TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Master Thesis

on

Prediction of Volatility of Stock Price using Tree Based Regression Model

At the,

Georg-Simon-Ohm University of Applied Sciences Nuremberg

Faculty of Business Administration

Degree Program: International Finance and Economics (M.Sc.)

Submitted By: Ajith Rajan

Matriculation No: 3231781

Professor and Mentor: Dr. Hans Dieter Gerner

Submission Date: 31 October 2022

Master Thesis

Prediction of Volatility of Stock Price using Tree Based Regression Model

Abstract

Volatility forecasting is a vital task for asset valuation and risk management for investors and financial intermediaries. Various volatility forecasting models have been introduced and practiced over the period. However, the demand for better prediction models is always there as the financial markets update itself on a continuous basis. The purpose of this thesis is to examine the volatility forecasting performance of different Tree Based Regression models with GARCH models on NIFTY50 stock index. The GARCH models examined in this thesis are SGARCH, EGARCH and GJR-GARCH. And the Tree Based Regression models examined are Bagging, Random Forest and Gradient Boosting. The statistical loss functions used for evaluating the performance of the models are MSE and RMSE. The result of this thesis is that Tree Based Regression models have better prediction accuracy compared to GARCH models. Within Tree Based Regression Models Gradient Boosting method outperformed all other models with least MSE and RMSE. This is because of the fact that Gradient Boosting approach utilizes sequential training by constructing an ensemble of shallow and weak consecutive trees, each of which learns from and improves upon itself.

Key words: Volatility, GARCH, Bagging, Random Forest, Boosting, MSE, RMSE

Submitted By: Ajith Rajan

Matriculation No: 3231781

Supervisor: Prof. Dr. Hans Dieter Gerner

Technische Hochschule Nürnberg

Georg Simon Ohm

Faculty of Business Administration

Table of Contents

Chapter 1
Introduction

1.1 Motivation

Predicting stock prices has always been an interesting topic since the inception of stock markets since it is valuable as one can make both financial gains out of it and have insight on the economic situation of a company or of an industry or a country at large. Accurate stock prediction is extremely challenging; reason being that the stock market is highly volatile. However, many volatility prediction models have been introduced and practiced over the period. Some of the models have created fortunes for the investors and this in turn motivated people to update existing models or create new models for better prediction accuracy.

Machine Learning is a sub field of computer science that allows software applications to find patterns, generalize, learn, and predict outcomes without being explicitly programmed. Machine learning is widely used in recent years for predicting stock markets. Stock markets are highly volatile and numerous factors influence the volatility of stock volatility. This makes Machine Learning a highly demanded tool for predicting stock volatility since it can deal with vast amounts of data and can give accurate outcome based on the algorithm. Some have applied Machine Learning to the National Stock Exchange of India (NSE), the largest stock exchange in India. However, still there is scope for improvement in prediction accuracy.

1.2 Aim

The main purpose of this thesis is to predict the volatility of the stock market using the symmetric SGARCH model and asymmetric EGARCH and GJR GARCH model, and to compare with Decision Tree Regression Models like Bagging, Random Forest and Boosting. The performance measures Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are used to evaluate the performance of different models.

Chapter 2
Theory

2.1 Volatility in Financial markets

The rate at which the price of a stock rises or falls over a specific period is known as stock price volatility. If the volatility is large, the probability of a large positive or negative return is high. Stock market crash of 1987 led to numerous studies on various investment techniques on the volatility of share price (Schwert, 1990).

Forecasting Volatility is a crucial task in the financial markets, and it has captured the interest of investors and academics alike. Risk and volatility are not the same thing. When seen as uncertainty, it becomes a crucial factor in many investing choices and portfolio constructions. Certain amount of risk is acceptable to portfolio managers and investors. As a starting point these professionals rely on forecast of the volatility of the asset prices for the purpose of evaluating investment risk (Poon & Granger, 2003).

Even while the stock market's behaviour is linked to the most important macroeconomic indicators of the economy, it is unclear, volatile, and probabilistic. Robust capital markets with strong macroeconomic fundamentals are essential for stability of the stock market. The capital market becomes unstable because of the stock market's excessive volatility, which also will affect the value of currencies and hinders international trade and finance (Bhowmic, 2013).

Volatility can either be measured by using the annualized standard deviation of daily changes in stock price or variance between returns of same stock or market index. Taking into account a time series of returns r_t with t, the time variable that has been given in the sample as t = 1, 2, ... , T. The standard deviation of returns is calculated by applying the following formula (Poon, 2005).

$$\sigma = \frac{\sqrt{1}}{T-1} \sum_{t=1}^{T} (r_t - \mu)^2 \qquad (1)$$

The variance of the returns is calculated using the formula (Poon, 2005).

$$\sigma^2 = \frac{1}{T-1} \sum_{t=1}^{T} (r_t - \mu)^2 \qquad (2)$$

In this thesis for computing return of security it is assumed, the stock market returns are continuously compounded over time. The return of the specific asset is computed using the formula.

$$r^t = \ln\left(\frac{S_t}{S_{t-1}}\right) \qquad (3)$$

2.1.1 Volatility Modelling

Predicting the volatility of stock is the central requirement of volatility modelling. Having a clarity on how and why these movements occur is important for many areas of economic theory, especially asset pricing, and is of key importance for participants in capital market (Rossie & Magistris, 2013).

In the paper "What is a good volatility model" written by Engel and Patton (2001), the authors implies that a volatility model should be able to determine the relevant quantiles of returns as well as the future features of those returns. A robust volatility forecasting model, in the opinion of the authors, should take into account a particular set of stylized facts about the volatility of a financial asset prices that have arisen and been supported by several studies (Engel & Patton, 2001).

There exist different techniques to estimate volatility. Looking into historical volatility is one approach which is easy to forecast volatility. Under this method of estimation, past standard deviation/variance of returns over a period is considered in order to predict the future volatility (Brooks, 2014).

According to studies realized volatility is another method that can be used for estimating volatility. Realized volatility is a measure of return variability for an investment product based on an examination of past performance over a predetermined time frame. It is two factors that affect stock prices and causes realized volatility or actual volatility, in the market – a continuous volatility factor and a jump factor. Continuous volatility in share market is influenced by the intra-day trading volumes. Traders make use of high- frequency intraday data to determine measures of volatility at different frequency. The data could be utilized to predict the volatility in returns (Srivastav, 2022).

Auto regressive (AR) volatility models are another widely used techniques to estimate volatility. Under autoregressive process it is assumed that current values are related to past values. In other words, it is presumed that past values have an effect on current values. Since AR model linearly depends on previous terms it is basically a regression model where the past values are predictors. Auto regressive models evolved over the period and have become pillars for modelling financial markets that show volatility (Agung, 2008).

Moving average models account for the fact that returns rely on signals that came in over a past period in addition to information that is currently available. This may occur if latest information is only slowly assimilated or gets into different market players at separate times. As a result, any new signal has an impact that is both immediate and delayed (Guidolin & Pedio, 2018).

The ARMA model is a model which is a combination of AR and MA models. Impact of previous lags along with the residuals is taken into consideration for predicting future values of the time series. Volatility clustering is not considered under ARMA model there by it is not a conditionally heteroskedastic model (Fabozzi et al., 2014). The conditional volatility assumption was included into the autoregressive conditional heteroskedasticity (ARCH) model to improve econometric models. An alternative to using a constant or average for volatility was suggested by ARCH model.

2.1.2 ARCH Model

The ordinary least squared regression method makes the assumption that the squared anticipated value of error terms is the same at every point, which is termed as homoskedasticity. This is the pivotal point of focus of ARCH/GARCH models. Heteroskedasticity is the term used to describe data that have unequal variances in the error terms, meaning that it is reasonable to expect at some points or ranges of the data to have greater error terms than others (Engle, 2001).

The Autoregressive Conditional Heteroscedasticity (ARCH) is the model that initially offered a systematic framework for modelling time-varying conditional variance, developed by Engle (1982). The ARCH processes, in contrast to conventional models with constant variance, uses lagged disturbances to account for the time-varying conditional variance of financial time-series. This makes the ARCH model one of the initial models to successfully capture persistence in its forecasts (Poon, 2005). The basic expression of ARCH (q) model is as follows:

$$\sigma_t^2 = \omega + \sum_{j=1}^{q} \alpha_j \epsilon_{t-j}^2 \qquad (4)$$

, where $\omega > 0$ and $\alpha_j \geq 0$.

2.1.3 GARCH model

Generalization of ARCH model was introduced by Bollerslev (1986) and Taylor (1986) known as GARCH model. In GARCH (p, q) model for the conditional variance, forecasts depend on a (non-negatively) weighed sum of past squared residuals and past variance forecasts. The general expression of GARCH (p, q) model is as follows:

$$\sigma_t^2 = \omega + \sum_{i=1}^{p} \beta_i \, \sigma_{t-k}^2 + \sum_{j=1}^{q} \propto_j {\varepsilon^2}_{t-j} \tag{5}$$

where α + $\beta_i < 1$. For GARCH (1,1), the constraints $\alpha_1 \geq 0$ and $\beta_1 \geq 0$ are needed to ensure σ_t^2 is strictly positive (Poon, 2005).

GARCH model is sub-divided into Symmetric model and Asymmetric model. As per symmetric model good news and bad news have same effect. Empirical studies shows that symmetric GARCH models fails to capture the leverage effect, where bad news have more effect on the volatility of stock than good news. Asymmetric GARCH model was initially proposed by Nelson (1991), Exponential GARCH (EGARCH) to overcome the drawbacks of symmetric models (Arachchi, 2018).

i) EGACH model

The exponential GARCH (EGARCH) model is introduced by Nelson (1991). Volatility clustering and leptokurtosis can be modelled by symmetric ARCH and GARCH models. However, these models fail to capture the asymmetric relationship between asset returns and volatility changes which is a key requirement while dealing with time series data (Tsay, 2013). The EGARCH (p,q) model is given by (Y. Wang et. al., 2021):

$$\ln(\sigma_t^2) = \omega_+ \sum_{i=1}^{p} \alpha_i \left| \frac{\mu_{t-i}}{\sigma_{t-i}} - E\left(\frac{\mu_{t-i}}{\sigma_{t-i}}\right) \right| + \sum_{j=1}^{q} \beta_j \ln\left(\sigma_{t-j}^2\right) + \sum_{k=1}^{r} \gamma_k \frac{\mu_{t-k}}{\sigma_{t-k}} \quad (6)$$

In this model there is no requirement to impose an estimation constraint in order to avoid negative variance since the EGARCH (p,q) model specifies conditional variance in logarithmic form. As long as there is unsymmetrical term $\gamma \neq 0$ in the formula, its effect is measured by an exponential form and not in a quadratic form (Y. Wang et. al., 2021).

ii) GJR GARCH model

For modelling asymmetric volatility one of the other models that is commonly in use is GJR GARCH model which was developed by Glosten, Jagannathan & Runkle (1993). Compared to EGARCH, this model is much simpler for practical application as the variance is directly modelled and does not use the natural logarithm (Hayashi, 2000).

The GJR-GARCH (1,1) model equation is as given below (Asgharian, 2016):

$$\sigma_t^2 = \omega + \alpha_1 n_{t-1}^2 + \alpha_2 l_{t-1} n_{t-1}^2 + \beta \sigma_{t-1}^2 \quad (7)$$

, where

$\alpha_1 n_{t-1}^2$**:** is the variance that depends on previous lag error terms.

l_{t-1} : is the dummy variable

Dummy variable l_{t-1} gets activated only when the previous shock is negative, taking into consideration the leverage effect. A negative shock is captured by $(\alpha_1 + \alpha_2)$ and positive shock by α_1 (Asgharian, 2016).

If, α_2 = 0, symmetry (no asymmetric volatility).

$\alpha_2 > 0$, negative shock will increase the volatility more than positive shocks.

$\alpha_2 < 0$, positive shock will increase the volatility than negative shocks.

2.2 Machine Learning

Machine learning is a branch of Artificial Intelligence that systematically applies algorithms to synthesize the underlying relationships among data and information (Mariette & Rahul,2005).

Tom M. Michell defined ML as: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E" (Mitchell, 1997).

The field of computer science where raw data is transformed into intelligent action by applying algorithms is known as Machine Learning. This area of study revolves around an environment where available data, statistical methods and computing power is swiftly and concurrently evolving. Additional computing power was in demand when data grew exponentially, which in turn led to the development of statistical methods for scrutinizing huge datasets. This generated a cycle of advancement allowing this field of study to deal with big data (Brett Lantz, 2013).

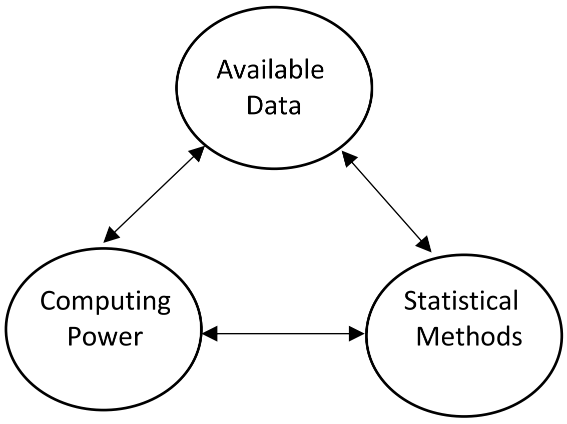


Figure: 1. Machine Learning Development cycle

Machine Learning is a significant task, and it flourishes in an abundant data environment. "Signal-to-noise-ratios" in available data is a major factor that determines the success of machine learning. The signal-to-noise ratio identifies the degree of predictability in a system. Some systems have a high signal-to-noise ratio and are hence highly predictable by nature. Others have low signal-to-noise ratios because randomness is dominant in them. Success has been reached on a variety of data and problems. Among them, machine learning has been applied to predict financial time series with success (Israel et al., 2020).

2.2.1 Raw data and building models

A necessary condition for learning is that there be material from which to learn; examples are required, which is the data collected. For a machine to learn and to build a successful model, it is indispensable that the data is relevant. The calibre of the data it uses determines the quality of the Machine learning project. Once the data is collected it is usually separated into two groups. The training set and Test set are the names of these sets of data. The data which is used to develop the model is known as the training set. By matching the input with the anticipated result and looking for relationships between the variables to reduce error, this set is utilized to fit the machine learning model. The fitting procedure is repeated until the model's error minimization has reached a certain minimal level. The Test set is used once the model has been fitted. As the Test set is not visible to the model, we can use it to obtain a final assessment of how well the Machine Learning model fits the data; this assessment should reveal how well the model will perform on real-world data. More sophisticated methods must be used to boost the model's performance if improvement in the model is required (Breet Lantz, 2013).

Supervised learning or unsupervised learning are the one of the two groups where most statistical problems fall into. Statistical learning methods such as linear regression and logistic regression, as well as modern approaches such as GAM, boosting, bagging and support vector machines, operates in supervised learning domain (James et al., 2021).

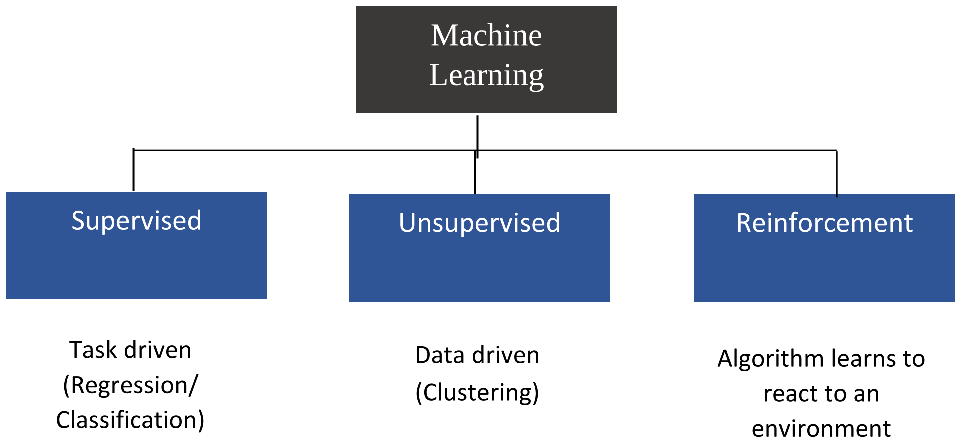

Machine Learning
Supervised
Unsupervised
Reinforcement
Task driven (Regression/ Classification)
Data driven (Clustering)
Algorithm learns to react to an environment

Figure: 2. Different types of Machine Learning techniques

2.2.2 Supervised learning

The field of Machine learning is wide and includes many types of decision-making and pattern recognition algorithms. In supervised learning, the objective is to learn the relationship between X and Y from the labelled data, such as pairs (x1, y1), . . . , (xn, yn), x1,...,xn ∈ X, y1,...,yn ∈ Y. Each observation xi is considered as feature vector and Yi as label or response. In other words, for supervised learning initially a set of data is provided which consists of input and target data. This data set is usually named as the training data. Learning the correlation between input and target variables is the primary part of supervised learning for prediction (Dixon et al., 2020).

We can categorize supervised learning into two subgroups: Predicting discrete categories for classification. Forecasting continuous values with regression.

2.2.3 Classification

Building a model using a training set of database instances and associated class labels is known as classification. When the values of the predictor characteristics are known, the model created can then be used to predict the class label of testing cases. Classification is known as supervised learning since the examples are given with known labels, in contrast to unsupervised learning where the labels are not provided. Classification can be used to generate models representing significant data classes or to forecast future data trends. Classification technique of analysis can provide a better grasp of vast dataset (Bhavsar & Ganatra, 2012).

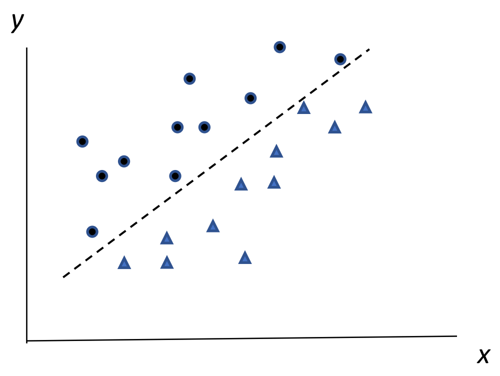


Figure: 3. A model of Classification

2.2.4 Regression

Regression is a supervised machine learning technique that aims at predicting values by building a learning model and tries to establish relationship between input and output variables. Simple linear regression model is used when there is only one explanatory variable, and multiple linear regression is used when there exists more than one explanatory variable. A straight line representing a linear relationship between the output variable and the input variable or variables is what the linear regression is always looking for. It is always assumed in linear regression that there exists a linear relationship between input and output variables, even though it is not always the case. In case of a nonlinear relationship between input and output variables polynomial regression model is used for predictions (Sakhare & Imambi, 2019).

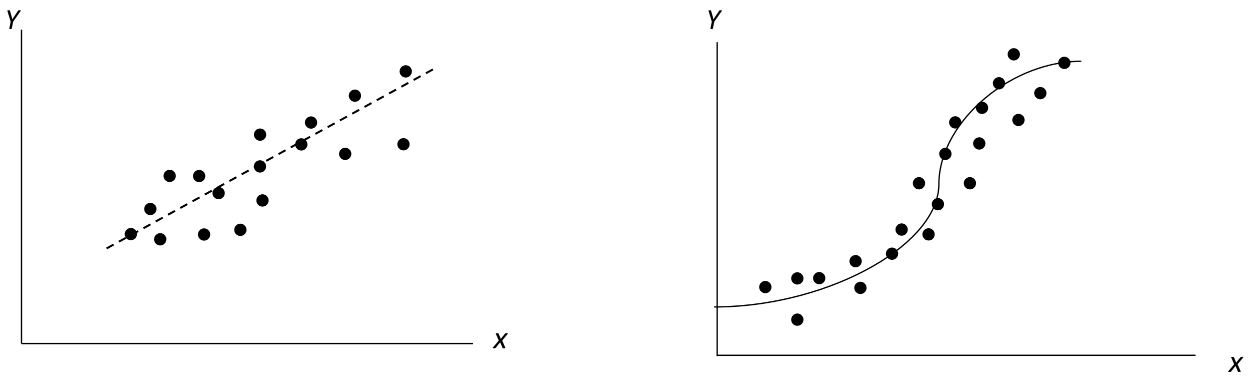


Figure: 4. Left: Linear Regression model, Right: Polynomial Regression model

2.2.5 Problem of over- and under-fitting

Over-fitting and under-fitting are the most prominent problems faced while performing Machine Learning. This is due to noise that the problem of over- and under-fitting occurs. Over-fitting is the situation when the model completely fits the training data but fails to generalize the testing data that is unknown. When a Machine Language fails to model and generalize the training data for prediction it is termed as under-fitting (James et al., 2021).

2.2.6 Cross - Validation

Validation is the process of determining whether numerical results are acceptable that quantify the supposed relationships between variables. Cross-validation is a statistical technique for assessing and contrasting learning algorithms that involves splitting the data into two sections: one for learning or training a model and the other for validating the model. The training and validation sets must overlap in subsequent rounds during a typical cross-validation so that every data point gets a chance to be validated against. K-fold cross-validation is the most fundamental type of cross-validation.

In k-fold validation, the data is initially divided into k segments or folds of equal size. In the next step k iterations of training and validation are performed, with each iteration holding out a different fold of the data for validation. Other variety of cross-validation techniques are variations on k-fold cross-validation or involves iterations of k-fold cross-validation (Liu & Ozsu, 2009b)

2.2.7 Time series Analysis

Time series analysis is used to examine time series data. Time series data are sequence of events or phenomena that happens over time. Since the observations depend on time, time series analysis differs from other kinds of analysis. Some considerations are required when performing machine learning using such data. One key requirement of time series data is – its values need to be captured at equally spaced time periods, such as seconds, minutes, hours, days, months, and so on (Krispin, 2019).

The sequence of random variables that moves through time discloses a certain behaviour and is explained as a stochastic process and serves as a model for an observed time series. For a stochastic process the mean function is defined by)

$$\mu = E(Yt) \quad \text{for t = 0, ±1, ±2,..} \tag{8}$$

The unconditional variance

$$\sigma_t^2 = E[(yt - \mu)^2] \tag{9}$$

And the autocorrelation function, $\gamma_{t,\tau}$ is given by

$$\gamma_{t,\tau} = E[(y_{t} - \mu_t)(y_{t-\tau} - \mu_{t-\tau})] \tag{10}$$

If the mean and variance are constant over time, and the value of covariance between two time period depends only on a gap or distance or lag between two time periods the stochastic process is termed to be stationary. Main precondition for time series analysis is stationarity.

2.3 Machine Learning Techniques for time Series Analysis

Machine Learning Techniques for Time Series Analysis that are used in practical world for various purposes includes Stock price forecasting, Demand and sales forecasting, Climate and weather forecasting, Demographics and Economics forecasting, web traffic forecasting. Some of the key Machine Learning techniques used for Time series forecasting are explained below, out of which this thesis paper will focus mainly on Decision Tree method.

2.3.1 Decision Tree

Decision tree method involves categorizing and subdividing the predictive space into number of simple groups to make prediction for the given observation. Mean or mode response value for the training observations in the group to which it belongs is used for prediction. This method is termed as “decision tree” because the set of splitting rules used to divide the predictor space can be condensed into a tree. Techniques like bagging, random forest, boosting and Bayesian additive regression trees could be used to make the prediction more accurate (James et. al., 2021).

- **Basic concepts**

Nodes and branches are the key structural components of a decision tree.

Nodes: Nodes can be of three different sorts. (a) A root node, also known as decision node, represents a decision that will divide the records into two or more sets which are mutually exclusive. (b) Internal nodes, also known as chance nodes, shows one of the options that might be accessible at that particular moment in the tree structure; the node’s top edge is connected to the parent node, and its bottom edge to its child nodes or leaf nodes. (c) Leaf nodes, also known as end nodes, reflect the outcome of a string of choices or occurrences.

Branches: The root nodes and internal nodes are represented by branches, which stand for random events or occurrences. Using a hierarchy of branches, a decision tree model is created. Every route from the root node through internal nodes to a leaf node symbolizes a classification decision rule (Song & Lu, 2015).

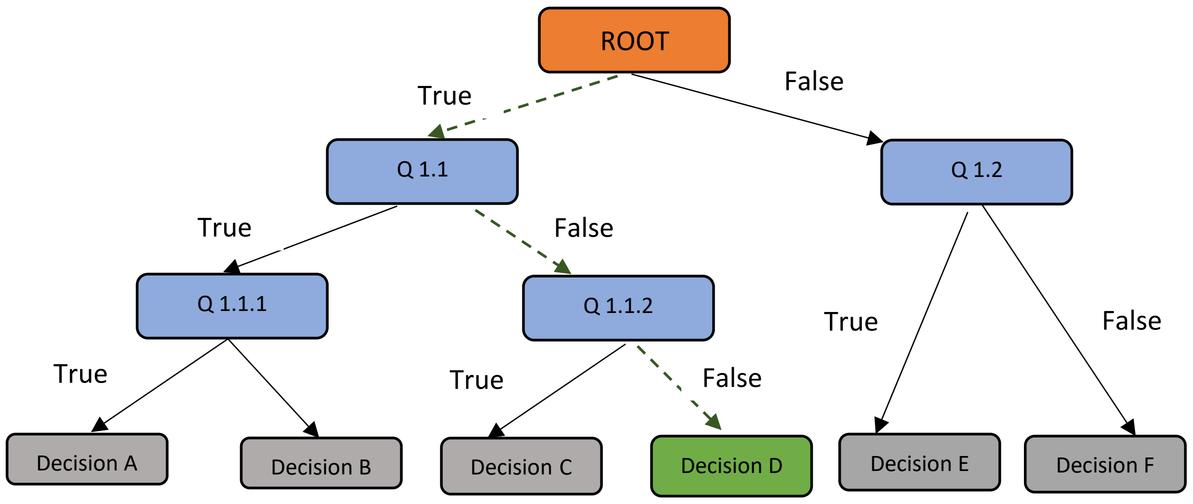


Figure: 5. A decision tree model – Starting from root node, multiple criteria are selected based on the information from each input. A decision is drawn at a particular leaf, i.e., Decision D, if all criteria along its path "==" are satisfied (Luong & Dokuchaev, 2018).

- **Tree Pruning**

Pruning is a technique used to address overfitting, one of the major problems while doing Machine Learning. In this method initially a large tree is created and then pruned it by removing nodes to reach optimal size to improve prediction accuracy by reducing overfitting (Song & Lu, 2015).

2.3.2 Logistic Regression

Method for modelling a binary response variable by taking two discrete values - "0" or "1" is called Logistic Regression. This regression method looks into the probability of an event.

For instance, pass or fail, based on the given set of data of independent variables. Here the dependent variable is bounded between 0 and 1 since the outcome is a probability.

This regression method is a part of the supervised learning model family in the context of Machine Learning. It is also regarded as a discriminative model because it makes an effort to distinguish between different groups. Contrary to generative algorithms like naive bayes, it is unable or fail to produce information of the classes that is attempting to predict, such as an image (e.g., A picture of a pet) (What is Logistic Regression? | IBM)

2.3.3 Random Forest

Random forest is an ensemble machine learning method. To create a forest like collection of classifiers, the method initially generates several decision tree classifiers one by one. Each tree in a random forest depends on the values of a random vector that was samples randomly and with the same distribution for all the trees in the forest. Once a huge number of trees are generated, next step involves the voting for the most prominent class. These processes are called Random Forest.

Random forest is defined as: “A random forest is a classifier consisting of a collection of tree-structured classifier $\{h(x, \theta_k),\ k = 1, \ldots\}$ where the $\{\theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x “ (Breiman, 2001).

2.3.4 Bagging

A technique for creating several iterations of a predictor and using these to produce an aggregated predictor is called bagging. When forecasting a numerical conclusion, the aggregate takes an average across all variants, and when predicting a class, it performs a plurality vote. By creating bootstrap copies of the learning sets and using these as brand-new learning sets, many versions are created. Bagging can result in significant improvement in accuracy, according to tests on real and simulated datasets utilizing classification and regression trees, as well as subset selection in linear regression. Bagging can increase accuracy if perturbing the learning set can result in noticeable changes to the predictor that was built (Breiman, 1996).

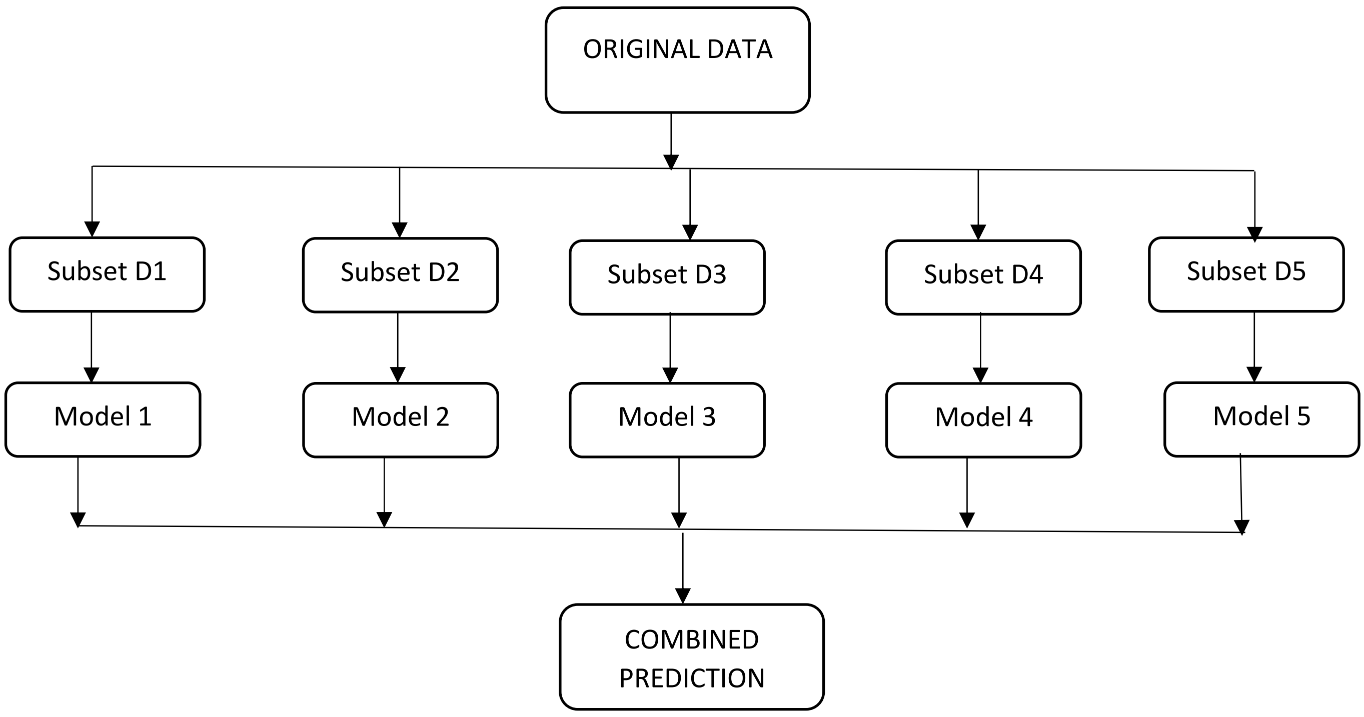


Figure: 5. Model of Bagging technique

2.3.5 Boosting

Boosting is a generalized technique used to enhance any learning algorithm's accuracy. Bagging uses simple majority voting among the classifiers, whereas boosting makes use of weighted majority voting mechanism. The boosting method creates base models consecutively in contrast to bagging. By focusing on these tough to estimate training events, multiple models are developed sequentially to increase prediction accuracy. During the boosting process, cases that are challenging to estimate with the prior base models show up in the training data more frequently than cases that can be accurately estimated. Every new base model aims to fix the errors created by the previous base models (Zhang & Haghani, 2015).

Initially boosting had many practical difficulties, out of which many of the difficulties were solved by *AdaBoost* algorithm, proposed by Freund and Schapire in 1995 (Denison et al., 2003). Other boosting techniques that are discussed in this paper are gradient boosting and XGBoost.

AdaBoost (adaptive boosting) algorithm aims at generating a strong classifier from a set of weak classifiers. AdaBoost technique maintains a collection of weights over training data and adaptively modifies them after each weak training cycle in order to produce a set of poor learners. Training sample that the present weak learner misclassifies will have their weights increased, whereas training samples that are successfully categorised will have their weights reduced (Wang, 2012).

Gradient boosting is a Machine language method that generates a prediction model in the form of a collection of weak prediction models, often decision trees. Similar to other boosting techniques, it builds the model in different steps. And then it generalizes them by enabling the optimisation of arbitrary differential loss function (Fafalios et al., 2020). In further studies it is shown that both approximation and execution speed of gradient boosting could be significantly improved by integrating randomization into the procedure (Friedman, 1999). The Extreme gradient Boosting algorithm, often known as *XGBoost* is an open-source software library that provides gradient boosting framework packaged in different languages, like C++, Java and Python.

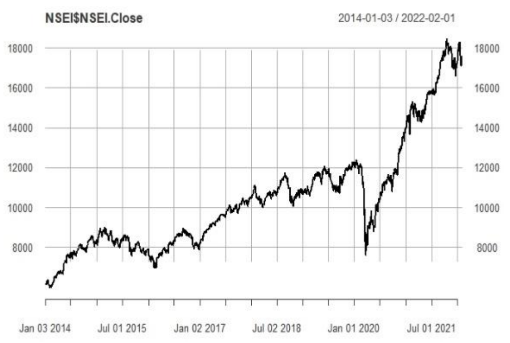
Chapter 3
Methodology and Empirical Evaluation

3.1 Data Collection and Methodology

For the purpose of the study the data used consist of daily closing price of Indian Stock Market Index NIFTY 50 (NSEI). It consists of the weighted average of the 50 biggest Indian companies listed on the National stock Exchange. The data is collected from Yahoo Finance, the time period is ranging from 2014-01-01 to 2022-02-02, consist of a total of 1981 observations. For the purpose of the study the data is divided into two sets, training dataset consisting of 1959 observations, from 2014-01-01 to 2022-01-01 and testing dataset consisting of 21 observations, from 2022-01-01 to 2022-02-02. The observations are then used to create the daily log returns.

$$r_t = \log(s_t^c) - \log(s_{t-1}^c) \tag{11}$$

Where s_t^c is the closing price at time t for NSEI. The daily log return is used for evaluation from the training dataset and for forecasting the volatility in the testing dataset. The GARCH models that are used to capture stock market volatility are SGARCH, EGARCH and GJR GARCH. And the decision tree models used are Bagging, Random Forest and Boosting.



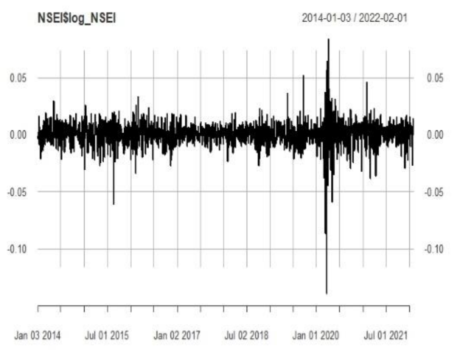


Figure: 6 shows the daily closing price and daily log-returns of NIFTY 50 index.

3.1.1 Diagnostic Tests

For modelling time series data, the important criteria is stationarity. In order to make the time series stationary daily log returns is taken. In order to evaluate the stationarity of the time series Augmented Dickey-Fuller test and Phillips-Perron Unit Root test was conducted. The test result showed that the time series data is stationary. The null hypothesis is rejected as the p-value is below the 5% significance level. The presence of volatility clustering can be seen from Figure.6-NSEI_Log_returns plot. The presence of high volatility can be seen during the period February 2020 to July 2020, showing the price fluctuations during the covid pandemic.

Dickey Fuller Test		Phillips-Perron (PP) Test	
Null Hypothesis	non stationary	Null Hypothesis	unit root or non-stationary
Data	trainingdf$log_NSEI	Data	trainingdf$log_NSEI
p-value	0.01	p-value	0.01

Table:1 ADF and PP Test Output

In order to check the ARCH effect ARCH-LM test was performed. The Autoregressive Conditional Heteroscedasticity-Lagrange Multiplier (ARCH-LM) test by Engle (1982) is performed to test the presence of ARCH effect in the residuals in the testing dataset. The presence of volatility clustering can be detected with the ARCH-LM test. The hypothesis of ARCH-LM test is H_0: No ARCH(q) effects in the residuals, $H_\propto$: Presence of ARCH(q) effect in the residuals. The given time series is appropriate for GARCH models if ARCH effects are spotted. H_0 can be rejected for NIFTY50 index, showing the presence of ARCH effect. The results can be seen in table 2.

ARCH LM-test	
Null Hypothesis	no ARCH effects
Data	trainingdf$log_NSEI
Chi-squared	573,03
p-value	2,20E-16

Table:2 ARCH Test Output

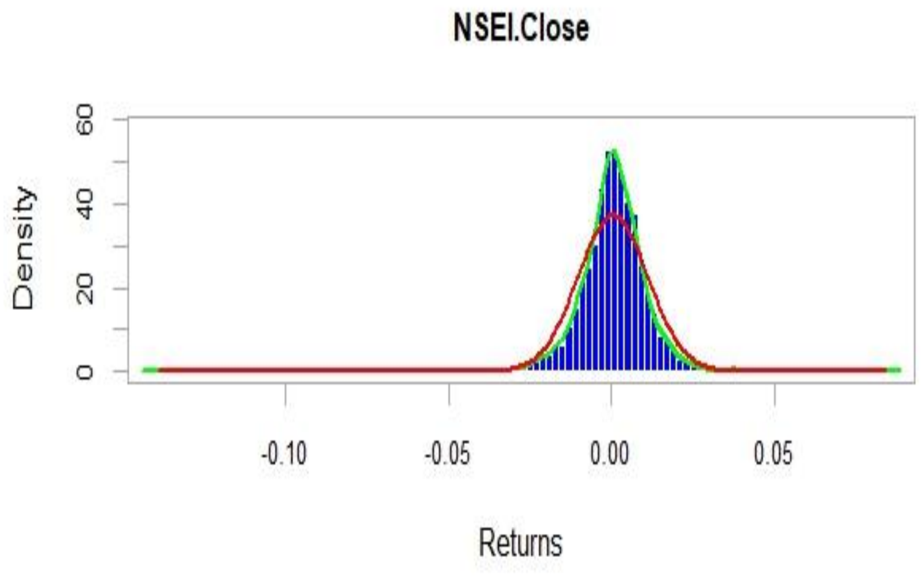


Figure: 7. NIFTY50 Histogram

Figure 7 shows the histogram of NIFTY50 returns represented in green line, the red line represents the normal distribution. It is clear from the figure that the curve for returns is taller and the tails are thicker.

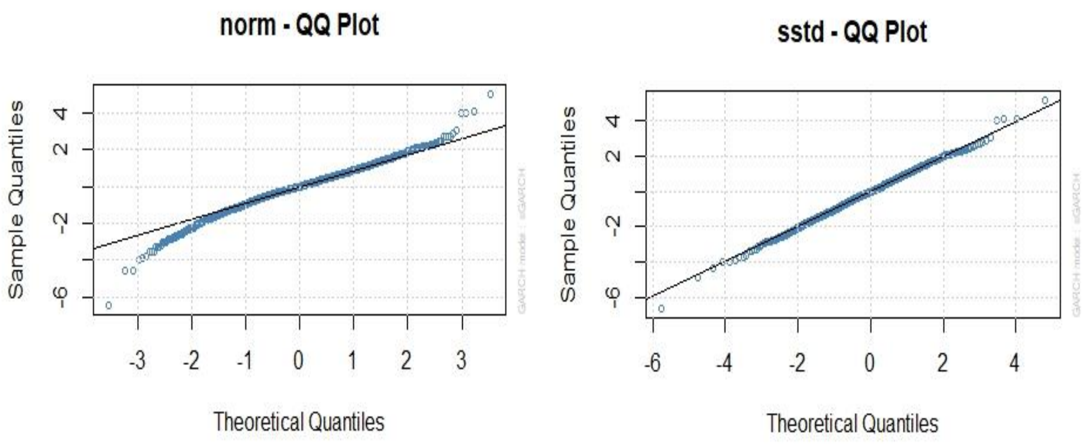


Figure: 8. QQ- plot, Normal distribution and skewed student t distribution for the residuals

Figure8. shows a Quantile-Quantile plot, a graphical representation for comparing probability distributions by plotting the quantiles of NIFTY50 standardized residuals. It can be inferred that the tails align with the straight line in a better way for student t-distribution.

3.1.2 GARCH Models and forecast method

GARCH volatility forecasting models taken into consideration in this thesis are SGARCH(1,1) with ARMA(0,0) order and EGARCH(1,1) with ARMA(1,1) order based on AIC. GJR GRACH model is not fit since the coefficients are not significant, appendix "A1. Coefficients of SGARCH and EGARCH is provided in table 3.

GARCH model	SGARCH(1,1)	EGARCH(1,1)
Mean Model	ARFIMA(0,0)	ARFIMA(1,1)
mu	0.000833	0.000408
ar1		-0.176773
ma1		0.270006
omega	0.000003	-0,282163
alpha	0.096792	-0,126237
beta	0.877358	0.969585

Table 3. Coefficient of GARCH Models

In order to forecast the volatility in the testing data, a one step ahead recursive forecast method is used. The models are estimated using an initial sample of data from t = 1,….,T, and h-step out-of-sample forecasts are created from period T onwards. The models are then re-estimated, the sample is expanded by one, and h-step ahead projections are generated beginning at T+1. This method is repeated until no longer possible to compute h-step ahead estimates (Alexander, 2008).

Bagging, Random Forest and Boosting

The R package used for running Bagging and Random Forest Algorithm is 'RandomForest'. And the package used for Boosting is 'gbm'. The data used in this thesis is regular stock data from Yahoo finance with details open, high, low, volume, close and adjusted values. These data were used to train and predict the volatility of NIFTY50 index stock price. Data pre-processing is an important step in machine learning to make sure right data is used for the machine learning model. For the same the column ‘Volume’ has been removed from the stock data as it contains ‘NAs’. The data is then divided into training and testing data set. Training set consist of 1959 observations and testing set consists of 21 observations.

Logarithm difference of the Closing price was calculated and added to the table for predicting the volatility. The dependent variable is then predicted using the remaining independent variables open, high, low, close and adjusted values. Training data set is used to train the Machine language algorithm, and the predicted value using Random Forest, Bagging and Boosting algorithm is compared with actual value for evaluating the model performance.

3.2 Model Evaluation

In this thesis, in order to measure the error to evaluate the forecasting performance, which is the difference between actual and predicted volatility of the models statistical loss functions Mean squared error (MSE) and Root Mean Squared Error (RMSE) is used. Mean Squared error is a commonly used statistical tool to compare the model’s forecasting performance. MSE formula is written as,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} \quad (y_i - \hat{f}(x_i)\,)\text{^}2 \qquad (12)$$

$\hat{f}(x_i)$ is the prediction that $\hat{f}$ gives for the *i*th observation (James et. Al, 2021).

Another statistical function that is very commonly used for measuring the model performance is root mean squared error. RMSE is the square of the mean of all of the error terms, formula is written as (Chai & Draxler,2014),

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} e_i^2} \tag{13}$$

3.2.1 Result Estimation and Discussion

Model	MSE	RMSE
SGARCH (1,1)	0.0001885146	0.01373006
EGARCH (1,1)	0.0001897961	0.01377665
Bagging	6.570256e-05	0.008105712
Random Forest	7.842966e-05	0.00856052
Gradient Boosting	5.601939e-05	0.00748461

Table: 5 Evaluation of forecast NIFTY50 INDEX

The key area where this thesis is focused on is the evaluation of the forecasting performance of testing dataset. The testing dataset for all the models consists of 21 observations. The forecasting performance is evaluated by MSE and RMSE. The model with least loss value indicates the model that has closest forecasted volatility to the actual volatility. The results of the model performance are shown in table 5 where the smallest loss value is highlighted in blue and the highest in red.

When analysing the GARCH models it can be seen that Standard GARCH model performed moderately better than asymmetric EGARCH model. This could be because the forecast is calculated for normal period (2022-01-01 to 2022-02-02) and asymmetric GARCH models are expected to achieve better predictions during crisis period. However, the difference in predictability of the models is low.

Analysing the results, it can be seen that decision tree models in general have better prediction accuracy than GARCH models. Gradient Boosting out performed Random Forest and Bagging algorithms within the decision tree model and is the best model among the four models tested in this thesis.

3.2.3 Conclusion

This thesis evaluated the volatility forecasting performance of Standard GARCH, EGARCH and GJR GARCH models and decision tree models Bagging, Random Forest and Gradient Boosting on the NIFTY 50 Stock Index. The main aim is to find the model that predicts the volatility closest to the actual volatility during the one-month period from 2022-01-01 to 2022-02-02.

The results of this paper shows that decision tree models performs better than GARCH models. Out of the decision tree models Gradient Boosting produced the least Mean squared error and Root mean squared error to become the best model among all the models tested. This is because Gradient Boosting approach utilizes sequential training by constructing an ensemble of shallow and weak consecutive trees, each of which learns from and improves upon itself.

3.2.4 Suggestions for future research

In this thesis volatility forecasting performance of only three decision tree models were evaluated. And the forecasting period consisted of only one month which was normal period. Due to this one could further extend the study by looking into the performance of other asymmetric GARCH models and advanced decision tree models during crisis period. Furthermore, the loss functions used to evaluate the performance of the models in this thesis were MSE and RMSE. Additional loss functions could be used for better assessment of the model’s prediction accuracy since the evaluation measures penalize differently with respect to the underlying loss function.

References

1. K. M. D. P. Kande Arachchi. (2018). Comparison of Symmetric and Asymmetric GARCH Models: Application of Exchange Rate Volatility. *American Journal of Mathematics,Statistics*,151–159. http://www.sapub.org/global/showpaperpdf.aspx?doi=10.5923/j.ajms.20180805.08

2. Agung, I. G. N. (2008). Time Series Data Analysis Using EViews. *Time Series Data Analysis Using EViews*. https://doi.org/10.1002/9780470823699

3. Autoregressive Conditional Heteroskedasticity (ARCH). (2021, November 1). *Investopedia*. https://www.investopedia.com/terms/a/autoregressive-conditional-heteroskedasticity.asp

4. Awad, M., & Khanna, R. (2015). *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers* (1st ed.). Apress.

5. Bhowmic, D. (2013). STOCK MARKET VOLATILITY: AN EVALUATION. *STOCK MARKET VOLATILITY: AN EVALUATION*, ISSN 2250-3153. http://www.ijsrp.org/research-paper-1013/ijsrp-p2212.pdf

6. Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140. https://doi.org/10.1007/bf00058655

7. Brooks, C. (2014). *Introductory Econometrics for Finance* (3rd ed.). Cambridge University Press.

8. Carol Alexander. (2008). *Market risk analysis II: practical financial econometrics. John Wiley & Sons*. http://srodev.sussex.ac.uk/id/eprint/40643/

9. Dixon, M. F., Halperin, I., & Bilokon, P. (2020). *Machine Learning in Finance: From Theory to Practice* (1st ed. 2020). Springer.

10. Engle, R. (2001). GARCH 101: The Use of ARCH/GARCH Models in Applied Econometrics. *Journal of Economic Perspectives*, *15*(4), 157–168. https://doi.org/10.1257/jep.15.4.157

11. Engle, R., & Patton, A. (2001). What good is a volatility model? *Quantitative Finance*, *1*(2), 237–245. https://doi.org/10.1088/1469-7688/1/2/305

12. Fabozzi, F. J., Focardi, S. M., Rachev, S. T., Arshanapalli, B. G., & Hoechstoetter, M. (2014). *The Basics of Financial Econometrics: Tools, Concepts, and Asset Management Applications (Frank J. Fabozzi Series)* (1st ed.). Wiley.

13. Guidolin, M., & Pedio, M. (2018). *Essentials of Time Series for Financial Applications* (1st ed.). Academic Press.

14. Gupta, P. (2018, June 21). Cross-Validation in Machine Learning - Towards Data Science. *Medium*. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

15. Israel, R. (2020, January 10). *Can Machines "Learn" Finance?* https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3624052

16. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R (Springer Texts in Statistics)* (2nd ed. 2021). Springer.

17. Krispin, R. (2019). *Hands-On Time Series Analysis with R: Perform time series analysis and forecasting using R*. Packt Publishing.

18. Lantz, B. (2013a). *Machine Learning with R*. Packt Publishing.

19. Lantz, B. (2013b). *Machine Learning with R*. Packt Publishing.

20. Liu, L., & Özsu, M. T. (2009a). *Encyclopedia of Database Systems*. Springer Publishing.

21. Liu, L., & Özsu, M. T. (2009b). *Encyclopedia of Database Systems*. Springer Publishing.

22. *Machine Learning by Tom M. Mitchell (1997-03-01)*. (2022a). McGraw-Hill Education; 1 edition (1997-03-01).

23. *Machine Learning by Tom M. Mitchell (1997-03-01)*. (2022b). McGraw-Hill Education; 1 edition (1997-03-01).

24. Poon, S. H., & Granger, C. W. (2002). Forecasting Volatility in Financial Markets: A Review (revised edition). *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.331800

25. Refaeilzadeh, P., Tang, L., & Liu, H. (2009). Cross-Validation. *SpringerLink*. https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_565?error=cookies_not_supported&code=923cef06-c922-4a4a-a3b2-af0de1e86b87

26. Rossi, E., & de Magistris, P. S. (2013). Long Memory in Integrated and Realized Variance. *Advances in Theoretical and Applied Statistics*, 523–532. https://doi.org/10.1007/978-3-642-35588-2_47

27. Sakhare, N. N., & Imambi, S. S. (2019). Performance Analysis of Regression Based Machine Learning Techniques for Prediction of Stock Market Movement. *Performance Analysis of Regression Based Machine Learning Techniques for Prediction of Stock Market Movement*, *Volume-7*(Is6S).

28. SCHWERT, G. W. (1990). Stock Returns and Real Activity: A Century of Evidence. *The Journal of Finance*, *45*(4), 1237–1257. https://doi.org/10.1111/j.1540-6261.1990.tb02434.x

29. Ser-Huang Poon, & Clive W. J. Granger. (2003). Forecasting Volatility in Financial Markets: A Review. *Journal of Economic Literature*. https://doi.org/10.1257/002205103765762743

30. Sheppard, C. (2017). *Tree-based Machine Learning Algorithms: Decision Trees, Random Forests, and Boosting*.

31. Srivastav, A. K. (2022, September 13). Realized Volatility. *WallStreetMojo*. https://www.wallstreetmojo.com/realized-volatility/

32. Tsay, R. S. (2013). Multivariate Time Series Analysis: With R and Financial Applications. *Multivariate Time Series Analysis: With R and Financial Applications*. http://ci.nii.ac.jp/ncid/BB14647521

33. Wang, R. (2012). AdaBoost for Feature Selection, Classification and Its Relation with SVM, A Review. *Physics Procedia*, *25*, 800–807. https://doi.org/10.1016/j.phpro.2012.03.160

34. Wang, Y., Xiang, Y., Lei, X., & Zhou, Y. (2021). Volatility analysis based on GARCH-type models: Evidence from the Chinese stock market. *Economic Research-Ekonomska Istraživanja*, *35*(1), 2530–2554. https://doi.org/10.1080/1331677x.2021.1967771

35. What is Logistic regression? | IBM. (n.d.). *Ibm*. Retrieved October 13, 2022, from https://www.ibm.com/topics/logistic-regression

36. Yan Yan Song, & Ying Lu. (2015). Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, *27*(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044

37. Zhang, Y., & Haghani, A. (2015). A gradient boosting method to improve travel time prediction. *Transportation Research Part C: Emerging Technologies*, *58*, 308–324. https://doi.org/10.1016/j.trc.2015.02.019

Appendix

A1

Model	Estimate	P- Value	Information Criteria	
SGARCH(1,1) – ARMA (0,0)				
mu	0.000833	0.000006	Akaike	-6.5759
Omega	0.000003	0.048709	Bayes	-6.5645
Alpha1	0.096792	0.000000	Shibata	-6.5759
Beta1	0.877358	0.000000	Hannan-Quinn	-6.5717

A1: Table. 1 SGARCH(1,1) – ARMA (0,0

Model	Estimate	P- Value	Information Criteria	
sGARCH(1,1) – ARMA (1,1)				
mu	0.000839	0.000013	Akaike	-6.5788
Omega	0.000003	0.047089	Bayes	-6.5617
Alpha1	0.096919	0.000000	Shibata	-6.5788
Beta1	0.878037	0.000000	Hannan-Quinn	-6.5725
ar1	-0.359700	0.356024		
ma1	0.430699	0.253723		

A1: Table. 2 SGARCH(1,1) – ARMA (1,1)

***Not statistically significant, ar$_1$, ma$_1$ P-value > 0.05**

Model	Estimate	P- Value	Information Criteria	
EGARCH(1,1) – ARMA (1,1)				
mu	0.000408	0.028313	Akaike	-6.6248
Omega	-0.282163	0.000000	Bayes	-6.6048
Alpha1	-0.126237	0.000000	Shibata	-6.6248
Beta1	0.969585	0.000000	Hannan-Quinn	-6.6175
Gamma1	0.139850	0.000000		
ar1	-0.176773	0.000123		
ma1	0.270006	0.000000		

A1: Table. 3 EGARCH(1,1) – ARMA (1,1)

Model	**Estimate**	**P- Value**	**Information Criteria**	
EGARCH(1,1) – ARMA (0,0)				
mu	0.000466	0.00618	Akaike	-6.6187
Omega	-0.277372	0.00000	Bayes	-6.6045
Alpha1	-0.115652	0.000000	Shibata	-6.6188
Beta1	0.970092	0.000000	Hannan-Quinn	-6.6135
Gamma1	0.139200	0.000000		
ar1				
ma1				

A1: Table. 4 EGARCH(1,1) – ARMA (0,0)

***Not selected as AIC value is higher than ARMA (1,1) EGARCH model**

Model	**Estimate**	**P- Value**	**Information Criteria**	
gjrGARCH (1,1)-ARMA(0,0)				
mu	0.000532	0.002672	Akaike	-6.6147
Omega	0.000003	0.000000	Bayes	-6.6005
Alpha1	0.000001	0.999857	Shibata	-6.6148
Beta1	0.880222	0.000000	Hannan-Quinn	-6.6095
Gamma1	0.163478	0.000000		

A1: Table. 4 gjrGARCH (1,1)-ARMA(0,0)

****Not statistically significant, Alpha P-value > 0.05**

Model	**Estimate**	**P- Value**	**Information Criteria**	
gjrGARCH (1,1)-– ARMA (1,1)				
mu	0.000472	0.011971	Akaike	-6.6215
Omega	0.000003	0.000000	Bayes	-6.6015
Alpha1	0.000000	0.999950	Shibata	-6.6215
Beta1	0.876182	0.000000	Hannan-Quinn	-6.6141
ar1	-0.324006	0.194445		
ma1	0.419565	0.079679		
Gamma1	0.173875			

A1: Table. 5 gjrGARCH (1,1) – ARMA (1,1)

****Not statistically significant, Alpha, ar$_1$, ma$_1$ P-value > 0.05**

######################Ajith Rajan_Master Thesis#######################

R packages used - e1071, FinTs, quantmode, random Forest, rpart, rpart.plot, rugarch, tseries, TS Studio, Xts, zoo, Metrics, gbm

```
#### NSEI SGARCH ##########

getSymbols("^NSEI", from = "2014-01-01", to = "2022-02-02")

# Remove NA

NSEI <- NSEI[complete.cases(NSEI), ]
summary(NSEI) ##confirm that there are no NA's in the dataset


# Check stationarity of the closing price

  adf.test(NSEI$NSEI.Close) #Data is non stationary


#log_NSEI (Take log difference)

log_NSEI <- diff(log(NSEI$NSEI.Close))

NSEI$log_NSEI <- log_NSEI

NSEI <- NSEI[-1]

log_NSEI<-log_NSEI[-1]


# Check stationarity- log_NSEI

  adf.test(NSEI$log_NSEI)


#plot chart (To check volatility clustering in log_NSEI data)

plot(NSEI$NSEI.Close)

plot(NSEI$log_NSEI)

chart.Histogram(log_NSEI,methods = c('add.density', 'add.normal'),

            colorset = c('blue','green','red'))
```

```
#Convert dataset to time series dataset to split the data

NSEI_ts <- ts(NSEI,start = c(2014,01,03), frequency = 252)



#plot time series

plot.ts(NSEI_ts)


#Data frame creation

NSEI_df <- data.frame(NSEI_ts)


#Descriptive statsitics

summary(NSEI_ts)


#Split data

split_NSEI <- ts_split(NSEI_ts,sample.out = 21)

training <- split_NSEI$train

testing <- split_NSEI$test

nrow(testing)

nrow(training)


##converting Test and training sets into data frame

testing_df <- data.frame(testing)

training_df<- data.frame(training)


#remove volume column(not required for random forest)

testing_df$NSEI.Volume <- NULL

training_df$NSEI.Volume<- NULL
```

```
#Diagnostic test


ArchTest(training_df$log_NSEI)

#ARCH effect is confirmed as p-value is less than 0.05

pp.test(training_df$log_NSEI)  #pp test rejects the null hypothesis

#GARCH Models

garch(training_df$log_NSEI,grad="numerical",trace=FALSE)


#GARCH(1,1) model selected


#GARCH model with ARMA(0,0) order

X1 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),

                distribution.model='norm',mean.model = list(armaOrder=c(0,0)))

XGARCH1 <- ugarchfit(spec = X1, data = training_df$log_NSEI )

XGARCH1

#All the parameters are significant (P-value < 0.05)

# plot chart

plot(XGARCH1)


#ARMA(1,1)

X2 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),

mean.model = list(armaOrder=c(1,1)))

XGARCH2 <- ugarchfit(spec = X2, data = training_df$log_NSEI )

XGARCH2

### ar1 and ar2 not significant ###


#ARMA(2,2)

X3 <- ugarchspec(variance.model = list(model="sGARCH", garchOrder = c(1,1)),

               mean.model = list(armaOrder=c(2,2)))
```

```
XGARCH3 <- ugarchfit(spec = X3, data = training_df$log_NSEI )

XGARCH3

###Not significant model##



#forecast (GARCH(1,1)- ARMA(0,0))

SGARCH_NSEI_forecast  <- ugarchforecast(XGARCH1,n.ahead = 21)


#Creating data frame of forecasted data

fitted(SGARCH_NSEI_forecast)

forecast<-sigma(SGARCH_NSEI_forecast)

sGARCH_forecast_df<- data.frame(forecast)


#MSE calculation

sGARCH_forecast_df$testing_data <- testing_df$log_NSEI

sGARCH_forecast_df$predicted <- sGARCH_forecast_df$X1975.05.14.05.30.00

sGARCH_forecast_df$X1975.05.14.05.30.00  <-  NULL

sGARCH_RMSE <- rmse(sGARCH_forecast_df$testing_data,SGARCH_forecast_df$predicted)

sGARCH_MSE <- mse(sGARCH_forecast_df$testing_data,SGARCH_forecast_df$predicted)



#### NSEI EGARCH ##########


getSymbols("^NSEI", from = "2014-01-01", to = "2022-02-02")


# Remove NA

NSEI <- NSEI[complete.cases(NSEI), ]

summary(NSEI)
```

```
#log_NSEI

log_NSEI <- diff(log(NSEI$NSEI.Close))

NSEI$log_NSEI <- log_NSEI

NSEI <- NSEI[-1]

log_NSEI<-log_NSEI[-1]


#plot chart

plot(NSEI$NSEI.Close)

plot(NSEI$log_NSEI)

chart.Histogram(log_NSEI,methods = c('add.density', 'add.normal'),

        colorset = c('blue','green','red'))


#Time series variable

NSEI_ts <- ts(NSEI,start = c(2014,01,03), frequency = 252)


#plot time series

plot.ts(NSEI_ts)


#Data frame creation

NSEI_df <- data.frame(NSEI_ts)


#Descriptive statsitics

summary(NSEI_ts)


#Split data

split_NSEI <- ts_split(NSEI_ts,sample.out = 21)

training <- split_NSEI$traintesting <- split_NSEI$test

nrow(testing)

nrow(training)
```

```
##converting Test and training sets into data frame

testing_df <- data.frame(testing)

training_df<- data.frame(training)


#remove volume column

testing_df$NSEI.Volume <- NULL

training_df$NSEI.Volume<- NULL


##EGARCH - ARMA(0,0)

Y1 <- ugarchspec(variance.model = list(model="eGARCH", garchOrder = c(1,1)),

          mean.model = list(armaOrder=c(0,0)))

Y_EGARCH1 <- ugarchfit(spec = Y1, data =  training_df$log_NSEI )

Y_EGARCH1

#All parameters are significant


##EGARCH - ARMA(1,1)

Y2 <- ugarchspec(variance.model = list(model="eGARCH", garchOrder = c(1,1)),

          mean.model = list(armaOrder=c(1,1)))


Y_EGARCH2 <- ugarchfit(spec = Y2, data =  training_df$log_NSEI )

Y_EGARCH2

#All parameters are significant, based on AIC ARMA(1,1) is selected


##EGARCH - ARMA(2,2)

Y3 <- ugarchspec(variance.model = list(model="eGARCH", garchOrder = c(1,1)),

          mean.model = list(armaOrder=c(2,2)))

Y_EGARCH3 <- ugarchfit(spec = Y3, data =  training_df$log_NSEI )

Y_EGARCH3
```

```
# Parameters not significant


###forecast###

EGARCH_NSEI_For <- ugarchforecast(Y_EGARCH2, n.ahead = 21)


fitted(EGARCH_NSEI_For)

forecast<-sigma(EGARCH_NSEI_For)


#forecasted Values data frame

EGARCH_forecast_df<- data.frame(forecast)

EGARCH_forecast_df$Predicted <- EGARCH_forecast_df$X1975.05.14.05.30.00

EGARCH_forecast_df$X1975.05.14.05.30.00<-NULL

EGARCH_forecast_df$testing_data <- testing_df$log_NSEI


#MSE

EGARCH_RMSE <- rmse(EGARCH_forecast_df$testing_data,EGARCH_forecast_df$Predicted)

EGARCH_MSE <- mse(EGARCH_forecast_df$testing_data,EGARCH_forecast_df$Predicted)




####NSEI GJR GARCH####


#ARCH(0,0)

Z1 <- ugarchspec(variance.model = list(model="gjrGARCH", garchOrder = c(1,1)),

            mean.model = list(armaOrder=c(0,0)))

Z_GJRGARCH1 <- ugarchfit(spec = Z1, data = training_df$log_NSEI )

Z_GJRGARCH1


#Not significant model as Parameter alpha1 (p > 0.05)
```

```
#ARCH(1,1)


Z2 <- ugarchspec(variance.model = list(model="gjrGARCH", garchOrder = c(1,1)),

            mean.model = list(armaOrder=c(1,1)))


Z_GJRGARCH2 <- ugarchfit(spec = Z2, data =  training_df$log_NSEI )

Z_GJRGARCH2

#Not significant model as Parameter alpha1,ar1,ma1 (p > 0.05)


#ARCH(2,2)

Z3 <- ugarchspec(variance.model = list(model="gjrGARCH", garchOrder = c(1,1)),

            mean.model = list(armaOrder=c(2,2)))

Z_GJRGARCH3 <- ugarchfit(spec = Z3, data =  training_df$log_NSEI )

Z_GJRGARCH3


#Not significant models


#forecast

NSEI_For <- ugarchforecast(Z_GJRGARCH1, n.ahead = 20)


#plot

plot(Z_GJRGARCH1)


#Histogram

chart.Histogram(NSEI_insample,

          methods = c('add.density','add.normal'),

          colorset = c('blue','red','green'))


######gjr Model not used in thesis as P-values of parameters were not significant#####
```

```
# Decision Tree Models

getSymbols("^NSEI", from = "2014-01-01", to = "2022-02-02")

# Remove NA's
NSEI <- NSEI[complete.cases(NSEI), ]
summary(NSEI)

#log_NSEI

log_NSEI <- diff(log(NSEI$NSEI.Close))
NSEI$log_NSEI <- log_NSEI
NSEI <- NSEI[-1]

#plot chart

chartSeries(NSEI$log_NSEI)

#Time series variable

NSEI_ts <- ts(NSEI,start = c(2014,01,03), frequency = 252)

#plot time series
plot.ts(NSEI_ts)

#Data frame creation
NSEI_df <- data.frame(NSEI_ts)
```

```
#Descriptive statsitics

summary(NSEI_ts)


#Split data

split_NSEI <- ts_split(NSEI_ts,sample.out = 21)

training <- split_NSEI$train

testing <- split_NSEI$test

nrow(testing)

nrow(training)


##Converting Test and training sets into data frame

testing_df <- data.frame(testing)

training_df<- data.frame(training)


#remove volume column

testing_df$NSEI.Volume <- NULL

training_df$NSEI.Volume<- NULL

training_df$NSEI.Adjusted<-


#Random forest dependent variable = log_NSEI & independent variable = open,high,low,close,adjusted


 #Only Boosting, Random forest and bagging are used in thesis and not single tree & tree pruning
method------1


  #Building Regression Tree------1


regtree <- rpart(formula = log_NSEI~.,data = training_df,

            control = rpart.control(maxdepth = 4))
```

```
#plot the decision tree

rpart.plot(regtree,box.palette = "RdBu", digits = -5)


#Predict value

testing_df$predicted <- predict(regtree,testing_df,type = "vector")


##MSE

MSE2 <- mean((testing_df$predicted-testing_df$log_NSEI)^2)

?rmse


#Tree Pruning

fulltree <- rpart(formula = log_NSEI~., data = training_df,

            control = rpart.control(cp=0))


rpart.plot(fulltree,box.palette = "RdBu", digits = -3)

print(fulltree)

plot(regtree)


mincp <- regtree$cptable[which.min(regtree$cptable[,"xerror"]),"CP"]


prunedtree <- prune(fulltree,cp=mincp)

rpart.plot(prunedtree,box.palette="RdBu",digits = -3)


testing_df$fulltree <- predict(fulltree, testing_df,type="vector")

MSE2_full_tree <- mean((testing_df$fulltree - testing_df$log_NSEI)^2)

MSE2_full_tree


plot.ts(testing_df$log_NSEI)
plot.ts(testing_df$predicted)

plot.ts(testing_df$fulltree)


testing_df$pruned <- predict(prunedtree, testing_df,type="vector")

MSE2_pruned <- mean((testing_df$pruned - testing_df$log_NSEI)^2)

MSE2_pruned


########Bagging######

bagging <- randomForest(training_df$log_NSEI~., data = training_df, mtry = 5)

testing_df$bagging <- predict(bagging,testing_df)

MSE2bagging <- mean((testing_df$bagging-testing_df$log_NSEI)^2)


#Using Metrics package

MSE2bagging <- mse(testing_df$log_NSEI,testing_df$bagging)

RMSE2bagging <- rmse(testing_df$log_NSEI,testing_df$bagging)



##########Random Forest#########

randomfor <- randomForest(log_NSEI~.,data = training_df,ntree=500)


#predict output

testing_df$randomforest<- predict(randomfor,testing_df)

MSE2randomfor <- mean((testing_df$random-testing_df$log_NSEI)^2)


#Using Metrics package

RandomFor_RMSE <- rmse(testing_df$log_NSEI,testing_df$random)

RandomFor_MSE <- mse(testing_df$log_NSEI,testing_df$random)
```

```
#########Boosting########

install.packages('gbm')
Boosting <- gbm(log_NSEI~., data = training_df, distribution = "gaussian", n.trees = 1000,
          interaction.depth = 4,shrinkage = 0.2,verbose = F)
testing_df$boosting <- predict (Boosting, testing_df,n.trees = 1000)
MSE2Boosting <- mse(testing_df$log_NSEI,testing_df$boosting)
RMSE2Boosting <- rmse(testing_df$log_NSEI,testing_df$boosting)
print(MSE2Boosting)
print(RMSE2Boosting)
```