Modelling Gains and Falls in Financial Asset Returns

An Extreme Value Theory and Machine Learning Approach

Prepared by:

Olatomiwa Akinlaja

201628555

Supervisor:

Dr. M.S. Mosia

A research project

submitted in partial fulfilment of the BSc Honors in Data Science Degree

June 17, 2020

Contents

Abstract

Financial assets are non physical types of assets that only has value in terms of ownership. The financial asset in question is stock. A stock trader thrives on the volatility of stock price movements within the financial stock market. It is due to flunctuations within the market that profit can be made, however with great profit also comes great loss.

The ability to have insight is a great priviledge when it comes to stock trading, It provides the trader with foresight knowledge on what position to take within the stock market. By modelling financial asset returns through extreme value theory and machine learning, it is possible to determine the tail distributions of companies within the S&P 500 stock market index, extreme value theory is used to determine extreme events that take place within the stock market. The machine learning aspect sought to predict whether a gain or fall happened within the financial market using various predictive models.

This research project proposes a strategy which can help business organisation, investors and traders alike to optimise profit and further reduce risk assessment by modelling gains and falls in financial asset returns through extreme value theory and machine learning, the strategy was then deployed on a web application system, in order to provide the user with an interactive insighful system with which to make future business decisions.

Keywords: Financial Assets, Extreme Value Theory (EVT), Machine Learning, Artificial Neural Network (ANN)

Chapter 1

Introduction & Background

1.1 Introduction

Stocks are how corporations and investors aquire assets to grow their businesses. Investopedia (2018) defines stock a.k.a "shares" or "equity" as a type of security that signifies proportionate ownership in the issuing corporation. The proposal of the use of an intelligent trading system by Musah, Abdul-Aziz, Salah, Abdul-Rasheed (2018) plays a major role when it comes to monitoring and determining the rise and fall of stock value within the market. This helps businesses, organizations, investors and stock traders to stay ahead of the competition, when making key decisions regarding financial asset returns.

According to Musah et al. (2018) the unpredictability of the market leads decision makers to acquire systems that can help predict prices or assistance with the descision of selling and buying financial assets based on various situations and conditions, this assists with making key investment decisions.

Thus, in making these key decisions, a model that is able to accurately predict gains and falls within financial asset returns (stock data) would provide intelligence and would help generate profit for the user, whether a business, organization or stock trader.

Lingxue and Nickolay (2017) concluded that a neural network forecasting model is able to outperform classical time series methods in use cases with long, interdependent time series. The fact stated by the empirical study performed by Lingxue and Nickolay

(2017), combined with the Big Data generated by the vast amounts of organizations used within this research project, therefore justifies the need for machine learning techniques. Statistical methods are also very robust hence why it is also combined with machine learning within this project to focus on anomalies and extreme events.

By using Extreme Value Theory to target extreme prices within the stock data in question, this ensures that important aspects of the data is targeted by fitting the data with the extreme value distribution that compensates for the fat tails of the respective data, a process suggested by McNeil (1999).

The Literature Review (Chapter 2) provides a review of the methods used within this research project from EVT to Machine learning. A brief motivation for the problem statement involved is also provided. The chapter also provides in depth explanation about financial assets and what it is all about, the methodologies involved when performing stock data predictions. The S&P 500 is also properly explained with an in depth description of how the organization was founded. The analysis process provides a step by step guide of how the analysis was carried out.

The data science methodology adopted within this research project and the steps involves from Business understanding to final deployment of the model on the interactive system (as seen in Chapter 3 (Methodology)). This chapter provides in depth explanations of the processes carried out in order to achieve the overall aims and objectives.

The Analysis & Results chapter (chapter 4), consists of various statistical and machine learning techniques applied to the stock data, multiple tests conducted in order to test the normality of the data. Other subsections include, feature engineering, Fitting EVT distributions and the R Dashboard subsection.

1.2 Problem Statement / Project Motivation

The problem regarding stock prices is that they are stochastic and often volatile in nature, while volatility is a necessary condition for traders to make higher returns/profit, it is also a double edged sword. This means that the likelihood of making profits is similar to making losses, which makes it difficult for a trader to decide on what position to take. What remains the biggest challenge for traders is finding strategies and ways to be able to determine when the gains and falls in financial asset returns happen.

One of the major fears for most financial analysts and risk administrators are that the events of abnormal market conditions produce massive and unforeseen stock losses that could upset and likely leads to liquidations and absolute risk. (Musah et al. (2018))

The importance of modeling gains and falls regarding stock data is to prevent unforseen losses because the stochatic nature of stock data introduces a factor of unpredictability. By analysing the trends within the stock data, forecasts and recommendations can be made.

Generally, it can be difficult to estimate these extreme events due to the scarcity of past data available. De Haan and Ferreira (2007) mentions how the scarcity of data weakens the ability to help determine these instances, it is due to this fact that estimates would be uncertain. The level of uncertainty extends not only within a certain range but beyond. These statements builds upon the aim of this research project as it also seeks to strengthen the generic stock intelligent trading system by including additional layers of pre-processing, feature engineering and predictive modeling.

This introduces challenges that arise when modeling gains and falls of stock data. Feature engineering would help determine how to classify a fall and gain within stock data. Thus providing a target variable which can be fed into the predictive model. Thus, building upon the use of machine-learning techniques to predict rises and falls within financial asset returns within the stock market.

A lot of interesting work has been done in the area of applying Machine learning algorithms to analyse price patterns and predicting stock prices. Liu and Zio (2018),

Hsu and Lin (2002) and Tang, Adam and Si (2018) are some of most of the research work that employ the use of support vector machines as a means for classification Liu and Zio (2018) Hsu and Lin (2002) Tang et al. (2018). Essentially, they fail to parametrise the stock data through extreme value theory.

Furthermore, these reasearch projects also fail to provide a system which can be used by stock traders to gain insights and make key descisions from the data provided by the organizations. The system makes use of the Big Data generated by the stock companies, while also exploring the combination of extreme value theory and machine learning, in order to provide a model that can accurately predict gains and falls within financial asset returns.

1.3 Research Question

- Given the trader's challenge to determine when the gains and falls in financial asset returns happen. This research project sought to respond to the research question:
  - How can gains and falls in financial asset returns be modelled using extreme value theory and machine learning.

1.4 Research Aim

The aim of this reasearch project is to develop an interactive web application system that models gains and falls in financial asset returns using EVT and machine learning.

Furthermore to also determine if the combination of EVT, Machine Learning, probabilistic programming and statistics can be used to provide a reliable stock trading system for an end user, be it an organization or stock trader.

1.5 Research Objectives

The aim can be achieved by fufilling the following objectives:

- Model tail behaviour of returns using EVT distributions.
- Predict gains and falls in financial asset returns through the use of a machine learning model
- Deploy a machine learning model that predicts gains and falls in financial asset returns on the interactive web application system.

1.6 Why Extreme Value Theory

The distribution of financial asset returns exhibit the volatility of financial markets. stock returns are computed to show the profit/loss experienced within the stock market as a result of price change over a specific period of time.

The choice of using Extreme Value Theory(EVT) is well documented in literature (Makhwiting & Sigauke & Lesaona (2014), Gilleland & Katz (2006), Bali (2007), Markos & Alertorn (2011)). The empirical study of Makhwiting et al. (2014) show that JSE Stock Returns could best be modeled using a Weibull distribution, which is one of the EVT distributions

What makes EVT an attractive distribution is its capabilities to model the tail behaviour of the distribution of the returns. The tail is where losses and gains happen within the data distribution.

1.7 Layout of Chapters

- Introduction
- Literature Review & Review of Methods
- Methodology
- Analysis & Results
- Conclusions & Recommendations
- Appendix

Chapter 2

Literature Review / Review of Methods

2.1 Introduction

This chapter provides an overall description, explanation and review of the different components integrated within this project. Other explanations include, financial assets and stock data methodologies. The source of the data used is also described followed by the analysis process. Furthermore, an extensive explanation of extreme value theory and machine learning is provided in order to motivate why each of these components are integrated within this project.

2.2 Financial Asset?

Investopedia (2018) defines a financial asset as a liquid asset that gets its value from a contractual right or ownership claim. Investopedia (2018) provides examples of financial assets, namely: Cash, Stocks, bonds, mutual funds and bank deposits. Financial assets generally have neither physical worth or form. Its value within the marketplace in which they trade reflects factors of supply and demand as well as the risks in which they carry.

Investopedia (2018) categorizes financial assets as real, intangible or financial. **Real** assets are physical assets which attain value from substancial entities or properties, for example real estate. **Intangible** assets are valuable and non physical. such as intellectual property and patents. **Financial** assets can be seen as a combination of both real and intangible as it falls within a sweet spot right in between. It might appear to be intangible with its value only being stated on a piece of paper, such as money or a listing but the piece of paper provides a claim of ownership of an entity.

2.3 Stock Data Prediction Methodologies

Shah (2007) decribes two stock prediction methologies. **Fundamental Analysis** was performed by the fundamental analysts, this is a method that is more concerned with the company than it is with the actual stock. The decisions made by the analysts are based on the company's past performance. Shah (2007) also described **Technical Analysis**, which was performed by the technical analysts. This process involved using past pattern behaviours of stock to determine stock prices.

The application of machine learning on stock data results in a technical analysis. This determines if the predictive model can accurately learn from the underlying patterns in the stock data. It is also possible to apply machine learning in order to evaluate the performance of a company through fundamental analysis. Basically the most successful mechanized stock forecast and proposal frameworks utilize a crossbreed involving both techniques specified.

The integration of the Yahoo-Finance API and Beautiful Soup library within python for webscraping. The datasets for all companies within the S&P 500 can be aquired and saved as csv files. This provides access to a wide variety of stock data that can be used for analysis. Further manipulations are necessary in order to store the multiple files as a single dataset.

The use of the very versatile, python & R programming languages and the modules and packages they offer makes it possible to demostrate an integration of machine learning

and extreme value theory. This would help to predict gains and falls within stock data returns in order for investors or companies to gain an advantage based on the business intelligence provided.

Scipy, a statistical library provides multiple statistical distributions and tests for normality, was used to test the stock data for normality before fitting with the weibull, gumbell and frechet extreme value theory distributions. The predictive neural network was developed within the tensorflow and keras frameworks. Other major libraries that were used are matplotlib for data visualization and pandas for data manipulation. (as seen in chapter 3, section 3.4)

It is necessary to have a means to showcase the results of the whole process. The use of an interactive web application will provide the end user with a stock trading system which provides multiple interactions and operations to the user.

2.4 Extreme Value Theory

Extreme Value Theory (EVT) is a statistical theory of extreme events, which can categorized as very unlikely to occur. In terms of creating financial damage, it can be costly to handle due to its low frequency and high severity. EVT is applicable to almost all univariate external problems. It seeks to assess a random variable from a given order of sample, the probability of events that are more extreme than any observed. . EVT extremes form a very large domain of stochastic processes which follow one of the three generalised extreme value distribution types, Gumbel, frechet/pareto, or Weibull. These distribution types referred to as heavy, median and short tailed respectively Friederichs (2007). Only these three types characterise the behaviour of extremes shown in figure 2.1.

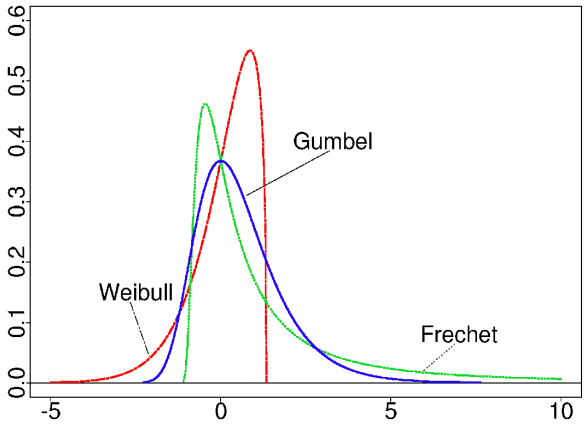


Figure 2.1: Extreme Value Theory Distributions

EVT can be applied in multiple areas such as; Finance (value at risk), re-assurance(protection against floods); Meteorology (extreme winds). (Friederichs (2007)) Other areas include heavy precipitation events, heatwaves, hurricanes, droughts, extremes in changing climate, traffic prediction, structural engineering, and risk assessment, which is the basis of this report. According to Friederichs (2007) two approaches exist for practical EVT:

- Deriving block maxima/minima series as a preliminary step
- Extracting from a continuous record the peak values reached for any period during which values exceed a certain threshold falls below a certain threshold referred to as the peak over threshold method.

EVT is used in risk management, as a method for modeling and measuring extreme risks. In order to properly model extreme stock prices the tails are of interest. The POT method provides a simple tool for estimating measures of tail risk. Another key area in predicting the expected rises and falls of stock prices is the Value-at-Risk (VaR) and a way in which useful estimates of VAR can be determined is through the embedded nature of the POT method in a stochastic volatility framework.

McNeil (1999)) stated that extreme events occur when a risk takes values from the tail of its distribution. When it comes to modeling extreme risks, which are random variables. There is a certain difficulty involved in mapping unforeseen future states into values representing profits and losses. The probability distribution of the data is

key in developing a model. Using EVT as a tool to provide the best possible estimate of the tail area of the distribution, EVT also determines the kind of distribution that is essential for dealing with extreme values within the tail. As stated by McNeil (1999), when it comes to measuring extreme risks, it is necessary to describe the tail distribution of the data.

2.4.1 EVT Models

There are two different kinds of models for extreme values. The oldest group of models are the block maxima models; these are models for the largest observations collected from large samples of identically distributed observations. A more modern group of models are the peaks-over-threshold (POT) models; these are models for all large observations which exceed a high threshold. The POT models are generally considered the most useful for practical applications, due to their more efficient use of extreme value data according to McNeil (1999).

It is possible to distinguish two classes of analysis within the POT class of models: **semi-parametric models** and the **fully parametric models** based on the Generalized Pareto distribution (GDP). The latter becoming the focus of this research paper in order to model the extreme price within stock data, because the tail of the normal distributions are too thin to address the extreme gains and losses.

In Mathematical terms, EVT is the absolute maximum and minimum value within a defined function F as seen in figure 2.2.

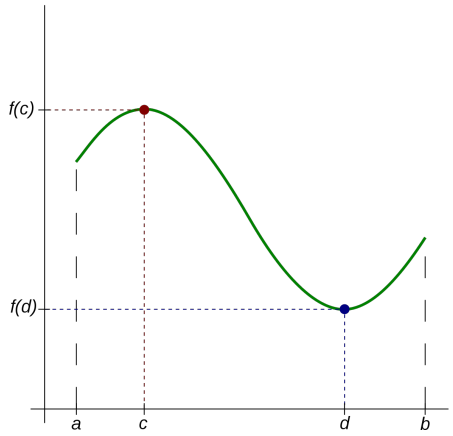


Figure 2.2: Extreme Value Theory Diagram

2.5 Machine Learning

2.5.1 Learning

Krenker et al. (2011) Mentions three major learning paradigms; supervised, unsupervised and reinforcement learning. All of which can be employed by any given type of Artificial Neural Network (ANN) architecture. figure 2.3 illustrates.

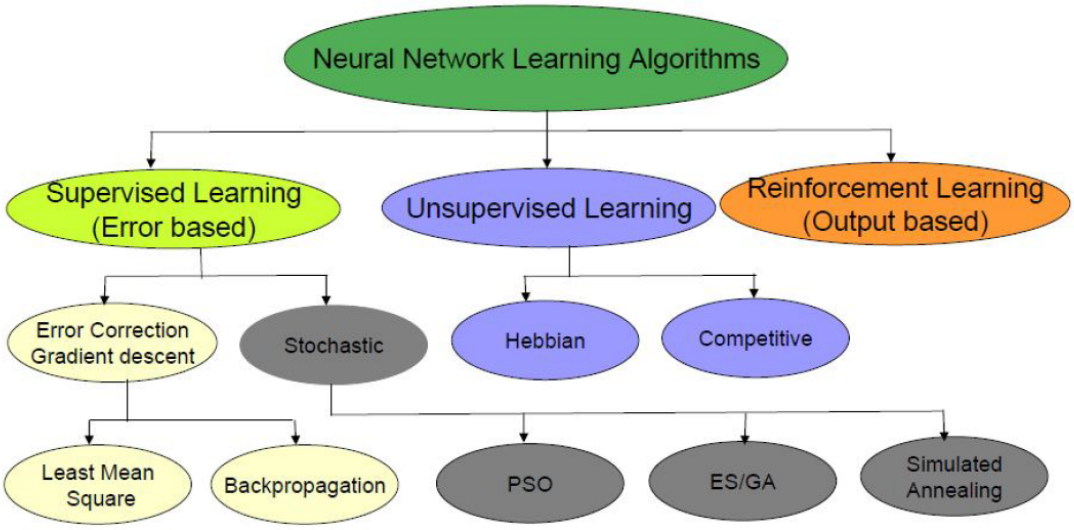


Figure 2.3: Classification of neural network algorithms

Supervised learning is a machine learning technique that sets the parameters of an ANN from training data (Krenker et al. (2011)). The training data contains input features with its corresponding target values, represented as data vectors. Another word for supervised learning is classification; there is a wide variety of classifiers like Support Vector Machines, Naïve Bayes, Decision Trees, Multi-layer Perceptron and k-nearest neighbour etc. It becomes more of an art, than a cience when it comes to choosing a classifier that is suitble to solve the probem at hand.

Unsupervised learning on the other hand is a technique used in the field of machine learning where the data provided and cost function is influenced by the defined parameters of the ANN. Determination of data organization is one of the aims of unsupervised learning. The main dofference between unsupervised and the other forms of learning is that the ANN is provided with only unlabelled attributes. Unsupervised learning is commonly used for compression, statistical modelling, clustering and filtering. (Krenker et al. (2011))

Reinforcement learning is a technique used to define ANN parameters in such a way that the algorithm is given some kind of reward based on correct predictions. (Krenker et al. (2011)) mentions that data is usually not given, but generated by interactions with the environment. The ANN seeks to maximise performance in order to attain some form of long term reward.

In the case of this research project, the proposed ANN algorithm incorporates supervised learning, by feeding the ANN with labelled stock data. The label is generated through feature engineering which determines the gains and falls within the data. The interactions recorded by the network makes it for learning to occur through backpropagation by the means of error correction and gradient descent.(as seen in figure 2.3). This process makes it possible to model rises and falls within stock data.

There has been multiple contributions by researchers who have dedicated time and resources in order to forecast flunctuations of financial asset returns. Similar efforts have also been taken to create trading strategies which are meant to provide insights which could in turn generate profit. In addition to these contributions, In the area

of financial forecasting and trading, It has been demonstrated that ANNs provide promising results. (Chen et al. (2003))

2.5.2 Artificial Neural Network

The human brain has the ability to absorb new information, and is dynamic enough to react to multiple factors. Regardless of whether the information it receives is unclear or incomplete, the brain is still capable of analysis. Thus. in an abstracted form, the ANN seeks to imitate the human brain. (Chen et al. (2003))

Krenker, Bester and Kos (2011) defines an Artificial Neural Network as a mathematical model that tries to simulate the structure and functionalities of biological neural networks. The basic building blocks of every ANN is the artificial neuron. The neuron performs three kinds of operations namely: multiplication, summation and activation.

For example, given an ANN that has an input layer, a single hidden layer and an output layer. The process involves feeding inputs into the neuron within the input layer; the next step involves multiplying each input neuron to form a weighted sum before feeding it into the middle layer, called the hidden layer. After another weighted sum, the output gets fed into an activation function a.k.a the transfer function, before generating the final output within the output layer. figure 2.4 illustrates.

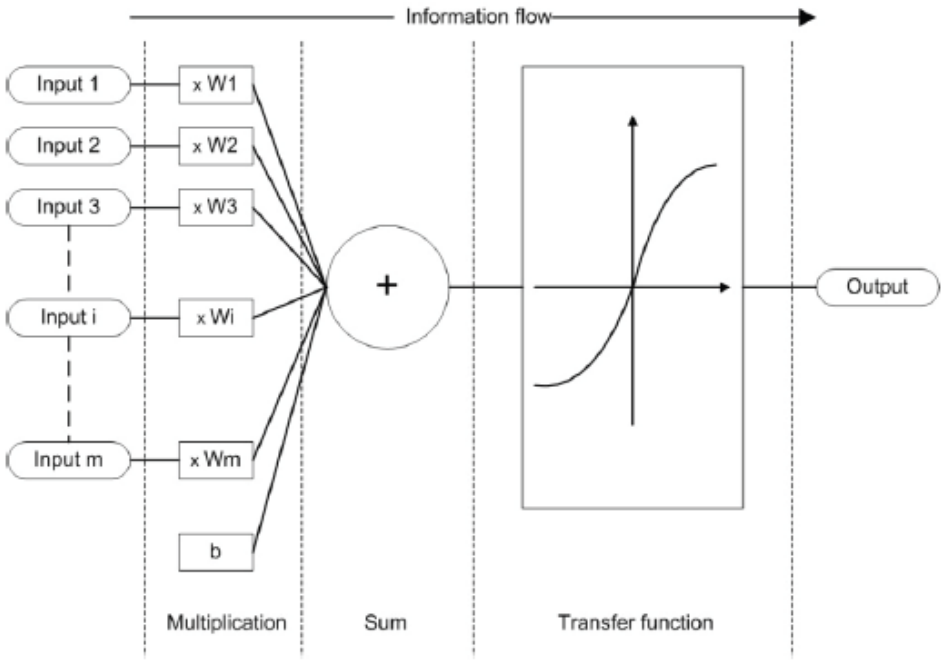


Figure 2.4: Overview of neural network

Some areas where ANNs are applied includes, game logic, predictive modelling, radar systems, fraud detection, and many other areas like chemistry, automative and space industry. They are also used for problem solving, problems which generally involve classification, clustering, function approximation, time series prediction and pattern recognition. This research project encorporates the use of ANNs for the purpose of classification and decision making.

Neuron

The basic building block of every ANN is the neuron. Kukreja, Bharath, Siddesh, and Kuldeep (2016) stated that its design and functionalities are derived from observation of a biological neuron that is a basic building block of biological neural networks, which includes the brain, spinal cord and peripheral ganglia. In figure 2.5 there is a representation of a biological neuron which has dendrite, soma and axon compared with an artificial neuron which has inputs, bias, activation function and output. (figure2.5)

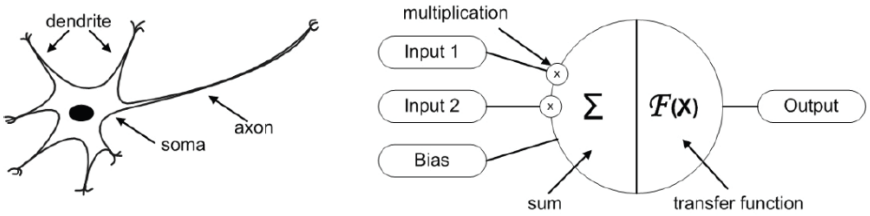


Figure 2.5: Biological and Artificial Neuron design

A biological neuron takes in information via the dendrite for the soma to process before passing it on via the axon. In the case of the artificial neuron, information is taken in from the input nodes. The input gets multiplied by the weights to form a weighted sum, which is then fed into the body of the neuron. Within the body is where the weighted sum gets fed into an activation function, the processed information is then passed to the output node.

$$y(t) = F \cdot \left(\sum_{i=0}^{j} w_i(t) \cdot u_i(t) + b \right) \tag{2.1}$$

Where:

- $u_i(n)$ represents input value in discrete time t where i goes from 0 to j,
- $w_i(t)$ represents a weight value in discrete time t where i goes from 0 to j,
- b represents the bias,
- F represents an activation function which defines the characteristics of an artificial neuron
- $y_i(t)$ represents the output value in discrete time t.

An ANN is the result of the combination of two or more neurons. A neuron is useless by itself, the tight coupling of multiple neurons working together makes them useful

in solving a problem. ANNs are not only useful for solving real-life problems they are also known for their capability of solving complex real-life problems by processing information in their neurons.

Neural Network Topology

Multiple ways in which a neural network can be interconnected are referred to as a architecture, topology or graph of an ANN. The numerous ways in which a neural network can be interconnected would result in multiple possible architectures that can be divided into the two basic classes as seen in figure 2.6

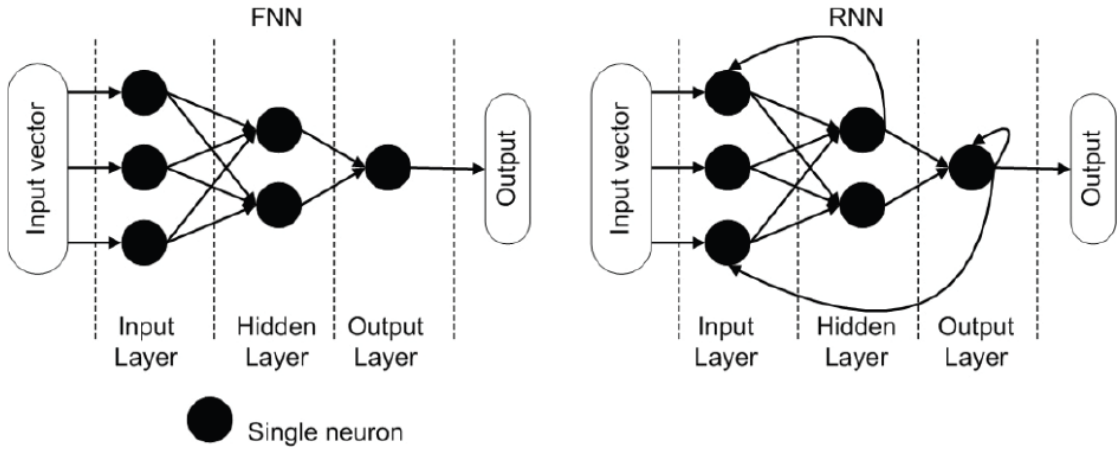


Figure 2.6: Feed-Forward (FNN) and Recurrent (RNN) graph of an ANN

In the feed-forward neural network, there are no back loops involved. There is a natural uni-directional flow of information from input layer to output layer. There is no limit to the amount of layers within the network, the number of connections and type of activation function used in each neuron is also not limited.

The recurrent neural network is not limited to the amount of back loops it can have and also the direction in which information flows, as opposed to the feed-forward network which has no back-loops and is uni-directional. This means information can be transmitted backwards, this gives it a form of memory to process any sequence of inputs. This research project sought to incorporate a simple feed forward neural network in order to predict rises and falls within financial asset values, the use of

recurrent neural networks would help create a comparable model, to see which neural network generates the most accurate prediction.

Kukreja et al. (2016) emphasized that the categories of ANN are based on supervised and unsupervised learning methods. The simplest form of ANN architecture is the perceptron, which consists of one neuron with two inputs and one output. The activation function used is called the step function or ramp function. Perceptrons are used for classification of data into two separate classes. For more complex applications, multi-layer perceptrons (MLPs) are used, which is one of the model architectures used within this project. The MLP contains one input layer, one output layer, and one or more hidden layers as given in Figure 2.4. The most common method used when training the neural network is through backpropagation. (Kukreja et al. (2016))

Training

Aydogdu (2018) provides an explanation about the main task involved when building a network. The first phase which is known as the training the network, involves coming up with a reasonable network structure followed by making a decision about the weights, biases, and activation functions. This is an iterative, potentially time consuming process. The typical strategy involves first coming up with a paramater space or a set (or a range) of reasonable values for hyperparameters. For instance, the number of hidden layers is a hyperparamater; and two hidden layers could be considered as the parameter space.

After the network has been fully defined and developed, The data is then split into training and testing sets. The network is then evaluated on test data, after learning from the training data. The model should learn enough to make accurate predictions without bias, but not to the point where it memorizes in order to give the best form of accuracy.

Backpropagation

A way in which to train the network is through backpropagation. Once the weights of the network are initaialised to random values, the weights being the arrows feeding from one neuron to the next as seen in figure 2.6. Once the error is computed, The use of partial deratives are used to determine new values for the neural network weights in order to minimize the error. The error is calulated as seen in the equation below.

$$E = \frac{1}{2} \cdot \Sigma \left(t_i - O_i\right)^2 \tag{2.2}$$

Where:

- t_i is the actual target value.
- O_i is the predicted value.
- E is the error or cost function that has to be minimised.

In order to minimise the error the following partial deivatives are necessary.

$$\frac{\partial E}{\partial w_j i} = \frac{\partial E}{\partial O_i} \times \frac{\partial O_i}{\partial a_i} \times \frac{\partial a_i}{\partial w_j i} \tag{2.3}$$

Where:

- $\frac{\partial E}{\partial w_j i}$ represents the partial derivative of the error generated (E) , with respect to the partial derivative of the weights $w_j i$ feeding into the final layer within the neural network.
- $\frac{\partial E}{\partial O_i}$ is the partial derivative of E with respect to the partial derivative of the output O_i.

- $\frac{\partial O_i}{\partial a_i}$ is the partial derivative of O_i with respect to the partial derivative of the final layer a_i.
- $\frac{\partial a_i}{\partial w_j i}$ is the partial derivative of a_i with respect to the partial derivative of the weights within the previous layer $w_j i$.

Neural Network Strengths

Kukreja et al. (2016) discussed the different artificial neural network strengths. NN outperforms other types of generic computer programs because of the following characteristics:

- Adaptive learning: ANNs is able to mimic the operation of a human brain.
- Self organization: ANNs are dynamic enough to be self organized during the process of learning, instead of being focused on a predetermined set of tasks.
- Parallel operation: NN is parallel in nature, just like the human brain as opposed to the serial execution nature of a computer program
- Fault tolerance: ANNs are capable of handling noisy and incomplete data,
- ANNs are fast and are capable of processing information quickly.
- The use of ANNs extends to areas which involve pattern recognition and data classification.

Chapter 3

Methodology

3.1 Introduction

This research project adopts the use of the methodology for data science proposed by IBM Big Data & Analytics Hub as seen in figure 3.1. Rollins (2015) elaborates that the methodology aims to provide data scientists with a framework for solving problems using data science techniques. In order words to provide a guiding strategy with stages that represent an iterative process leading from the conception of a solution to its deployment, feedback and refinement as seen in figure 3.1.

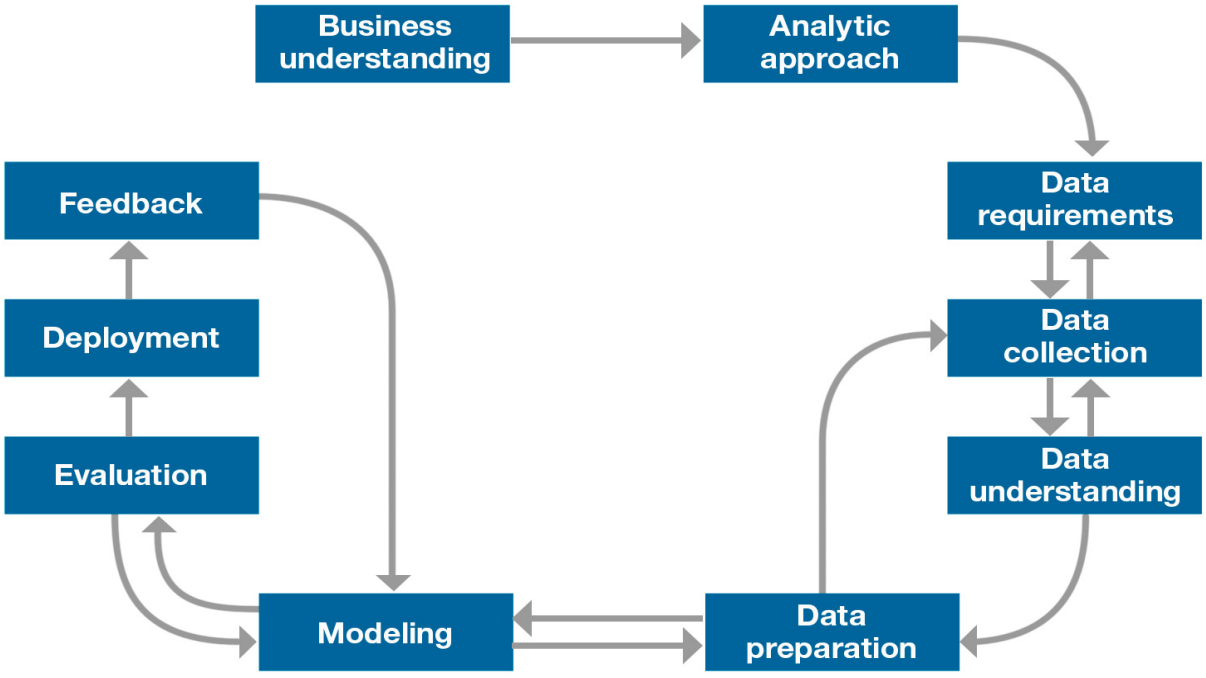


Figure 3.1: Data Science Methodology adopted by IBM Big Data & Analytics Hub

3.2 Data Science Methodology

- **Business Understanding** is the first stage within the methodological process that involves defining the problem, objectives and solution requirements.
- **Analytic Approach** follows the business problem statement, the analytic approach used to solving it, from statistical to machine learning techniques. This project used various techniques within the analytic approach.
- **Data Requirements** guide the choice of the analytical methods. In order to properly utilize the Stock data required within this project statistical tests and machine learning techniques were used.
- **Data Collection** involves gathering of data resources, whether structured, unstructures and semi-structured. structured data was used within this project.
- The **Data Understanding** stage involves in depth comprehension of the data in question.
- The **Data Preparation** stage comprises all the activities used to construct the data set that will be used in the modeling stage. These activities include data cleaning, combining data from multiple sources and transforming the data into more useful variables.
- **Modeling** involves using a training set to develop predictive or descriptive models using the analytic approaches already described. A test set is then used to forecast future targets.
- The **Evaluation** stage evaluates the model's quality and checks whether it addresses the business problem fully and appropriately.
- The **Deployment** stage involves the release of a satisfactory model into a production environment within the business sector.
- **Feedback** is provided on the model's performance, refinement is necessary in order to increase overall accuracy and model usefulness.

3.3 Business Understanding

The area of Financial assets deals with a major percentage within an organization as a whole. It is defined as a liquid asset that gets its value from a contactual right or ownnership claim. The financial asset reflects value within the marketplace in which they are traded. (As seen in chapter 2, section 2.1). As a trader, whether individual or organization, stock is the financial asset in which i am interested in. Profit or loss can be generated form the volatility of this type of financial asset within the financial market.

3.4 Data Requirements, Collection & Understanding

This reasearch project used stock data from companies listed witin the S&P 500. Standard & Poor (S & P) are the founding financial companies. It is an American stock market index based on the market capitalizations of 500 large companies. It represents the stock market's performance by reporting the risks and returns of the biggest companies.

According to Global (2019) S & P 500 was officially introduced on March 4, 1957, by Standard & Poor, It was later acquired by McGraw Hill in 1966, It is not owned by the S & P Dow Jones Indices. That is a joint venture between McGraw Hill Financila, CME Group, and News Corp. It is used as an overall market benchmark by investors and stock traders alike to compare all other investments.

Balance (2019) provides an in depth explanation of how S & P works. The founding companies track the market capitalization of the companies in its index. Market capitalization is the total value of all shares of stock a company has issued. It is calculated by multiplying the number of shares issued by stock price.

The index is weighted by a *float-adjusted market capitalization*, the goal of float-adjusted market capitalization is to distinguish between long-term strategic shareholders, whose

holdings are considered to not be available to the market, and shareholders who are considered more short-term in nature.

The selections of the companies within the S & P 500 is done by a committee based on each company's liquidity, size, and industry. All of which are based on a quaterly rebalanced index. According to Balance (2019), the criteria required in order for a company to qualify for the index is:

- The company has to be based in the United States and have a market capitalization of at least $6.1 billion.
- A minimum of fifty percent of the corporation's stock must be available to the public.
- A minimum stock price of $1 per share.
- A filed 10-k annual report(required by the U.S. Securities and Exchange Commission (SEC), that gives a comprehensive summary of a company's financial performance).
- A minimum of 50 percent of its fixed assets and revenues must be in the United States.
- A compulsory four consecutive quarters of positive earnings.

Furthermore, according to Indices (2019), S & P includes business development companies and real estate investment trusts. An example of the top 10 largest companies with a weighted market cap, in the S & P 500 in the year 2017, from first to last were Apple; Microsoft; Berkshire Hathaway B; Facebook; JP Morgan Chase; Johnson & Johnson; Exxon Mobil; Alphabet C, formerly Google; and Alphabet A.

3.4.1 Ethical Considerations

There are no ethical implications involved with the use of stock data of the companies within this project. The data is open source and free to be used by the public. Google and Yahoo! provide free access to download stock quotes using an Application Programming Interface(API). All companies have stock tickers. E.g. Apple has the stock ticker AAPL. Stock data can be scraped through the use of the provided API by specifying the ticker and tag within the beautiful-soup python module for web scraping, in order to acquire information such as trading data. (Index (2019))

Figure 3.2 is a visual representation of one of the company's stock data. The major attributes in question would be the volume generated and the Adj Close column for all companies within the S&P 500 list.

Date	High	Low	Open	Close	Volume	Adj Close
2014-09-10	19.680000	19.430000	19.650000	19.610001	4309400.0	18.881136
2014-09-11	19.540001	19.200001	19.469999	19.410000	6268000.0	18.688570
2014-09-12	19.530001	19.100000	19.530001	19.120001	6563400.0	18.409348
2014-09-15	19.209999	18.780001	19.180000	18.860001	7353800.0	18.159008
2014-09-16	19.240000	18.750000	18.809999	19.139999	5498400.0	18.428604

Figure 3.2: view of a company's stock data

Acquiring the data from Yahoo! finance is one, if not the most crucial step within this resrearch project. Overall the S&P 500 index contains 505 stocks. It is fairly easy to scrap the stock of a single company, the problem arises when scraping over 500 stocks from multiple companies while also keeping track of each company and aggregating all of them at the end.

Python offers multiple libraries to help with this tedious process by making the job of compiling and processing multiple files much easier. The libraries used within

the scraping process are: *beautifulsoup, pickle, requests, os, pandas, np, collections, pandasdatareader, datetime.*

The process starts with the *datetime* library which helps to set a date range, to determine which years to start pulling stock data from. The range chosen is from the year 2014 till the end of 2018. The *pandasdatareader* library takes as parameters, the company ticker, which is usually an abbreviation of the company name, followed by the website the data should be scrapped from and the start date which is determined by the datetime object.

The *beautifulsoup* library is used to scrap all S&P 500 company names and tickers, which is stored as a pickle file using the *pickle* library. Once the process is complete, the creation of multiple functions and loops are necessary in order to utilise the libraries in an iterative manner. Each company's stock data is then scrapped within a for loop using the pickled tickers file saved with a combination of the *pandasdatareader* library. The *os* library is used to manipulate directories within the local machine in order to save the scrapped stock data. All csvs are then read using *pandas* and stored as a single dataframe for further proceessing.

3.5 Data Preparation

3.5.1 Feature Engineering

All machine learning algorithms required some form of input data to generate some form of output. Rencberoglu (2018) explains the importance of feature engineering in regards to preparation of a proper input dataset that would be compatible with the machine learning model requirements. This would on turn lead to improvements in model performance. Shekhar (2018) describes feature engineering as art. It involves the process of using domain knowledge of data to create features which make machine learning algorithms work. Shekhar (2018) supports the statement of how feature engineering increases the predictive power of machine learning algorithms through the creation of features from raw data that help facilitate the process of machine learning.

Some techniques used for feature engineering according to Rencberoglu (2018) are:

- Scaling
- Log Transform
- Handling Outliers
- Imputation

Different techniques have various effects depending on the dataset it is used on, This research project encorporates the use of all the techniques listed above in order to create a reliable predictive model for determining gains and falls within a dataset. A brief description of each technique is provided below.

Scaling uses normalization and standardization techniques to scale multiple columns so they can be properly compared within the same range. **Log Transform** is a technique used to handle skewed data in order to approximate the distribution to normal. **Handling Outliers** through visual methods is advisable over statistical methods because statistical methods are prone to mistakes. The removal of outliers improves the overall performance of a machine learning algorithm by eliminating the biases within the dataset. **Imputation** simply involves dealing with the missing values within the dataset numerically or categorically. This techniques is a preferable option to dropping a dataset column or row because it preserves the data size. Emphasis is placed on the chosen imputed values as they have to be proper representation of the value in question.

The process of feature engineering is not limited to the techniques listed above. In order to generate a feature that can work with the model, there are brainstorming processes involved in order to create a feature that works perfectly. To illustrate further, Shekhar (2018) exclaims that feature engineering is the process of transforming raw data into features that better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

A target attribute was created in order to determine gains and falls within the stock prices of each dataset, The source code for the target generation is provided in the aapendix (chapter 5, section 6.2)

3.6 Modelling & Evaluation

The hypothesis regarding stock data states that; Stock data is normally distributed. Tests for normality are performed on the data in order to test the null hypothesis. It ended up rejecting the null hypothesis by showing that stock data is not normally distributed (as seen in chapter 4, sec 4.1). Due to the fat tails within the data, the next process involves fitting the EVT distributions with the tail of best fit by determining the goodness of fit test for each EVT distribution using the chi-square, and p-value tests.

The use of multiple machine learning algorithms are used within this research project namely: Logistic regression, Multi-layer Perceptron and a Deep Neural Network. All of which were evaluated using k-fold cross valuation (up to 10 folds) and are all compared using the accuracy metric. Logistic regression basically takes in data as input and generates an output. A multi-layer perceptron introduces a hidden layer in which interactions are recorded, while a deep neural network has multiple hidden layers within the network in order to record exponentially record interactions within features in order to generate a prediction. (results can be seen in chapter 4, section 4.3)

3.7 Deployment

The results of the model performance and visualizations are deployed on the interactive shinydashboard web application.(seen in chapter 4, section 4.4). This provides the user with access to all the companies within the S&P 500 index. The user which could be a trader may visualize how the company of their choice has perfomred over the previous years, and also show how the model perfomed based on the company chosen. They are also provided with download options for the dataset of the company of their choice, including a summary of each company's performance.

Chapter 4

Analysis & Results

4.1 Introduction

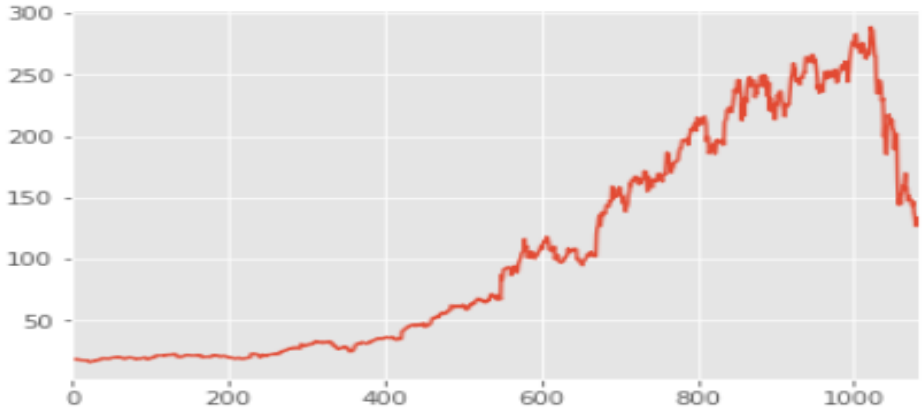
This chapter showcases the methodological process used to solve the problem statement at hand, including the various tools and techniques used to achieve the proposed aim of modeling gains and falls in financial asset returns through the integration of extreme value theory and machine learning.

This chapter presents results acquired during the analysis process, the overall challenges involved are also specified. Various techniques used during the process of testing for normality are also provided in order to test the null hypothesis which states that stock data is normally distributed, alleviating the normality assumption. After the hypothesis is tested, The data was fitted with best extreme value theory distribution. The data was then pre-processed, feature engineering was incorporated in order to determine the gains and falls. The data was then fed into the deep neural network model, which predicts the gains of falls within the data.

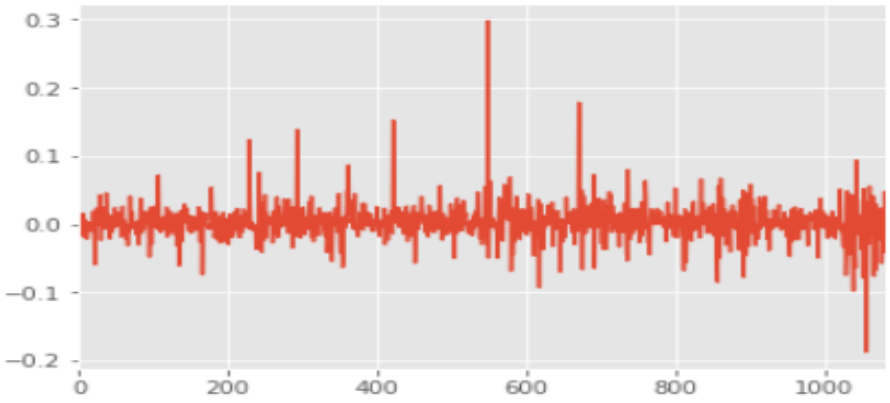
Furthermore a brief overview of the working system is presented to showcase the working model via a R Dashboard, which provides a user interface to interact with the S&P 500 data. It provides various manipulations and visualizations for the data.

In Figure 4.1 , Figure 4.1a represents the stock price data of NVIDIA, with major rises and falls within an overall period of five years from 2014 to 2018. Figure 4.1b

represents the returns which basically showcases profit/loss after a certain amount of time, usually a day.



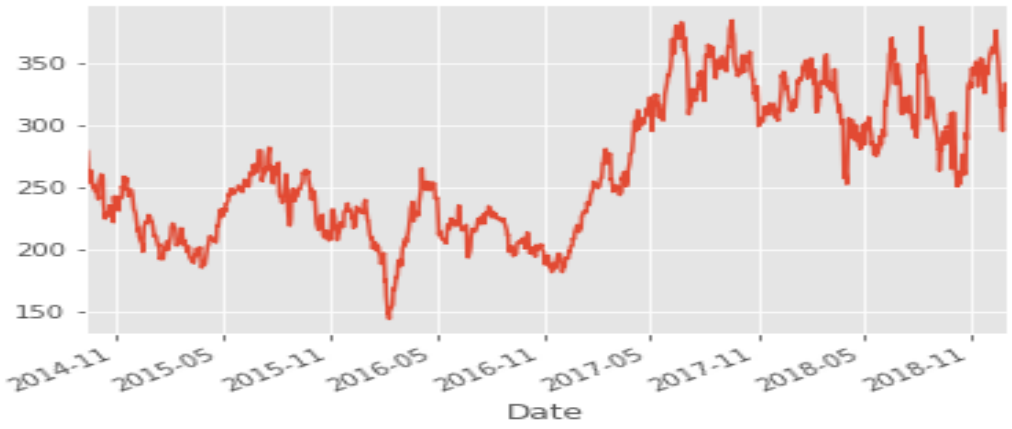
(a) NVIDIA Stock Prices



(b) NVIDIA Stock Returns

Figure 4.1: NVIDIA

Exploring Tesla's stock data; The moving average visualization in figure 4.2b shows reduced effect of temporary variations in the stock data in order to define a clearer view of the flunctuating trends within the data, by filtering out the 'noise'.



(a) TESLA Stock Price

(b) TESLA Moving Average

Figure 4.2: TESLA

By viualizing the distribution plot of the stock data (figure 4.3), it is possible to see the spread of the data. The distribution has fat tails of really high and really lows recorded values. It would be inaccurate to assume that the data is normally distributed, as emphasized by the fat tails shown by the plot. Additional Distribution plots are provided in the appendix, to show the fat tails of the stock returns of the companies. (see Appendix, chapter 6, section 6.2)

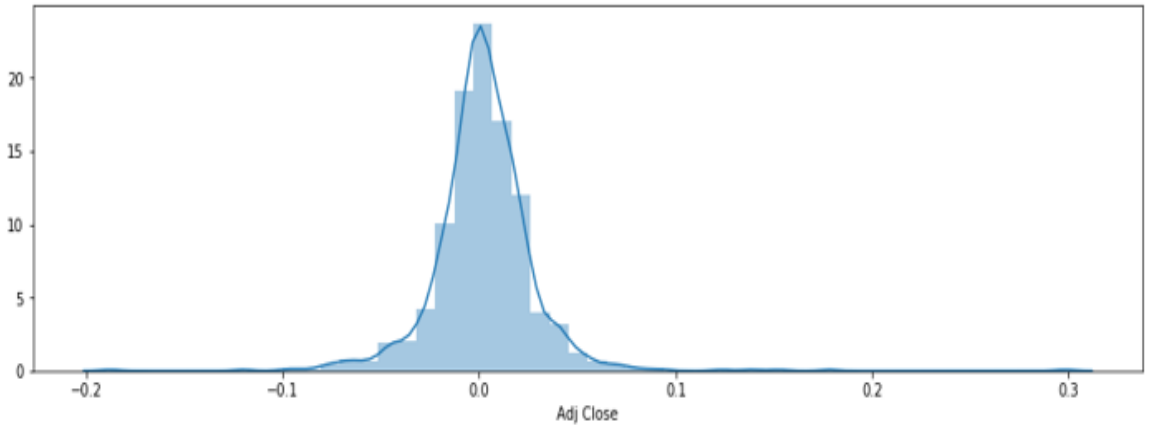
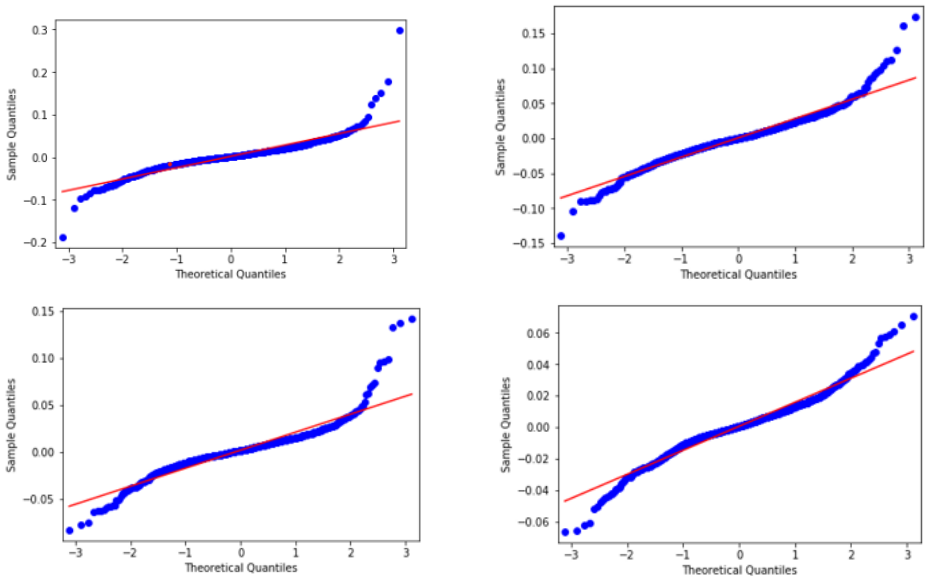


Figure 4.3: Density Plot of NVDA Stock Distribution

4.2 Test for Normality

Since the assumption of the stock data normality would be inaccurate, it would be necessary to test the hypothesis that stock data is normally distributed through statistical tests.

The quantile-quantile plot (Q-Q plot), is a plot commonly used to check the data sample distribution. Mastery (2019) motivates that the plot generates its own sample of the idealized distribution that is being compared with. Mastery (2019) elaborates that the idealized samples are divided into groups called quantiles. The sample contains data points that are paired with a comparative part from an ideal distribution. The subsequent focuses are plotted as a disperse plot with the ideal values on the x axis and the information test on the y axis.



(a) Q-Q Plots for Multiple Company

The Q-Q plots of 4 major companies in figure 4.4a shows how the stock data deviates from normality at the fat tails. An argument can be raised about disregarding the tails, due to the fact that the rest of the data is normally distributed. but the tails are the important aspects of the distribiution because that is where the gains and falls happen. It is due to this fact that the assumption of normality would be in-accurate as this would neglect the gains and falls within the fat tails.

To further substantiate the non normality of the stock data, the Anderson-Darling test was used to test the hypethesis that the data is normally distributed. Mastery (2019) defines the Anderson-Darling test as a statistical test that can be used to evaluate the normality of a data sample after returning a list of critical values rather than a single p-value. This provides the means for a more thorough interpretation of the result.

The *anderson* function within the scipy library was used to conduct the test. The test uses critical values in order to determine whether to reject the null hypothesis or not, in this case each of the critical of the critical values rejects the null hypothesis which stated that the data is normally distributed. Other tests for normality were also conduction such as the Kurtosis (6.5), Shapiro-Wilk (6.6) and D'Augostino's K^2 (6.7) tests, all of which supported the rejection of the claim that stock data is normally distributed. The outputs for these tests can be seen in the Appendix (chapter 6, section 6.2.2).

4.3 Fitting EVT distributions

Since the non normality of the data has been established. Fitting the data with the best distribution will be necessary in order to account for the fat tails of the data. The data was fitted with the following distributions:

- general pareto
- frechet(left)
- frechet(right)
- gumbel(left)
- gumbel(right)
- weibull(min)
- weibull(max)

Figure 4.5 shows the overall results of the fit, from top to bottom each distribution is ranked based on the approximate chi-squared goodness of fit. Ideally the lower the chi-squared, the better the distribution fits the data.

Distributions sorted by Goodness of fit:

	Distribution	chi_square	p_value
1	frechet_l	133.215843	0.16773
6	weibull_max	133.215843	0.16773
4	gumbel_r	179.859895	0.21053
3	gumbel_l	336.386819	0.28843
0	genpareto	977.350495	0.47868
2	frechet_r	3199.671718	0.81299
5	weibull_min	3199.671718	0.81299

Figure 4.5: Goodness of Fit

The distribution parameters of the best fits in 4.6 are fitted against the data in order to account for the fat tails within the non-normal data distribution.

	Frechet_L	Weibull_max	Gumbel_r
Location	10.201	10.201	0.037
Shape	0.319	0.319	0.037
Scale	0.328	0.328	-0.011

Figure 4.6: Distribution Parameters

4.4 Machine Learning

The use of three different algorithms as opposed to one. A **Logistic Regression Model** which can be described as a neural network without an activation function, it takes in input and generates output. A **Multi layer Perceptron (MLP)** model was used which is a neural network with a single hidden layer, it measures interactions within the hidden layer and learns through backpropagation (as seen in chapter 2, section 2.2.4). Finally a **Deep Neural Network** was used in order to determine if accuracy would be increased. It consists of multiple hidden layers, with dropout layers to avoid overfitting after each hidden layer. (as seen in Appendix, chapter 6, section 6.3)

The performance of these three algorithms were evaluated using a **k Fold Cross-Validation** which simply divides the data into k equal segments (10 in this case) in order to test overall model performance. It also helps determine if there was overfitting involved during model training. Overfitting is when the model memorises the target variable in order to provide the best predictions that acquire the highest accuracy. By using k fold cross-validation this randomises the testing of random instances of the dataset which prevents the model from memorising to provide the highest accuracy.

4.4.1 Logistic Regression Performance

The performance of the logistic regression model can be seen in figure 4.7, it performed relatively poorly as compared to the other models. A reason could be due to the fact that the interactions within the data could not be quantfied, and the model could not backpropagate in order to learn from its mistakes, it simply took in input and generated an output.

4.4.2 Multi-layer Perceptron

The MLP performed excellently and generated accurate predictions. The cross validation output in figure 4.7 shows how the data performed with 10 instances of the divided dataset.

4.4.3 Deep Neural Network Performance

The deep neural network performed excellently, it generated extremely accurate results due to the multiple hidden layers within its architecture, enabling it to records major amounts of interactions between the data. This lead to impressive results as seen in 4.7 below

The Accuracies for each model can be seen in the table below:

K Fold	Linear Regression (%)	Multi-Layer Perceptron (%)	Deep Neural Network (%)
1	43.42	97.92	99.54
2	50.63	98.15	100
3	44.47	98.15	100
4	49.32	98.84	99.54
5	47.01	97.80	99.07
6	44.06	97.80	100
7	48.77	98.38	98.61
8	47.89	98.26	99.54
9	44.88	97.45	99.53
10	48.92	99.07	99.07
Overall Model Acc (%)	**46.93**	**90.12**	**99.07**

Figure 4.7: Overall Model Accuracies

4.5 R Dashboard (shiny & shinydashboard)

The web application was designed using the shinydashboard package/library that was developed by R Core Team (2017).

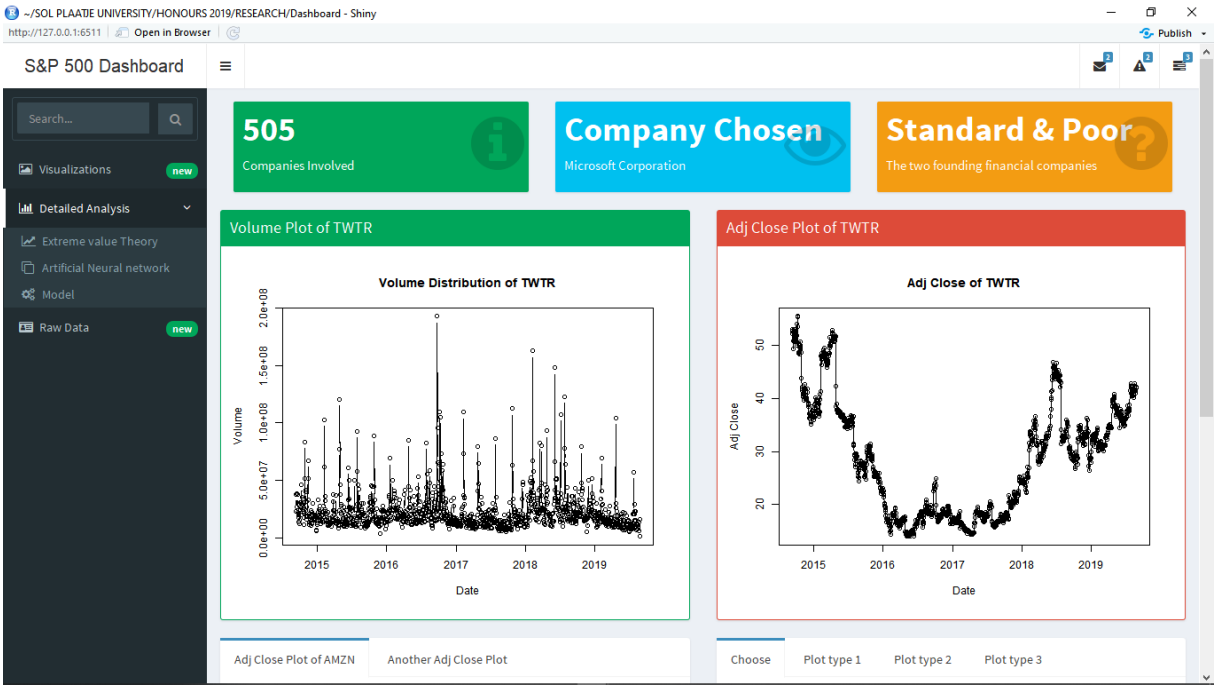


Figure 4.8: Dashboard Overview

Figure 4.8 shows the overview of the dashboard, the sidebar provides access to various options for the sake of exploring the companies within the S&P 500. The dashboard changes dynamically and creates plots based on the user's choice.

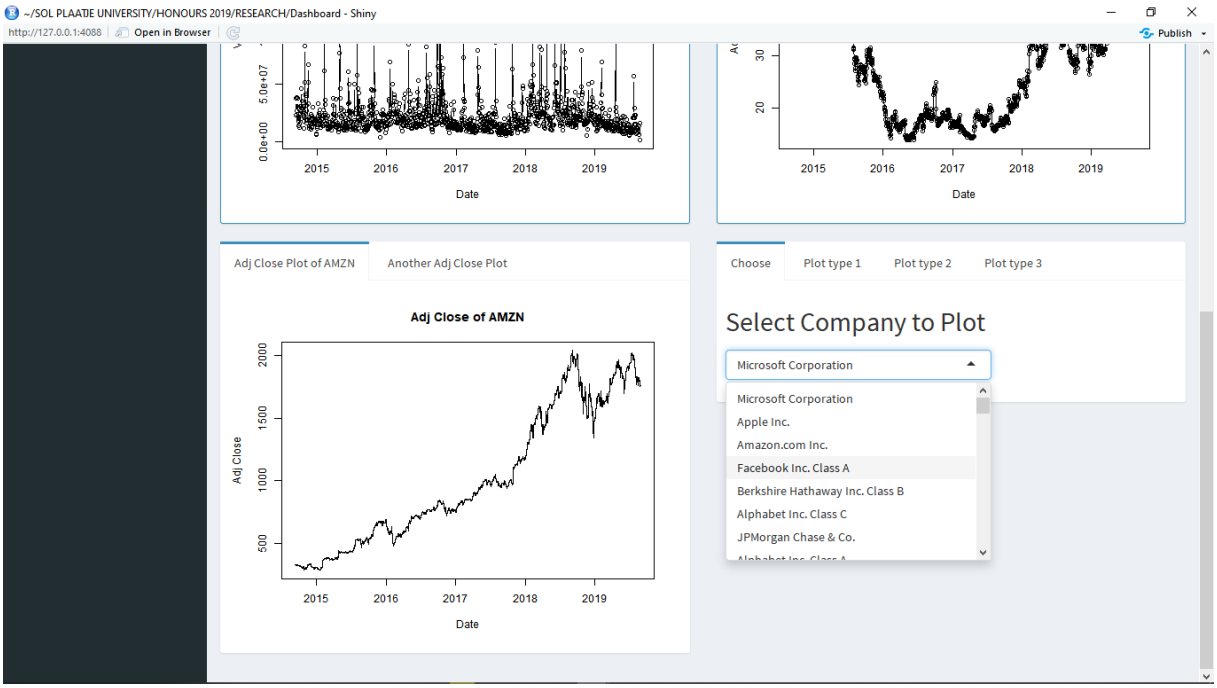


Figure 4.9: Dashboard Overview

Figure 4.9 shows the dropdowm box option the user can use to create plots for the company of their choice in order to gain insights from the past data, and also having an overview of the company's performance within the stock market over the years.

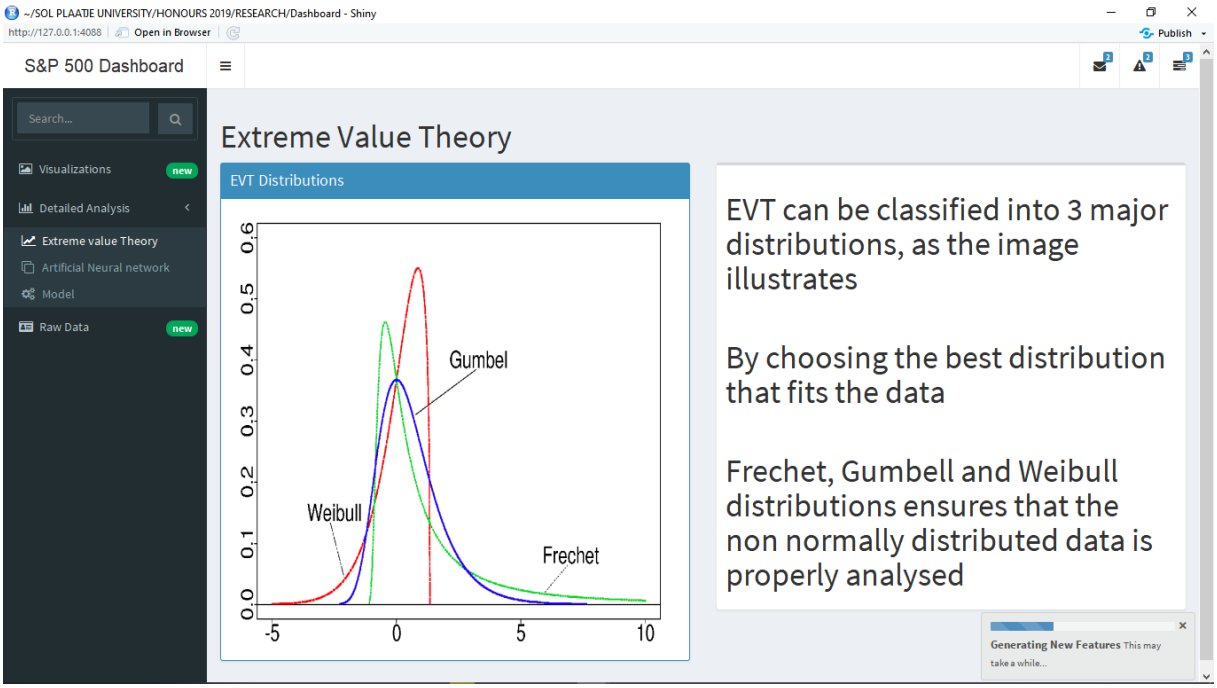


Figure 4.10: Dashboard Overview

The extreme value theory section as shown in figure 4.10 provides a brief explanation

of how it works.

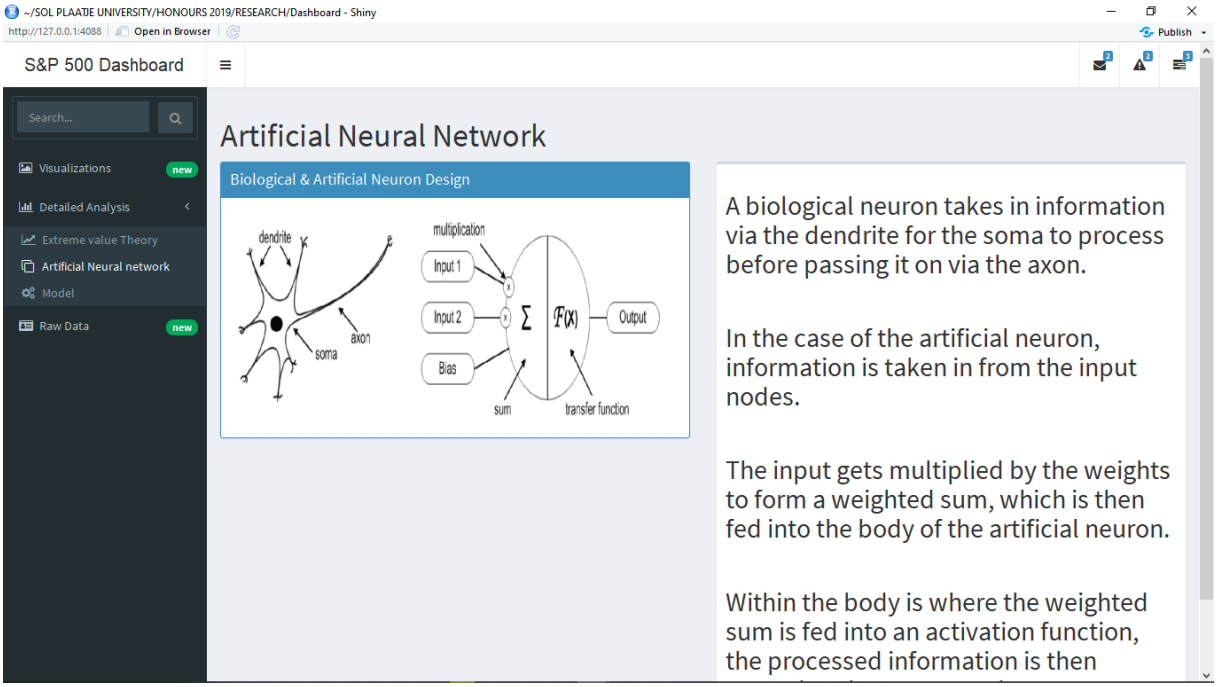


Figure 4.11: Dashboard Overview

The artificial neural network section as shown in figure 4.11 provides a brief explanation of artificial neural networks.

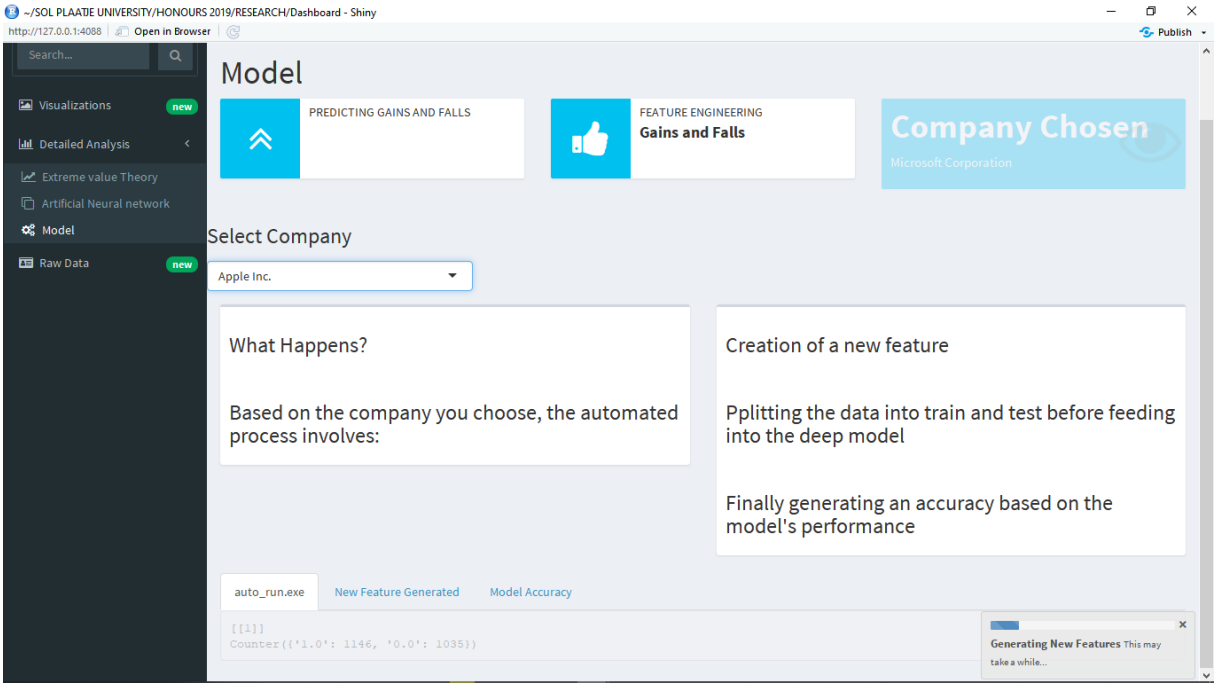


Figure 4.12: Dashboard Overview

The model section allows the user to choose a specific company in order to perform

pre-processes on the chosen dataset in the background. The background processes involve generating extreme value theory parameters, creating a new feature(determining gains and falls), Splitting the dataset into train and tests sets before feeding it into the deep neural network. Once the process is complete the model accuracy for the specified dataset is generated, as well as the k-fold cross validation accuracies.

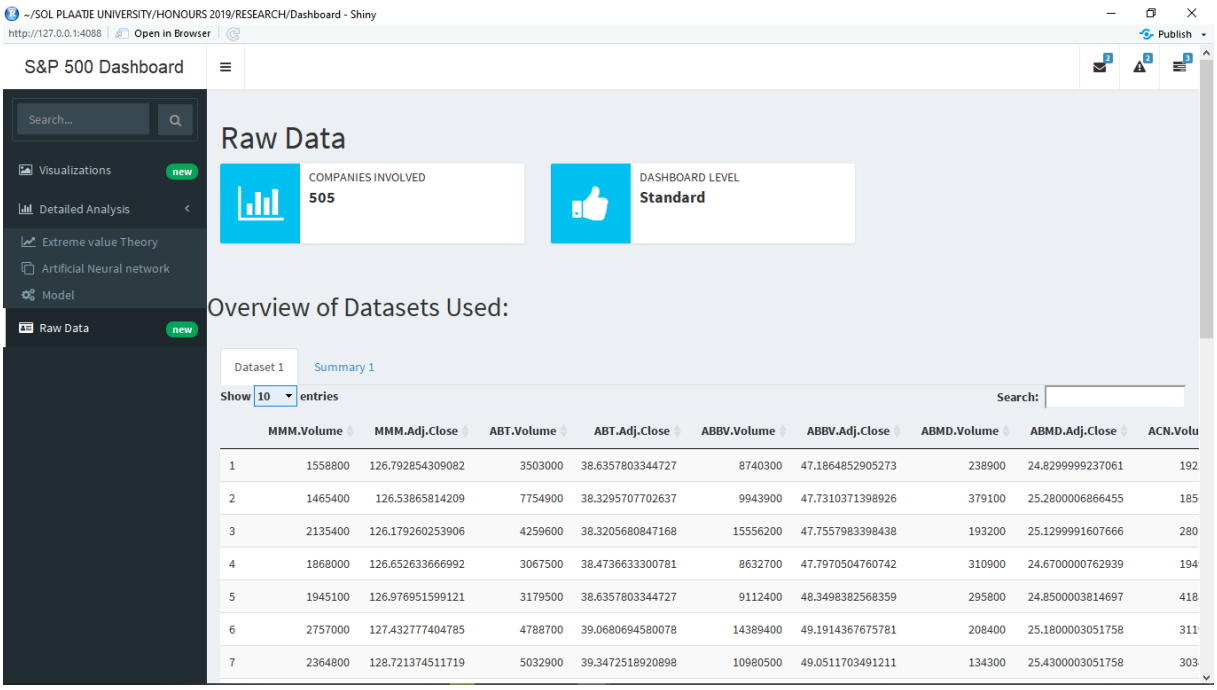


Figure 4.13: Dashboard Overview

The Raw data section provides the user with access to view and download S&P 500 datasets for further exploratory purposes.

Chapter 5

Conclusion(s) & Recommendation(s)

After working with over 500 datasets of the world's biggest companies, getting to spot the trends over the years through visualizations provides insights on company growth over the years. The development of a trading system provides access to open source data manipulations.

EVT forms a large domain of stochastic process which follow one of the 3 distributions namely; Gumbell, Frechet and Weibull, The stochastic nature of stock data makes the combination with EVT a perfect fit.

A combination of EVT, ANN was achieved through the use of machine learning and statistical techniques. A web application stock trading system was developed to help organizations and stock traders alike. The system provides access to visualizations and models of all the companies within the S&P 500.

The result of fufiling the objectives which involves testing for normality to show that stock data is not normally distributed, measuring the tail behaviour of stock data returns with EVT distributions and providing a trading strategy based on a predictive model which integrates EVT and machine learning.

The results show the efficiency of a certain model when compared to that of others in terms of performance and overall accuracy. (as seen in chapter 4, section 3). The ANN models outweighed that of the logistic regression model.

All of these points mentioned achieved the aim by answering the research question that were intially stated:

- How can gains and falls in financial asset returns be modelled using extreme value theory and machine learning.

This research project determined that the combination of EVT, machine learning without neglecting statistics can therefore be used to provide a reliable stock trading system for an end user.

5.1 Limitations

The web application system suffered limitations due to lack of computational resources. It is unable to process stock data generated every second, but instead works with past data that was generated within previous years. Despite of this, the system is still very useful, and performs major functions it was initally intended to, which is providing a stock trading system.

5.2 Recommendations for further Work

Being able to effectively use EVT can prove to be really challenging. Three EVT distributions were used within the scope of this research project, and further work can choose to add more distributions even beyond that of EVT that could possible fit the data better, as oppose to the three used within this reasearch project.

The use of neural networks as a means of classification was due to the empirical studies that justified better predictions using ANNs as opposed to support vector machines or the various other classification algorithms. This justification was further proven due to the excellent accuracy generated by neural network models used for predictions. This raises the question, can a better accuracy be achieved when modeling financial asset returns in order to forecast gains and falls within stock data through the use of other classification methods apart from artificial neural networks?

Chapter 6

Appendix

6.1 GitHub Repository Link

The source codes provided within the appendix chapter are only snippets of the original code, The complete source codes created and used during the duration of this reasearch project can be found at my gihub repository below:

https://github.com/NonchalantLaja/Research_Project_Honours_2019

6.2 Figures

6.2.1 Distributions

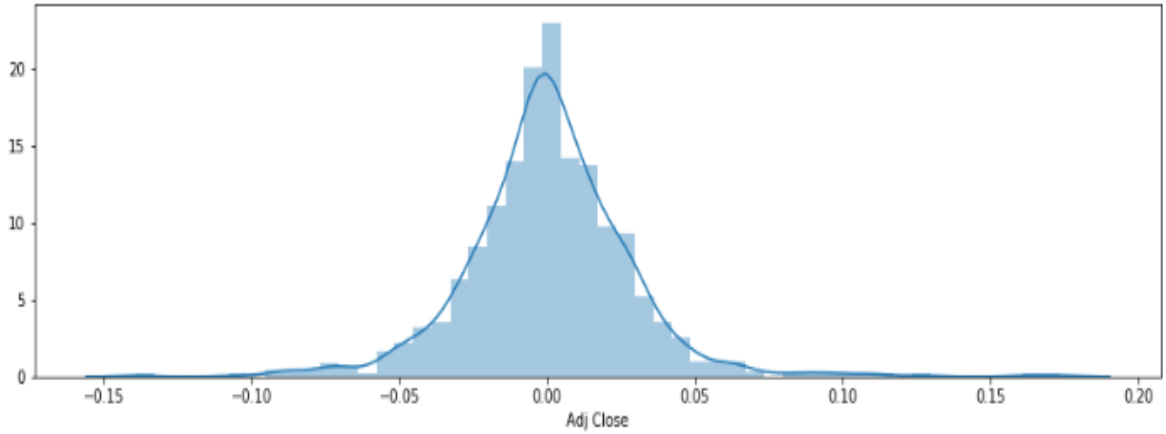


Figure 6.1: TSLA Stock Distribution

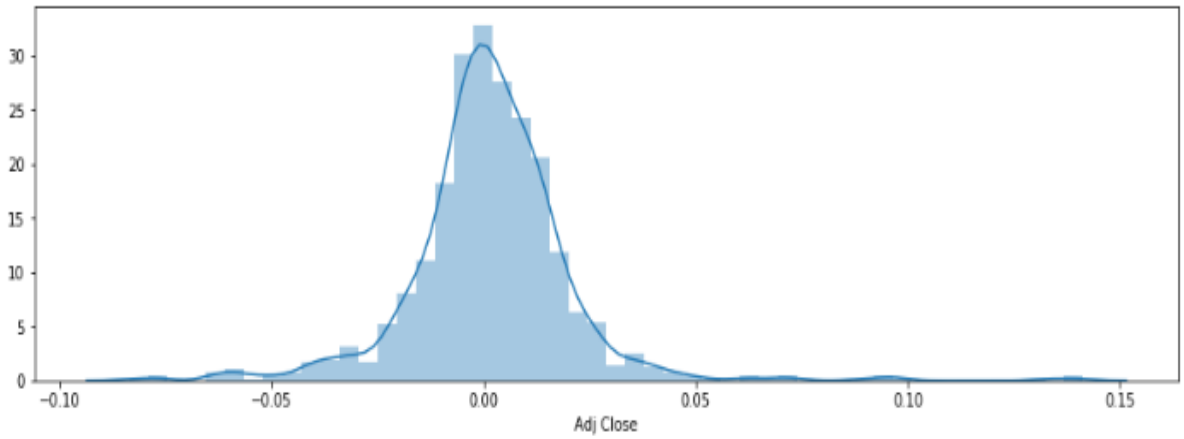


Figure 6.2: AMZN Stock Distribution

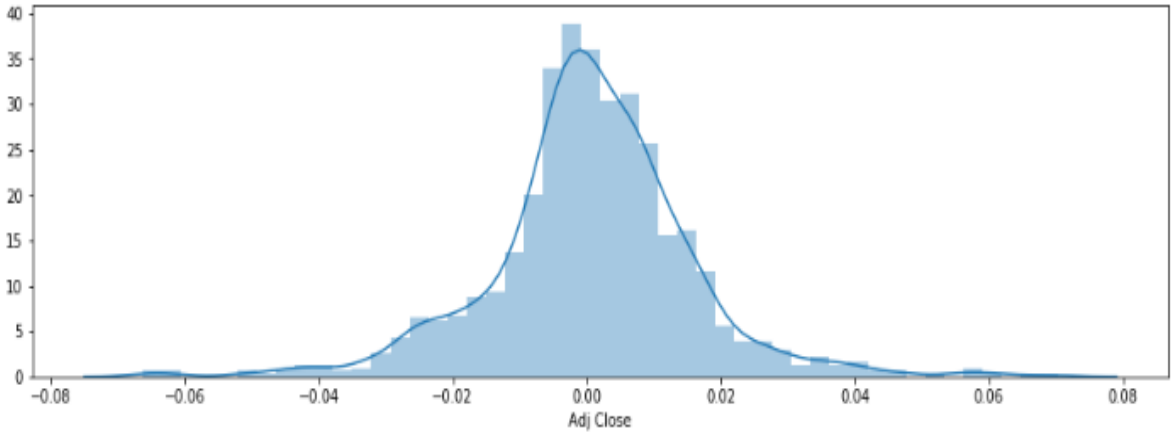


Figure 6.3: AAPL Stock Distribution

6.2.2 Normality Tests

```
In [29]: # Anderson-Darling Test
         from scipy.stats import anderson

In [30]: # normality test
         result = anderson(r.dropna())
         print('Statistic: %.3f' % result.statistic)
         p = 0
         for i in range(len(result.critical_values)):
             sl, cv = result.significance_level[i], result.critical_values[i]
             if result.statistic < result.critical_values[i]:
                 print('%.3f: %.3f, data looks normal (fail to reject H0)' % (sl, cv))
             else:
                 print('%.3f: %.3f, data does not look normal (reject H0)' % (sl, cv))

         Statistic: 24.044
         15.000: 0.574, data does not look normal (reject H0)
         10.000: 0.654, data does not look normal (reject H0)
         5.000: 0.784, data does not look normal (reject H0)
         2.500: 0.915, data does not look normal (reject H0)
         1.000: 1.088, data does not look normal (reject H0)
```

Figure 6.4: Kurtosis Test

```
In [18]: from scipy import stats
         k, p = stats.mstats.normaltest(r.dropna())

In [19]: print(p)

5.509640097009715e-103

In [20]: if p<0.05:
             print('Variable is NOT Normal')
         else:
             print('Variable is Normal')

Variable is NOT Normal
```

Figure 6.5: Kurtosis Test

```
In [23]: # Shapiro-Wilk Test
         from scipy.stats import shapiro

In [24]: # normality test
         stat, p = shapiro(r.dropna())
         print('Statistics=%.3f, p=%.3f' % (stat, p))

Statistics=0.855, p=0.000

In [25]: # interpret
         alpha = 0.05
         if p > alpha:
             print('Sample looks Gaussian (fail to reject H0)')
         else:
             print('Sample does not look Gaussian (reject H0)')

Sample does not look Gaussian (reject H0)
```

Figure 6.6: Shapiro Wilk Test

```
In [26]: # D'Agostino and Pearson's Test
         from scipy.stats import normaltest

In [27]: # normality test
         stat, p = normaltest(r.dropna())
         print('Statistics=%.3f, p=%.3f' % (stat, p))

Statistics=470.920, p=0.000

In [28]: # interpret
         alpha = 0.05
         if p > alpha:
             print('Sample looks Gaussian (fail to reject H0)')
         else:
             print('Sample does not look Gaussian (reject H0)')

Sample does not look Gaussian (reject H0)
```

Figure 6.7: D'Augostino's K^2 Test

6.3 Feature Engineering Source Code

```
In [39]: def gain_fall(*args): # *args lets us pass any parameters, any number of arguments which becomes an iterable
             cols = [c for c in args] # passing each column mapping it row wise
             for col in cols:
                 if(col > 0):
                     return(1)   # GAIN
                 if(col < 0):
                     return(0)# FALL
```

```
In [40]: df['1day_Lag'] = (df['Adj Close'].shift(-1) - df['Adj Close']) / df['Adj Close']
         # .shift shifts up to get the future value old - new divided by

         df['1day_target'] = list(map(gain_fall, *[df['1day_Lag']]))

         df['1day_target'] = df['1day_target'].shift(1)
         df['1day_target'].fillna(0, inplace = True)
```

Figure 6.8: Kurtosis Test

6.4 Model Source Code

```
model2 = Sequential([
    # Flattening the input Layer into a vector because it is a matrix
    Flatten(),

    Dense(128, activation = 'relu'),

    Dropout(0.20),

    #output Layer with 1 neuron with the sigmoid function to counter for the 2 classes (1 or 0)
    Dense(1, activation = 'sigmoid') # or tf.nn.softmax

])
```

Figure 6.9: MLP Architecture

```
model = Sequential([
    # Flattening the input Layer into a vector because it is a matrix
    Flatten(),

    #First hidden Layer
    Dense(256, activation = 'relu'), # or tf.nn.relu

    #to prevent overfitting
    Dropout(0.20),

    Dense(128, activation = 'relu'),

    Dropout(0.20),

    #output Layer with 1 neuron with the sigmoid function to counter for the 2 classes (1 or 0)
    Dense(1, activation = 'sigmoid') # or tf.nn.softmax

])
```

Figure 6.10: Deep Neural Network Architecture

6.5 ShinyDashboard Source Code - R

6.5.1 UI

```
library(reticulate)
library(shiny)
library(shinydashboard)



#######################################################

# Changing the working directory to the source file location
setwd(dirname(rstudioapi::getActiveDocumentContext()$path))

# Dataset 1
data <- read.csv('data/sp500_All_alt.csv')
data$Date = as.Date(data$Date)

# Dataset 2
data2 <- read.csv('data/sp500_new2.csv')

comp_names <- unique(data2$Company)



# Model Things
# Gain and fall script
source_python('gain_fall.py')

# The Model
source_python('py_model_script.py')

# EVT tings
source_python('evt.py')
```

```
##############################################

# Defining the UI for the application
shinyUI(
dashboardPage( title = "Stock Dashboard",  skin = "black",

dashboardHeader(title = "S&P 500 Dashboard",

dropdownMenu(type = "message",
messageItem(from = "Update", message = "New Stock Prices",
time = "04:20"),
messageItem(from = "Charts", message = "Multiple Chart Options",
icon = icon("bar-chart"), time = "29-08-2019")
),
dropdownMenu(type = "notifications",
notificationItem(text = "New menu items added", icon = icon("dashboard"),
 status = "success"),
notificationItem(text = "S & P 500 Loading...", icon = icon("warning"),
 status = "warning")
),
dropdownMenu(type = "tasks",
taskItem(value = 60, color = "red", "Overall daily loss"),
taskItem(value = 75, color = "green", "Trading Percentage"),
taskItem(value = 60, color = "aqua", "Window period")
)
),
dashboardSidebar(

sidebarMenu(

sidebarSearchForm("searchTest", "buttonSearch", "Search..."),
```

```
menuItem("Visualizations", tabName = "Visualizations", icon = icon("image")
,badgeLabel = "new", badgeColor = "green"),

menuItem("Detailed Analysis", tabname = "Detailed",
icon = icon("bar-chart"),

menuSubItem("Extreme value Theory", tabName = "EVT",
icon = icon("line-chart")),
menuSubItem("Artificial Neural network", tabName = "ANN",
 icon = icon("clone")),
menuSubItem("Model", tabName = "model", icon = icon("cogs"))
),
menuItem("Raw Data", tabName = "Raw", icon = icon("id-card"), badgeLabel =
 badgeColor = "green")

)
),
dashboardBody(
tabItems(
tabItem(tabName = "Visualizations",

fluidRow(
column(width = 12,
valueBox(505, "Companies Involved", icon = icon("info-circle"),
color = "green"),
valueBoxOutput("chosen"),
valueBox("Standard & Poor", "The two founding financial companies",
icon = icon("question-circle"), color = "yellow")
)
),

fluidRow(
box(title = "Volume Plot of TWTR", status = "success" , solidHeader = T,
```

```
plotOutput("volume")),
box(title = "Adj Close Plot of TWTR", status = "danger", solidHeader = T,
plotOutput("close"))
),

fluidRow(
tabBox(
tabPanel(title = "Adj Close Plot of AMZN", status = "warning",
solidHeader = T, plotOutput("close2")),
tabPanel(title = "Another Adj Close Plot", status = "warning",
solidHeader = T, plotOutput("close3"))
),
tabBox(
tabPanel(title = "Choose",
selectInput("choose_plot",h2("Select Company to Plot"),
choices = comp_names, selected = comp_names[1])),
tabPanel(title = "Plot type 1", status = "primary",
solidHeader = T, plotOutput("type1"),
downloadButton("download_type1", "Download Plot")),
tabPanel(title = "Plot type 2", status = "secondary",
solidHeader = T, plotOutput("type2"),
downloadButton("download_type2", "Download Plot")),
tabPanel(title = "Plot type 3", status = "warning",
solidHeader = T, plotOutput("type3"),
downloadButton("download_type3", "Download Plot"))
))),
tabItem(tabname = "Detailed", h1("Detailed Analysis")
),
tabItem(tabName = "EVT", h1("Extreme Value Theory"),
fluidRow(
box(title = "EVT Distributions", status = "primary" , solidHeader = T,
img(src = 'distribution.png', width = 500, height = 500)
),
```

```
box(
h1("EVT can be classified into 3 major distributions,
as the image illustrates"), br(),
h1("By choosing the best distribution that fits the data"), br(),
h1("Frechet, Gumbell and Weibull distributions ensures that
the non normally distributed data is properly analysed")
))),
tabItem(tabName = "ANN", h1("Artificial Neural Network"),
fluidRow(
box(title = "Biological & Artificial Neuron Design", status = "primary" ,
solidHeader = T,
img(src = 'bio_art_neuron.png', width = 500, height = 250)
),
box(
h2("A biological neuron takes in information via the dendrite for the
soma to process before passing it on via the axon."), br(),
h2("In the case of the artificial neuron, information is taken in
from the input nodes."), br(),
h2("The input gets multiplied by the weights to form a weighted sum,
which is then fed into the body of the artificial neuron."), br(),
h2("Within the body is where the weighted sum is fed into an activation
function, the processed information is then passed to the output node.")
))),
tabItem(tabName = "model", h1("Model"),
fluidRow(
infoBox("Predicting Gains and Falls", icon = icon("angle-double-up")),
infoBox("Feature Engineering", "Gains and Falls", icon = icon("thumbs-up"))
valueBoxOutput("chosen_model")
),
fluidRow(
selectInput("choose_comp_model",h3("Select Company"),
choices = comp_names, selected = comp_names[1])
),
```

```
fluidRow(
box(h3("What␣Happens?"), br(),
h3("Based␣on␣the␣company␣you␣choose,␣the␣automated␣process␣involves:␣")
),
box(h3("Creation␣of␣a␣new␣feature"), br(),
h3("Fitting␣the␣data␣with␣the␣best␣EVT␣Distribution"), br(),
h3("Splitting␣the␣data␣into␣train␣and␣test␣before␣feeding
into␣the␣deep␣model"), br(),
h3("Finally␣generating␣an␣accuracy␣based␣on␣the␣model's␣performance")
)),
tabsetPanel(type = "tab",
tabPanel("auto_run.exe", verbatimTextOutput("model_running")),
tabPanel("New␣Feature␣Generated", DT::dataTableOutput("new_feature")),
tabPanel("DistPlot␣&␣EVT:␣Goodness␣of␣fit",
fluidRow(
box(title = "Distribution␣Plot", status = "primary", solidHeader = T,
plotOutput('distplot')),
box(title = "Goodnest␣of␣Fit␣Table", status = "info", solidHeader = T,
 verbatimTextOutput("goodness"))
)),
tabPanel("Model␣Accuracy", verbatimTextOutput("model_acc"))

)),
tabItem(tabName = "Raw", h1("Raw␣Data"),

fluidRow(
infoBox("Companies␣Involved", 505, icon = icon("bar-chart-o")),
infoBox("Dashboard␣Level", "Standard", icon = icon("thumbs-up"))
),
fluidRow(br(), h2("Overview␣of␣Datasets␣Used:␣"), br()
),
tabsetPanel(type = "tab",
tabPanel("Dataset␣1", DT::dataTableOutput("company_all"),
```

```
downloadButton("download_data1", "Download Dataset 1")),
tabPanel("Summary 1", verbatimTextOutput("summary1"),
downloadButton("download_summary1", "Download Summary 1"))
),
tabsetPanel(type = "tab",
tabPanel("Dataset 2", DT::dataTableOutput("company_data_table"),
downloadButton("download_data2", "Download Dataset 2")),
tabPanel("Summary 2", verbatimTextOutput("summary2"),
downloadButton("download_summary2", "Download Summary 2"))
))))))
```

6.5.2 Server

```
library(shiny)
library(shinydashboard)

# Defining the server logic required
shinyServer(function(input, output) {

# Plots
output$volume <- renderPlot({

with(data, plot(data$Date, data$TWTR.Volume, type = "b",
main = "Volume Distribution of TWTR", xlab = "Date", ylab = "Volume"))
})
output$close <- renderPlot({

with(data, plot(data$Date, data$TWTR.Adj.Close, type = "o",
main = "Adj Close of TWTR", xlab = "Date", ylab = "Adj Close"))

})
output$close2 <- renderPlot({

with(data, plot(data$Date, data$AMZN.Adj.Close, type = "s",
```

```
main = "Adj Close of AMZN", xlab = "Date", ylab = "Adj Close"))

})
output$close3 <- renderPlot({
with(data, plot(data$Date, data$GOOG.Adj.Close, type = "l",
main = "Adj Close of GOOG", xlab = "Date", ylab = "Adj CLose"))
})
output$chosen <- renderValueBox({
valueBox("Company Chosen ", input$choose_plot, icon = icon("eye") )
})
output$chosen_model <- renderValueBox({
valueBox("Company Chosen ", input$choose_comp_model, icon = icon("eye") )
})
output$type1 <- renderPlot({

with(data2, plot(data2[data2$Company == input$choose_plot,"Date"], data2[da
})
output$type2 <- renderPlot({
with(data2, plot(data2[data2$Company == input$choose_plot,"Date"],
data2[data2$Company == input$choose_plot,"Volume"], type = "b",
main = paste("Volume of ", input$choose_plot), xlab = "Date", ylab = "Volum
})
output$type3 <- renderPlot({
hist(data2[data2$Company == input$choose_plot,"Volume"], xlab = "Volume",
main = paste("Volume Histogram of ", input$choose_plot))
})
output$company <- {(
renderText(input$comp)
)}
#Summaries
output$summary1 <- renderPrint({
summary(data[,-1])
})
```

```
output$summary2 <- renderPrint({
summary(data2[data2$Company == input$choose_comp_raw,])
}
output$new_feature <- DT::renderDataTable({
d <- data2[data2$Company ==  input$choose_comp_model,]
df_and_spread = gain_fall(d)
new_data <- as.data.frame(df_and_spread[1])
subset(new_data, select = -c(1, 2))
})
# Output the model running
output$model_running <- renderPrint({
d <- data2[data2$Company ==  input$choose_comp_model,]
df_and_spread = gain_fall(d)
withProgress(message = "Generating␣New␣Features",
detail = "This␣may␣take␣a␣while...", value = 0,{
for(i in 1:80){
incProgress(1/80)
Sys.sleep(0.25)
}
})
df_and_spread[2]
})
output$goodness <- renderPrint({
})
output$distplot <- renderPlot({
})
output$model_acc <- renderPrint({
# Print out the data spread
d <- data2[data2$Company ==  input$choose_comp_model,]
df_and_spread = gain_fall(d)
withProgress(message = "Running␣Deep␣Model",
detail = "This␣may␣take␣a␣while...", value = 0,{
for(i in 1:80){
```

```
incProgress(1/80)
Sys.sleep(0.25)
}
})
new_data <- as.data.frame(df_and_spread[1])
eval_array <- model(new_data)
eval_array2 <- model2(new_data)
print("Deep Neural Net Module Accuracy % : ")
print(eval_array[[1]][2])
print("MLP Neural Net Module Accuracy % : ")
print(eval_array2[[1]][2])
print("Logistic Regression Model Accuracy % :")
print("10 Fold Cross Validation Accuracies for Deep NN :")
print(eval_array[[2]])
print("10 Fold Cross Validation Accuracies for MLP :")
print(eval_array2[[2]])
print("10 Fold Cross Validation Accuracies for LR")
})

#Datatables
output$company_all <- DT::renderDataTable({
data[,-1]
})
output$company_data_table <- DT::renderDataTable({

data2[data2$Company == input$choose_comp_raw, ]
})
# Download Handels
output$download_data1 <- downloadHandler(
filename = function(){
paste("dataset_1", "csv", sep = ".")
},
content = function(file){
```

```
write.csv(data[, -1], file)
}
)
output$download_summary1 <- downloadHandler(
filename = function(){
paste("summary_1", "txt", sep = ".")
},
content = function(file){
write.txt(summary(data[, -1]), file)
}
)
output$download_data2 <- downloadHandler(
filename = function(){
paste("dataset_2", "csv", sep = ".")
},
content = function(file){
write.csv(data2, file)
}
)
output$download_summary2 <- downloadHandler(
filename = function(){
paste("summary_2", "txt", sep = ".")
},
content = function(file){
write.txt(summary(data2), file)
}
)
output$download_type1 <- downloadHandler(
filename = function(){
paste(paste("Adj-Close-of-", input$choose_plot), "png", sep = ".")
},
content = function(file){
png(file)
```

```
with(data2, plot(data2[data2$Company == input$choose_plot,"Date"], data2[da
dev.off()
}
)
output$download_type2  <- downloadHandler(
filename = function(){
paste(paste("Volume-of-", input$choose_plot), "png", sep = ".")
},
content = function(file){
png(file)
with(data2, plot(data2[data2$Company == input$choose_plot,"Date"], data2[da
dev.off()
})
output$download_type3 <- downloadHandler(
filename = function(){
paste(paste("Volume-Histogram-of-", input$choose_plot), "png", sep = ".")
},

cntent = function(file){
png(file)
hist(data2[data2$Company == input$choose_plot,"Volume"], xlab = "Volume", m
dev.off()
})# New Additions (Temp) Detailed Analysis
})
```

Bibliography

Aydogdu, M. (2018). Predicting stock returns using neural networks. *SSRN Electronic Journal*.

Balance, T. (2019). The S & P 500 and how it works. `https://www.thebalance.com/what-is-the-sandp-500-3305888`. [Online; accessed 29-Sep-2019].

Burdorf, T. and van Vuuren., G. (2018). An evaluation and comparison of value at risk and expected shortfall. *DInvestment Management and Financial Innovations*.

Chen, A.-S., Leung, M. T., and Daouk, H. (2003). Application of neural networks to an emerging financial market: forecasting and trading the taiwan stock index. *Computers & Operations Research*, 30(6):901–923.

De Haan, L. and Ferreira, A. (2007). *Extreme value theory: an introduction*. Springer Science & Business Media.

Friederichs, P. (2007). An introduction to extreme value theory. Meteorological Institute; University of Bonn.

Global, S. . P. (2019). S & P Global. `https://www.spglobal.com/en/who-we-are/our-history`. [Online; accessed 30-Sep-2019].

Hsu, C.-W. and Lin, C.-J. (2002). A comparison of methods for multiclass support vector machines. *IEEE transactions on Neural Networks*, 13(2):415–425.

Index, M. (2019). Yahoo Finance API. `https://www.marketindex.com.au/yahoo-finance-api`. [Online; accessed Aug-2019].

Indices, S. . P. D. J. (2019). S & P 500. `https://us.spindices.com/indices/equity/sp-500`. [Online; accessed 30-Sep-2019].

Investopedia (2018). Financial Asset. https://www.investopedia.com/terms/f/financialasset.asp. [Online; accessed 01-Aug-2019].

Krenker, A., Bester, J., and Kos, A. (2011). An introduction to artificial neural networks. *Methodological Advances and Biomedical Applications*.

Kukreja, H., Bharath, Siddesh, and Kuldeep (2016). An introduction to artificial neural network. *Department of Electrical & Electronics Engineering*.

Liu, J. and Zio, E. (2018). Integration of feature vector selection and support vector machine for classification of imbalanced data. *Applied Soft Computing Journal*.

Makhwiting, R., Sigauke, C., and Lesaoana, M. (2014). Modelling tail behaviour of returns using the generalized extreme value distribution. *Economics Management and Financial Markets*.

Mastery, M. L. (2019). A Gentle Intoduction to Normality Tests in Python. https://machinelearningmastery.com/a-gentle-introduction-to-normality-tests-in-python/. [Online; accessed 01-Aug-2019].

McNeil, A. J. (1999). Extreme value theory for risk managers. *Departement Mathematik ETH Zentrum*.

Musah, I. A.-A., Jianguo, D., Khan, u. d., Salah, H., Alhassan, A., and Abdul-Rasheed, A. (2018). The asymptotic decision scenarios of an emerging stock exchange market: Extreme value theory and artificial neural network. *Risks*, 6(4):132.

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rencberoglu, E. (2018). Fundamental Techniques of Feature Engineering for Machine Learning. https://towardsdatascience.com/feature-engineering-for-machine-learning. [Online; accessed 1-Oct-2019].

Rollins, J. (2015). Why we need a methodology for data science. https://www.ibmbigdatahub.com/blog/why-we-need-methodology-data-science. [Online; accessed 29-Sep-2019].

Shah, V. H. (2007). Machine learning techniques for stock prediction. *Foundations of Machine Learning— Spring*, 1(1):6–12.

Shekhar, A. (2018). What Is Feature Engineering for Machine Learning. `https://medium.com/mindorks/what-is-feature-engineering-for-machine-learning`. [Online; accessed 1-Oct-2019].

Tang, F., Adam, L., and Si, B. (2018). Group feature selection with multiclass support vector machine. *Neurocomputing*.