Test
jarqueberaTest(ret1$NSEI.Adjusted)#Jarque - Bera Normality Test
ksnormTest(ret1$NSEI.Adjusted)# Kolmogorov Smirnov Test
shapiroTest(ret1$NSEI.Adjusted)#Shapiro - Wilk Normality Test
jarqueberaTest(HDFC$HDFCBANK.NS.Adjusted)#Jarque - Bera Normality Test
ksnormTest(HDFC$HDFCBANK.NS.Adjusted)# Kolmogorov Smirnov Test
shapiroTest(HDFC$HDFCBANK.NS.Adjusted)#Shapiro - Wilk Normality Test
jarqueberaTest(Nifty$NSEI.Adjusted)#Jarque - Bera Normality Test
ksnormTest(Nifty$NSEI.Adjusted)# Kolmogorov Smirnov Test
shapiroTest(Nifty$NSEI.Adjusted)#Shapiro - Wilk Normality Test

# Auto correlation and partial auto correlation
acf(ret1$HDFCBANK.NS.Adjusted)
pacf(ret1$HDFCBANK.NS.Adjusted)
acf(ret1$NSEI.Adjusted)
pacf(ret1$NSEI.Adjusted)
acf(HDFC$HDFCBANK.NS.Adjusted)
pacf(HDFC$HDFCBANK.NS.Adjusted)
acf(Nifty$NSEI.Adjusted)
pacf(Nifty$NSEI.Adjusted)

#Box test for stationarity analysis
Box.test(ret1$HDFCBANK.NS.Adjusted, lag = 30, type = "Ljung-Box")
Box.test(ret1$NSEI.Adjusted, lag = 30, type = "Ljung-Box")
Box.test(HDFC$HDFCBANK.NS.Adjusted, lag = 30, type = "Ljung-Box")
Box.test(Nifty$NSEI.Adjusted, lag = 30, type = "Ljung-Box")

#Model Building using AUTO ARIMA
model1<-auto.arima(ret1$HDFCBANK.NS.Adjusted,max.p = 10, max.d=3, max.q = 20,
max.order = 20, trace = TRUE)
model2<-auto.arima(ret1$NSEI.Adjusted,max.p = 10, max.d=3, max.q = 10, max.order = 20,
trace = TRUE)
model3<-auto.arima(HDFC$HDFCBANK.NS.Adjusted,max.p = 10, max.d=3, max.q = 10,
max.order = 20, trace = TRUE)
model4<-auto.arima(Nifty$NSEI.Adjusted,max.p = 10, max.d=3, max.q = 10, max.order = 20,
trace = TRUE)
```

```
plot(ret1$HDFCBANK.NS.Adjusted)

#Model prediction for returns and price

summary(model1)
forecast(model1, h = 20)
par(mar = c(1,1,1,1))
tsdiag(model1)#for residual plot, ACF function and p values plot
#RMSE/Predicted
0.01462972/0.0002736028

summary(model2)
forecast(model2, h = 20)
par(mar = c(1,1,1,1))
tsdiag(model2)
#RMSE/Predicted
0.01060959/0.0004463128

summary(model3)
forecast(model3, h = 20)
par(mar = c(1,1,1,1))
tsdiag(model3)
#RMSE/Predicted
14.68772/1492.464

summary(model4)
forecast(model4, h = 20)
par(mar = c(1,1,1,1))
tsdiag(model4)
#RMSE/Predicted
116.2637/19430.42

#Residuals and squared residuals Stationarity analysis
Box.test(model1$residuals, lag = 20, type = "Ljung-Box")
Box.test(model1$residuals^2, lag = 20, type = "Ljung-Box")
Box.test(model2$residuals, lag = 20, type = "Ljung-Box")
Box.test(model2$residuals^2, lag = 20, type = "Ljung-Box")
Box.test(model3$residuals, lag = 20, type = "Ljung-Box")
Box.test(model3$residuals^2, lag = 20, type = "Ljung-Box")
Box.test(model4$residuals, lag = 20, type = "Ljung-Box")
Box.test(model4$residuals^2, lag = 20, type = "Ljung-Box")

dev.off()
```

```
#Modeling using AUTO ARIMA with power transformation
model11<-auto.arima(ret1$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE, lambda="auto")
summary(model11)
forecast(model11, h = 20)
0.01462706/0.0002736233

model21<-auto.arima(ret1$NSEI.Adjusted,max.p = 10, max.d=2, max.q = 10, max.order = 20,
trace = TRUE, lambda="auto")
summary(model21)
forecast(model21, h = 20)
0.01060685/0.0003080859

model31<-auto.arima(HDFC$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE, lambda="auto")
summary(model31)
forecast(model31, h = 20)
14.76288/1489.190

model41<-auto.arima(Nifty$NSEI.Adjusted,max.p = 10, max.d=2, max.q = 10, max.order = 20,
trace = TRUE, lambda="auto")
summary(model41)
forecast(model41, h = 20)
116.2459/19450.48
# ARIMA with BoXCox
y1<-c()
for (i in 1:100) {
        model1<-auto.arima(ret1$HDFCBANK.NS.Adjusted[i:(i+2568)],max.p = 10, max.q =
10, max.order = 20, lambda="auto")
        y<-as.vector(predict(model1,1)$pred)
        y1<-c(y1,y)
}
# ARIMA without BC
y2<-c()
for (i in 1:100) {
        model2<-auto.arima(ret1$HDFCBANK.NS.Adjusted[i:(i+2568)],max.p = 10, max.q =
10, max.order = 20)
        y<-as.vector(predict(model2,1)$pred)
        y2<-c(y2,y)
}

RMSE(y1,tail(ret1$HDFCBANK.NS.Adjusted,100))
RMSE(y2,tail(ret1$HDFCBANK.NS.Adjusted,100))
```

```
# Fitting appropriate distributions to the returns data
par(mfrow = c(1,1))
model14<-auto.arima(ret1$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE)
Desc(as.vector(model14$residuals))

normaldist<-fitdist(as.vector(model14$residuals),distr = "norm")
logisdist<-fitdist(as.vector(model14$residuals),distr = "logis")
cauchydist<-fitdist(as.vector(model14$residuals),distr = "cauchy")
tdist<-fitdist(as.vector(model14$residuals),distr =
"t.scaled",start=list(df=3,mean=mean(as.vector(model1$residuals)),sd=sd(as.vector(model1$resi
duals))))

summary(normaldist)
summary(logisdist)
summary(cauchydist)
summary(tdist)

#Fitting the distribution graphically
par(mfrow = c(1,1))
hist(model14$residuals, pch=20, breaks=50, main="", freq=FALSE)
curve(dnorm(x, normaldist$estimate[1], normaldist$estimate[2]),from = -0.15, to = 0.15,
col="red", lwd=2, add=T)
curve(dlogis(x, logisdist$estimate[1], logisdist$estimate[2]),from = -0.15, to = 0.15, col="blue",
lwd=2, add=T)

#Checking QQ plots
par(mfrow = c(2, 2))
qqcomp(list(tdist))
qqcomp(list(normaldist))
qqcomp(list(logisdist))
qqcomp(list(cauchydist))

#Goodness of fit tests
gofstat(list(normaldist, logisdist, cauchydist, tdist), fitnames = c("Normal Distribution",
"Logistic Distribution", "Cauchy Distribution", "t Distribution"))

# Monte carlo simulations
# Generate 10,000 observations
set.seed(1000)
x<-rnorm(10000,mean = -5.533844e-07 , sd = 1.462730e-02 )
head(x)
min(x)
```

```
max(x)
# 99% confidence level VaR and 95% confidence level VaR
quantile(x,c(0.01,0.05))

x1<-rlogis(10000,location=-0.0001203605 , scale=0.0073383519)
min(x1)
quantile(x1,c(0.01,0.05))

x2<-rcauchy(10000,location=-0.0003201685 , scale=0.0067499918 )
min(x2)
quantile(x2,c(0.01,0.05))

x3<-rt(10000,df = 3.4170956047)
min(x3)
quantile(x3,c(0.01,0.05))

#calculating t values
0.0421728 + 0.6574522*-4.168743
0.0421728 + 0.6574522*-2.237229

# Fitting Distribution to HDFC Bank Price now
par(mfrow = c(1,1))
model24<-auto.arima(HDFC$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE)
Desc(as.vector(model24$residuals))

normaldist1<-fitdist(as.vector(model24$residuals),distr = "norm")
logisdist1<-fitdist(as.vector(model24$residuals),distr = "logis")
cauchydist1<-fitdist(as.vector(model24$residuals),distr = "cauchy")
tdist1<-fitdist(as.vector(model24$residuals),distr =
"t.scaled",start=list(df=3,mean=mean(as.vector(model1$residuals)),sd=sd(as.vector(model1$resi
duals))))

summary(normaldist1)
summary(logisdist1)
summary(cauchydist1)
summary(tdist1)

#Graphical fitting of distributions
par(mfrow = c(1,1))
hist(model24$residuals, pch=20, breaks=50, main="", freq=FALSE)
curve(dnorm(x, normaldist1$estimate[1], normaldist1$estimate[2]),from = -100, to = 100,
col="red", lwd=2, add=T)
```

```
curve(dlogis(x, logisdist1$estimate[1], logisdist1$estimate[2]),from = -100, to = 100, col="blue",
lwd=2, add=T)
curve(dlogis(x, cauchydist1$estimate[1], logisdist1$estimate[2]),from = -100, to = 100,
col="blue", lwd=2, add=T)
curve(dlogis(x, tdist1$estimate[1], logisdist1$estimate[2]),from = -100, to = 100, col="blue",
lwd=2, add=T)

#Checking QQ plots
par(mfrow = c(2, 2))
qqcomp(list(tdist1))
qqcomp(list(normaldist1))
qqcomp(list(logisdist1))
qqcomp(list(cauchydist1))

#Goodness of fit tests
gofstat(list(normaldist1, logisdist1, cauchydist1, tdist1), fitnames = c("Normal Distribution",
"Logistic Distribution", "Cauchy Distribution", "t Distribution"))

# Monte carlo simulations for HDFC Stock price
# Generate 10,000 observations
set.seed(1000)
x<-rnorm(10000,mean = -0.007284773 , sd = 14.749587318 )
head(x)
min(x)
max(x)
# 99% confidence level VaR and 95% confidence level VaR
quantile(x,c(0.01,0.05))

x1<-rlogis(10000,location=-0.02590957 , scale=6.84564113)
min(x1)
quantile(x1,c(0.01,0.05))

x2<-rcauchy(10000,location=-0.1747337 , scale=5.2957935 )
min(x2)
quantile(x2,c(0.01,0.05))

x3<-rt(10000,df = 2.0060532)
min(x3)
quantile(x3,c(0.01,0.05))

#calculating t values
-0.1032074 + 6.9525745*-7.153685
-0.1032074 + 6.9525745*-2.950795
```

```
#======= Exploring ML models for stock price prediction=====
dim(HDFC)
dim(Nifty)
# data splitting into training and test data
train_data1<-HDFC[1:2000,]
test_data1<-HDFC[2001:2668,]
train_data2<-Nifty[1:2000,]
test_data2<-Nifty[2001:2668,]
#Splitting the test data for evaluation
X_test<-test_data1[,-6]
Y_test <-test_data1[,6]

#forecasting using Random Forest ML model
fit1<-randomForest(HDFCBANK.NS.Adjusted ~ .,data=train_data1)
fit1
final_predict<-predict(fit1,X_test)
dev.off()
plot(final_predict)
## calculate the rmse value
rmse1 <- sqrt(mean(as.matrix(log1p(Y_test) - log1p(final_predict))^2))
print(paste('RMSE:', rmse1))
require(Metrics)
rmse <- rmse(log1p(as.matrix(Y_test)), log1p(final_predict))
print(paste('RMSE:', rmse))

#prediction and comparing with actual test values
prediction_actual=data.frame(HDFCBANK.NS.Adjusted=final_predict, Y_test, (Y_test-
final_predict)*100/Y_test)
tail(prediction_actual)

#Neural Network Time Series Forecasts
L <- BoxCox.lambda(train_data1$HDFCBANK.NS.Adjusted, method="loglik")
fit_nn <- nnetar(train_data1$HDFCBANK.NS.Adjusted, lambda=L, size=3)
fit_nn
fcast_nn <- forecast(fit_nn, h=668, lambda=L)

mape <- function(real, forecast){
  len <- length(real)
  return(sum( abs(real - forecast$mean[1:len]) / real) / len * 100)
}
paste0('Error Neural Network: ', mape(test_data1$HDFCBANK.NS.Adjusted, fcast_nn), '%')

#prediction and comparing with actual test values
```

```
prediction_actual1=data.frame(HDFCBANK.NS.Adjusted=fcast_nn, Y_test)
tail(prediction_actual1)

## Nifty Prediction using ML Models
#splitting the test data for evaluation
X1_test<-test_data2[,-6]
Y1_test <-test_data2[,6]

#forecasting using Random Forest ML model
fit2<-randomForest(NSEI.Adjusted ~ .,data=train_data2)
fit2
final_predict1<-predict(fit2,X1_test)
dev.off()
plot(final_predict1)
## calculate the rmse value using two methods
rmse1 <- sqrt(mean(as.matrix(log1p(Y1_test) - log1p(final_predict1))^2))
print(paste('RMSE:', rmse1))
require(Metrics)
rmse <- rmse(log1p(as.matrix(Y1_test)), log1p(final_predict1))
print(paste('RMSE:', rmse))

prediction_actual2=data.frame(NSEI.Adjusted=final_predict1, Y1_test, (Y1_test-
final_predict1)*100/Y1_test)
tail(prediction_actual2)
tail(Y1_test)

#Nifty - Neural Network Time Series Forecasts
L <- BoxCox.lambda(train_data2$NSEI.Adjusted, method="loglik")
fit.nn <- nnetar(train_data2$NSEI.Adjusted, lambda=L, size=3)
fit.nn
fcast_nn1 <- forecast(fit.nn, h=668, lambda=L)

mape <- function(real, forecast){
  len <- length(real)
  return(sum( abs(real - forecast$mean[1:len]) / real) / len * 100)
}
paste0('Error Neural Network: ', mape(test_data2$NSEI.Adjusted, fcast_nn1), '%')

#prediction and comparing with actual test values
prediction_actual3=data.frame(NSEI.Adjusted=fcast_nn1, Y1_test)
tail(prediction_actual3)
```

#End of code for Time Series Analysis & Forecasting of Stock Prices using R for 10 years

R Code considering 3 years data for analysis:

```
library(quantmod)# Extract data from Yahoo Finance
library(qrmtools)
library(MASS)
library(PerformanceAnalytics)# Calculations that are performed on a stock/portfolio
library(TSA)
library(aTSA)
library(forecast)# Forecasting the time series
library(fBasics)# Basic Statistical Calculations
library(urca)# Stationarity Analysis
library(dlookr)
library("SmartEDA")# Exploratory Data Analysis
library(nortest)
library(DescTools)
library(TTR)
library(tidyverse)
library(neuralnet)
library('lattice')
library('randomForest')
library(fitdistrplus)
library(metRology)#Compare the performance of different distributions

# Download data from Yahoo Finance
HDFC<-getSymbols("HDFCBANK.NS",from="2020-12-31", To= "2023-11-10",
auto.assign=FALSE)
Nifty<-getSymbols("^NSEI",from="2020-12-31",To="2023-11-10",auto.assign=FALSE)

# Plotting the data downloaded from Yahoo Finance (Time Series Data)
summary(HDFC)# To understand any missing values in any columns
summary(Nifty)
HDFC<-na.omit(HDFC)# Removing the rows containing the missing data
Nifty<-na.omit(Nifty)

#Calculating returns of data
returns<-diff(log(HDFC$HDFCBANK.NS.Adjusted))
returns<-na.omit(returns)

Nifty_Returns<-diff(log(Nifty$NSEI.Adjusted))
Nifty_Returns<-na.omit(Nifty_Returns)
ret1<-merge(returns,Nifty_Returns)
ret1<-na.omit(ret1)
ret2<-as.data.frame(ret1)#converting data into data frame so as to use Desc function
```

```
#EDA - Calculating descriptive statistics for returns and price data
Desc(ret2)
HDFC1<-as.data.frame(HDFC$HDFCBANK.NS.Adjusted)
Desc(HDFC1)
Nifty1<-as.data.frame(Nifty$NSEI.Adjusted)
Desc(Nifty1)

# Auto correlation and partial auto correlation
acf(ret1$HDFCBANK.NS.Adjusted)
pacf(ret1$HDFCBANK.NS.Adjusted)
acf(ret1$NSEI.Adjusted)
pacf(ret1$NSEI.Adjusted)

acf(HDFC$HDFCBANK.NS.Adjusted)
pacf(HDFC$HDFCBANK.NS.Adjusted)
acf(Nifty$NSEI.Adjusted)
pacf(Nifty$NSEI.Adjusted)

#Box test for stationarity analysis
Box.test(ret1$HDFCBANK.NS.Adjusted, lag = 30, type = "Ljung-Box")
Box.test(ret1$NSEI.Adjusted, lag = 30, type = "Ljung-Box")
Box.test(HDFC$HDFCBANK.NS.Adjusted, lag = 30, type = "Ljung-Box")
Box.test(Nifty$NSEI.Adjusted, lag = 30, type = "Ljung-Box")

#Model Building using AUTO ARIMA
model1<-auto.arima(ret1$HDFCBANK.NS.Adjusted,max.p = 10, max.d=3, max.q = 20,
max.order = 20, trace = TRUE)
model2<-auto.arima(ret1$NSEI.Adjusted,max.p = 10, max.d=3, max.q = 10, max.order = 20,
trace = TRUE)
model3<-auto.arima(HDFC$HDFCBANK.NS.Adjusted,max.p = 10, max.d=3, max.q = 10,
max.order = 20, trace = TRUE)
model4<-auto.arima(Nifty$NSEI.Adjusted,max.p = 10, max.d=3, max.q = 10, max.order = 20,
trace = TRUE)

plot(ret1$HDFCBANK.NS.Adjusted)

#Model prediction for returns and price
summary(model1)
predict(model1,20)
forecast(model1, h = 20)
#RMSE/Predicted
0.01426/0.0000001
```

```
summary(model2)
forecast(model2, h = 20)
#RMSE/Predicted
0.009230436/-0.0007132400

summary(model3)
forecast(model3, h = 20)
#RMSE/Predicted
21.13312/1505.1

summary(model4)
forecast(model4, h = 20)
#RMSE/Predicted
153.28/19721.97

#Residuals and squared residuals Stationarity analysis
Box.test(model1$residuals, lag = 20, type = "Ljung-Box")
Box.test(model1$residuals^2, lag = 20, type = "Ljung-Box")
Box.test(model2$residuals, lag = 20, type = "Ljung-Box")
Box.test(model2$residuals^2, lag = 20, type = "Ljung-Box")
Box.test(model3$residuals, lag = 20, type = "Ljung-Box")
Box.test(model3$residuals^2, lag = 20, type = "Ljung-Box")
Box.test(model4$residuals, lag = 20, type = "Ljung-Box")
Box.test(model4$residuals^2, lag = 20, type = "Ljung-Box")
dev.off()
#Modeling using AUTO ARIMA with power transformation
model11<-auto.arima(ret1$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE, lambda="auto")
summary(model11)
forecast(model11, h = 20)
0.01417566/-0.0001191788

model21<-auto.arima(ret1$NSEI.Adjusted,max.p = 10, max.d=2, max.q = 10, max.order = 20,
trace = TRUE, lambda="auto")
summary(model21)
forecast(model21, h = 20)
0.009207518/-3.184620e-04

model31<-auto.arima(HDFC$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE, lambda="auto")
summary(model31)
forecast(model31, h = 20)
21.0107/1505.106
```

```
model41<-auto.arima(Nifty$NSEI.Adjusted,max.p = 10, max.d=2, max.q = 10, max.order = 20,
trace = TRUE, lambda="auto")
summary(model41)
forecast(model41, h = 20)
154.3305/19728.95

# ARIMA with BoXCox
y1<-c()
for (i in 1:100) {
  model1<-auto.arima(ret1$HDFCBANK.NS.Adjusted[i:(i+600)],max.p = 10, max.q = 10,
max.order = 20, lambda="auto")
  y<-as.vector(predict(model1,1)$pred)
  y1<-c(y1,y)
}
# ARIMA without BC
y2<-c()
for (i in 1:100) {
  model2<-auto.arima(ret1$HDFCBANK.NS.Adjusted[i:(i+600)],max.p = 10, max.q = 10,
max.order = 20)
  y<-as.vector(predict(model2,1)$pred)
  y2<-c(y2,y)
}
RMSE(y1,tail(ret1$HDFCBANK.NS.Adjusted,100))
RMSE(y2,tail(ret1$HDFCBANK.NS.Adjusted,100))

# Fitting appropriate distributions to the returns data
par(mfrow = c(1,1))
model14<-auto.arima(ret1$HDFCBANK.NS.Adjusted,max.p = 10, max.d=2, max.q = 10,
max.order = 20, trace = TRUE)
Desc(as.vector(model14$residuals))

normaldist<-fitdist(as.vector(model14$residuals),distr = "norm")
logisdist<-fitdist(as.vector(model14$residuals),distr = "logis")
cauchydist<-fitdist(as.vector(model14$residuals),distr = "cauchy")
tdist<-fitdist(as.vector(model14$residuals),distr =
"t.scaled",start=list(df=3,mean=mean(as.vector(model1$residuals)),sd=sd(as.vector(model1$resi
duals))))

summary(normaldist)
summary(logisdist)
summary(cauchydist)
summary(tdist)
```

```
#Goodness of fit tests
gofstat(list(normaldist, logisdist, cauchydist, tdist), fitnames = c("Normal Distribution",
"Logistic Distribution", "Cauchy Distribution", "t Distribution"))

# Monte carlo simulations
set.seed(1000)
x<-rnorm(10000,mean = 0.0001038853 , sd = 0.0142606189 )
quantile(x,c(0.01,0.05))
x1<-rlogis(10000,location=0.000109081 , scale=0.007340606)
quantile(x1,c(0.01,0.05))
x2<-rcauchy(10000,location=0.0002202082 , scale=0.0068455062 )
quantile(x2,c(0.01,0.05))
x3<-rt(10000,df = 1)
quantile(x3,c(0.01,0.05))

#calculating t values
-0.01745498 + 0.67119016*-31.022523
-0.01745498 + 0.67119016*-6.196629

#======= Exploring ML models for stock price prediction=====
dim(HDFC)
dim(Nifty)
# data splitting into training and test data
train_data1<-HDFC[1:600,]
test_data1<-HDFC[601:714,]
train_data2<-Nifty[1:600,]
test_data2<-Nifty[601:714,]

#Splitting the test data for evaluation
X_test<-test_data1[,-6]
Y_test <-test_data1[,6]

#forecasting using Random Forest ML model
fit1<-randomForest(HDFCBANK.NS.Adjusted ~ .,data=train_data1)
fit1
final_predict<-predict(fit1,X_test)
dev.off()
plot(final_predict)
## calculate the rmse value
rmse1 <- sqrt(mean(as.matrix(log1p(Y_test) - log1p(final_predict))^2))
print(paste('RMSE:', rmse1))

#prediction and comparing with actual test values
```

```
prediction_actual=data.frame(HDFCBANK.NS.Adjusted=final_predict, Y_test, (Y_test-
final_predict)*100/Y_test)
tail(prediction_actual)
#write.csv(prediction_data_frame1,file = "FINAL.csv" ,row.names=FALSE)

#Neural Network Time Series Forecasts
L <- BoxCox.lambda(train_data1$HDFCBANK.NS.Adjusted, method="loglik")
fit_nn <- nnetar(train_data1$HDFCBANK.NS.Adjusted, lambda=L, size=3)
fit_nn
fcast_nn <- forecast(fit_nn, h=114, lambda=L)

mape <- function(real, forecast){
  len <- length(real)
  return(sum( abs(real - forecast$mean[1:len]) / real) / len * 100)
}
paste0('Error Neural Network: ', mape(test_data1$HDFCBANK.NS.Adjusted, fcast_nn), '%')

#prediction and comparing with actual test values
prediction_actual1=data.frame(HDFCBANK.NS.Adjusted=fcast_nn, Y_test)
tail(prediction_actual1)

## Nifty Prediction using ML Models
#splitting the test data for evaluation
X1_test<-test_data2[,-6]
Y1_test <-test_data2[,6]

#forecasting using Random Forest ML model
fit2<-randomForest(NSEI.Adjusted ~ .,data=train_data2)
fit2
final_predict1<-predict(fit2,X1_test)
dev.off()
plot(final_predict1)

## calculate the rmse value using two methods
rmse1 <- sqrt(mean(as.matrix(log1p(Y1_test) - log1p(final_predict1))^2))
print(paste('RMSE:', rmse1))

prediction_actual2=data.frame(NSEI.Adjusted=final_predict1, Y1_test, (Y1_test-
final_predict1)*100/Y1_test)
tail(prediction_actual2)
tail(Y1_test)

#Nifty - Neural Network Time Series Forecasts
```

```
L <- BoxCox.lambda(train_data2$NSEI.Adjusted, method="loglik")
fit.nn <- nnetar(train_data2$NSEI.Adjusted, lambda=L, size=3)
fit.nn
fcast_nn1 <- forecast(fit.nn, h=114, lambda=L)

mape <- function(real, forecast){
  len <- length(real)
  return(sum( abs(real - forecast$mean[1:len]) / real) / len * 100)
}
paste0('Error Neural Network: ', mape(test_data2$NSEI.Adjusted, fcast_nn1), '%')

#prediction and comparing with actual test values
prediction_actual3=data.frame(NSEI.Adjusted=fcast_nn1, Y1_test)
tail(prediction_actual3)
```

#End of code for Time Series Analysis & Forecasting of Stock Prices using R for 3 years

CHAPTER 8: ABBREVIATIONS

ACF	Autocorrelation Function
AD	Anderson-Darling
ADF	Augmented Dickey-Fuller tests
AI	Artificial Intelligence
AIC	Akaike Information Criterion
AR	autoregressive models
ARCH	Autoregressive Conditional Heteroskedasticity
ARIMA	autoregressive integrated moving average models
BIC	Bayesian Information Criterion
CEO	Chief Executive Officer
CDF	cumulative distribution function
CVM	Cramér-von Mises
EDA	Exploratory Data Analysis
GARCH	Generalized Autoregressive Conditional Heteroskedasticity
GRU	Gated recurrent unit
KS test	Kolmogorov-Smirnov test
LL	Log likelihood
LSTM	Long Short-Term Memory
MA	moving average models
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
ML	Machine learning
MSE	Mean Squared Error
NN	Neural Network
NSE	National Stock Exchange of India
OHLC	Open High Low Close
p	Probability
PACF	Partial Autocorrelation Function
QQ	Quantile-Quantile
RF	Random Forest
RMSE	Root Mean Squared Error

CHAPTER 9: REFERENCES AND BIBLIOGRAPHY

Books

- Business Analytics Book by MIT School of Distance Education
- Data Mining for Business Analytics Book by MIT School of Distance Education
- Financial Analytics Book by MIT School of Distance Education
- Predictive Modelling Book by MIT School of Distance Education
- Financial Analytics with R: Building a Laptop Laboratory for Data Science by Mark J. Bennett and Dirk L. Hugen
- Hyndman, R.J., & Khandakar, Y. (2008). Automatic time series forecasting: the forecast package for R. Journal of Statistical Software, 27(3), 1-22.
- McNeil, A.J., Frey, R., & Embrechts, P. (2015). Quantitative Risk Management: Concepts, Techniques, and Tools. Princeton University Press.
- Mark P.J. van der Loo & Edwin de Jonge, P. (2012). Learning RStudio for R Statistical Computing. Packt Publishing Ltd

Websites

- https://finance.yahoo.com/quote
- ARIMA Model - Complete Guide to Time Series Forecasting in Python | ML+ (machinelearningplus.com)
- https://online.stat.psu.edu/stat510/lesson/1;
- https://towardsdatascience.com/analyzing-stocks-using-r-550be7f5f20d
- https://www.youtube.com/watch?v=qXoq6Lqb684
- https://www.kaggle.com/code/hangduong06/time-series-analysis-and-forecasting
- https://www.kaggle.com/code/sachhidanandmanjhi/stock-price-prediction-of-reliance-ind