



Analysis and Prediction of Stock Prices

by

Nishant Verma
D00251774

Supervised by Dr David O’Keeffe

Dundalk Institute of Technology
School of Informatics and Creative Arts
Department of Computing Science and Mathematics

September 2022

Acknowledgements

Firstly, I would like to thank my parents for letting me pursue that I like in my life. Without them, I would not be here and never would have gotten the opportunity to put my skills at test at this level. I'd also like to thank my grandfather who is also my idol. He has always motivated me and pushed me to achieve greater heights.


I would like to thank the management of Dundalk Institute of Technology for providing us the access of many learning platforms like Khan academy, Data camp at no extra cost. It really helped me learn new technologies required for my dissertation.

I would like to thank my professors, Dr Abhishek Kaushik for always keeping a check on me if I am struggling with my dissertation or any other subject, Dr Jack McDonnell for showing me a clear path to follow with my dissertation and my supervisor, Dr David O'Keeffe for providing a tremendous support since the day one. This project would never be possible without the support of my professors.

At last, I would like to thank my friends and my family away from my home, my housemates for their constant support throughout the year.

Declaration

I hereby declare that the work described in this project is, except where otherwise stated, entirely my own work and has not been submitted as part of any degree at this or any other Institute/University

Signed	 _____
Name	<u>Nishant Verma</u>
Date	<u>September 9, 2022</u>

Abstract

This is the era of big data where stock prices or trend prediction has become more famous than ever before by leveraging the data. I have collected the data of 5 years of stock prices from the Stock market index- S&P 500. The purpose of the study is to know whether stock prices can be predicted by making use of its historical data only. I have proposed a comprehensive approach of feature engineering with machine learning algorithm and deep learning approach. The main focus of the study was short-term predictions. By short-term, I mean the stock prices in the upcoming week or next 10 days but definitely not a month or later. The study includes the preprocessing of the stock market dataset, feature engineering, building different types of models and then lastly the comparison and evaluation of the models. It was expected from deep learning approach to have better accuracy but surprisingly the frequently used machine learning algorithm with feature engineering outperformed the deep learning approach. Generic results were also obtained by making use of the stock prices of top 50 companies of the index for backing the findings.

Keywords: *Stock market, feature engineering, prediction, deep learning.*

The code of the project and all the files associated with it is available at the link- <https://github.com/nishantverma1206/Stock-Prediction>. I have created a screencast for the demonstration of the code functionality of this project, it can be accessed at the link - <https://web.microsoftstream.com/video/dcad241d-8a81-4caf-ad31-76a24bc2e32e>. It would be preferred to go through the description before playing the screencast as it would help to navigate through it better.

List of abbreviations and symbols

AI	Artificial Intelligence
NN	Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-term Memory
MAE	Mean Absolute Error
MSE	Mean Square Error
RMSE	Root Mean Square Error
ANN	Artificial Neural Network
GA	Generic Algorithm
KOSPI	Korea Stock Price Index
PCA	Principal Component Analysis
HMM	Hidden Markov Models
RS	Rough Set
WNN	Wavelet Neural Network
SVM	Support Vector Machine
F_SSFS	F-score and Support Sequential Forward Search
BPNN	Back Propagation Neural Network
TEJD	Taiwan Economic Journal Database
ReLU s	Rectified Linear Units
SGD	Stochastic Gradient Descent
SSECI	Shanghai Stock Exchange Composite Index
SVRE	Support Vector Regression Ensemble
NNRE	Neural Network Regression Ensemble
S&P 500	Standard and Poor's 500
NYSE	New York Stock Exchange
ADF	Augmented Dickey Fullers
ARIMA	Autoregressive Integrated Moving Average
BRNN	Bidirectional Recurrent Neural Network
GRU	Gated Recurrent Units
MA	Moving Average
EWMA	Exponential Weighted Moving Average

SMA	Simple Moving Average
ACF	Auto Correlation Function
PACF	Partial Auto Correlation Function
LR	Linear Regression
EDA	Exploratory Data Analysis

Table of Contents

Acknowledgements.....	i
Declaration.....	ii
Abstract.....	iii
List of abbreviations and symbols	iv
Table of Contents.....	vi
List of Figures	viii
List of Tables	x
List of Equations.....	xi
1. Introduction.....	12
2. Literature review	18
2.1 Related Work	18
2.2 Stock Market Index.....	22
2.3 Time Series Analysis	24
2.3.1 Autocorrelation	25
2.3.2 Seasonality	26
2.3.3 Stationarity.....	26
2.4 Survey on stock price prediction algorithms	27
2.4.1 Auto-Regressive Integrated Moving Average (ARIMA)	28
2.4.2 Linear Regression	30
2.4.3 Long Short-Term Memory (LSTM)	31
3. Data and Methods	36
3.1 Data Source and Description	36
3.2 Data Preparation	38
3.3 Data Exploration.....	41
3.3.1 Resampling	42
3.3.2 Smoothing (MA).....	43
3.3.3 Exponential Weighted Moving Average (EWMA)	43
3.3.4 Test for stationarity	44
3.4 Data Pre-processing	44

3.4.1 For ARIMA.....	44
3.4.2 For Linear Regression.....	45
3.4.3 For LSTM	46
3.5 Core Technologies and Libraries.....	48
3.6 Ethical Considerations	48
4. Results and Discussions	50
4.1 Insights from Data Exploration	50
4.1.1 Why Closing price?.....	53
4.2 Models Implementation and Evaluation	53
4.2.1 ARIMA (Base Model)	53
4.2.2 Linear Regression	59
4.2.3 Long Short-Term Memory (LSTM)	60
4.3 Generalised Results	62
5. Conclusion and Future Work	65
6. Appendices.....	67
6.1 Exploratory Data Analysis (EDA) and Base Model.....	67
6.2 Hyper-tuning of ARIMA parameters.....	83
6.3 Linear Regression Model.....	85
6.4 LSTM Model	89
6.5 Top 50 stocks.....	97
6.6 Ethical clearance documents.....	104
7. References.....	114

List of Figures

Figure 1: CRISP-DM Diagram. (Source:Wikipedia)	16
Figure 2: Example of Autocorrelation	25
Figure 3: Example of seasonality	26
Figure 4: Example of stationarity	26
Figure 5: Library for ADF test.....	27
Figure 6: Code to implement ADF test.....	27
Figure 7: Example of Correlation	29
Figure 8: LSTM Cell.....	32
Figure 9: Data Information	37
Figure 10: First 5 records of Dataset	37
Figure 11: First 5 rows of Apple Stocks data	38
Figure 12: Code snippet to update datatype of column date.....	39
Figure 13: First 5 rows of Apple stock prices with dates as index	39
Figure 14: Plot of Apple Stock prices.....	40
Figure 15: Closing price of American Airline Group.....	41
Figure 16: Closing price of Cerner Corporation	41
Figure 17: Closing price of Alphabet Inc. (Google)	42
Figure 18: Code snippet of splitting the data for ARIMA.....	45
Figure 19: Function for lag creation	45
Figure 20: Series and its lags in a dataframe	46
Figure 21: Creation of training and testing data for linear regression	46
Figure 22: Log returns of closing prices	47
Figure 23: Labelling process for LSTM	47
Figure 24: Yearly minimum and maximum stock price	50
Figure 25: Quarterly minimum and maximum stock price.....	50

Figure 26: Monthly minimum and maximum stock price	51
Figure 27: Smoothing by Simple Moving Average.....	51
Figure 28: SMA vs EWMA (Window size:20)	52
Figure 29: SMA vs EWMA (Window size:30)	52
Figure 30: EWMA Plot with different alpha	52
Figure 31: ADF test on closing price	54
Figure 32: ACF plot of closing prices	54
Figure 33: ACF plot of differenced (once) closing prices	54
Figure 34: ACF plot of differenced (twice) closing prices	54
Figure 35: Code snippet to implement ndiffs function	55
Figure 36: PACF plot of differenced (once) closing prices.....	55
Figure 37: Summary of ARIMA model.....	56
Figure 38: Actual vs Predicted of ARIMA	57
Figure 39: RMSE of ARIMA	57
Figure 40: Code snippet of Auto ARIMA	59
Figure 41: Actual vs Predicted of Linear regression	60
Figure 42: RMSE of Linear Regression	60
Figure 43: Actual vs Predicted of LSTM.....	61
Figure 44: RMSE of LSTM.....	62

List of Tables

Table 1: Data Description	36
Table 2: Core technologies used in project.....	48
Table 3: RMSE of Top 50 companies	64
Table 4: Potential answers to research questions.....	65

List of Equations

Equation 1: Autocorrelation.....	25
Equation 2: Auto Regressive model	29
Equation 3: Moving Average model.....	29
Equation 4: Simple linear regression	31
Equation 5: Multiple linear regression.....	31
Equation 6: Forget gate.....	33
Equation 7: Input gate.....	34
Equation 8: New information	34
Equation 9: Output gate	34
Equation 10: Hidden state.....	35
Equation 11: SoftMax activation function.....	35
Equation 12: Exponential Weighted Moving Average.....	43

1. Introduction

Stock market has always been a hot topic for investors and traders as their only goal is to multiply the amount of capital they put in and drive the profit out of it. The humanity has witnessed a great advancement in technology sector over the last few decades which has changed the ways of doing businesses all around the world (Rouf et al. 2021). There's one word 'Fintech' which has been doing rounds around the world in last few years comes from two different words i.e., Finance and Technology, this shows how the technology has been impactful in financial industry. Over the last few years, trading and investing in stocks has stolen all the limelight in the financial market when it comes to association with technology. Investors and traders seek applications and tools that would make trading or investing easier, would maximize the profit and lower the risk (Upadhyay and Bandyopadhyay 2012). Stock market prediction has always been an intriguing matter for investors and technologists. In this research, my aim is to develop a prediction model that emphasizes on short-term stock price trend prediction.

Stock market prediction is known to be notoriously difficult task due to its nature of being highly volatile, stochastic, non-linear and having high level of noise (Tan et al. 2007). Stock prices depends on the various mini and macro factors ranging from profits to accounting errors or scandals (Gupta et al. 2013). Gathering all the information that affects stock prices is entirely impossible and also is illegal to make use of sensitive and confidential information for trading stocks (Insider Trading). In this research, I have collected the previous prices of stocks and trying to estimate the future trend based on historical data which is publicly available and can be accessed by anyone. In the world of academic, the stock market prediction is a problem of simple time-series analysis & forecasting in which the historical data is examined and based on that, the next values are estimated (Rouf et al. 2021). The efficient market hypothesis describes that stock prices are significantly dependent on the new information and follows completely random path hence, the stock prices cannot be entirely predicted based on the historical data (Yaes 1989) . This theory stood valid until the researchers made use of advanced technology and showed that the stock prices could be predicted to a certain extent from the historical data. The performance of any mathematical model depends on the on the inputs (features) used to develop

it and so is the case with stock prediction system, the better the quality of features, the better the performance would be of the system (Inthachot et al. 2016). In 2003, a team of researchers conducted an experiment by building a neural network to predict the stock prices by keeping their main focus on the volume of shares traded (Wang et al. 2003). The significance of volume turned out to be irrelevant in the dataset that they had used to build the model, the data belonged to the stock index S&P 500 and I am using the data of same stock market for this research. Ince and Trafalis (2008) aimed at short-term forecasting and applied support vector machine (SVM) to predict the stock prices. The entire idea of their research was to compare the results of multi-layer perceptron and support vector machine for stock predictions and SVM emerged as a winner in this comparison although the results differ by different trading strategies. Around the same time, the researchers having financial background were applying traditional statistical methodologies and signal processing techniques to visualize and predict stock prices. In the past, optimization techniques such as principal component analysis (PCA) has also been used in stock price prediction for fetching the most relevant features in the data engineering in order to attain the maximum possible accuracy (Lin et al. 2009). Over the last few years, researchers have not limit themselves in analyzing the stock prices but also took volume (number of shares) into consideration and have witnessed the change in the prediction's accuracy. This change does not only imply the significance of the volume of share but also redefine the scope and potential of the field (Shih et al. 2019). As the artificial intelligence (AI) methodologies and systems grown over the last few years, the researchers tried to make use those evolved system by combining machine learning and deep learning methods and developed a new metrics which later used as a feature in the process of feature engineering (Liu and Wang 2019). Such types of study would help us in our research for the purpose of feature engineering when implementing machine learning algorithms or while implementing any neural network methodology. Liu, Zhang and Ma developed a convolutional neural network (CNN) and long short-term memory (LSTM) neural network to examine the quantifiable approach in stock market (Liu et al. 2017). The CNN is applied for stock filtration approach, which automatically pulls the relevant features based on the quantifiable data and then proceeds to LSTM to keep the time-series affect intact for better results. Recent research has also introduced a similar kind of hybrid neural

network, combining convolutional neural network with bidirectional long short-term memory to analyze and predict the stock market trends (Eapen et al. 2019).

Based on the background study, it is most likely implied that either a machine learning or neural network-based model would provide me the better and accurate results at predicting stocks in near future trends than traditional and general time-series model like ARIMA. However, a basic time-series ARIMA model has been assembled to provide a base model which was later be used as a comparison with other methodology models.

At the very beginning of the project when only the idea of the project was proposed, I had put forward a few research questions which were quite generic – 1) Which market index to be considered for the project, 2) Should I target the single company or all the companies of the stock exchange, 3) What would be the source of data, 4) Is the data collected ethically right for the use in the research, 5) What are all the technologies and methodologies to be used for the research and 6) What would be final metric or final way to evaluate the models and results. Through the background research and study of this domain, I was able to answer the most of them already like – We are conducting the research on stock market index- Standard and Poor's 500 (S&P 500), tracks the performance of large 500 companies in the United States. From the research, one thing became crystal clear that all the company's stocks data cannot fed at once into any model hence each company's stocks data would be treated and dealt with separately. I chose an open-source data repository platform to collect the data from because of mainly two reasons, one is the platform being open-source hence there's no ethical compliances associated with any entity of the platform and every dataset available on the platform can be utilized by any individual in any possible way without the consent or any permission. However, the right way is to seek permission from the owner of the dataset via email, just to be on the safe side. This answers my research question 3 and 4, related to the source and ethical compliance of the dataset. The plan is to make use of the technologies learned over the course like Python programming for the coding, Jupyter notebook as the editor. Several libraries and frameworks like pandas for data analysis, numpy for numerical operations, matplotlib and seaborn for data visualization, stats model for implementing statistical models, scikit learn to implement machine learning models and many more. I have planned to carry out a new framework algorithm i.e., Long

short-term memory neural network which comes from the family of recurring neural network (RNN). I learned RNN-LSTM from the open-source resources available on the internet and implemented the same in the research. There are various metrics available like root mean square error (RMSE), mean square error (MSE), mean absolute error (MAE) to evaluate the performance of the state-of-art model. In the process of the research over the last few months, the research questions have now been revised and have become more focused and precise. The overall objective of the research can be encapsulated by the following research questions-

- What would be the best suited parameters to implement ARIMA efficiently?
- Which power transformation method would be appropriate for the data?
- Does the data follow any seasonality and trend?
- How would the data help to develop neural network-based prediction model?
- Which model would have the higher accuracy?
- What would be criteria of choosing the best model?

The general behavior of the stocks are the changing prices at a regular interval of time hence this behavior makes it risky to invest the money. The common view of society towards the stock market is that it is highly uncertain for the purpose of investment, and it cannot be trusted at all with one's money and that's majority of people are not even interested to put their money in trading by calling it the gamble. There are many factors that affects stock market, and it is indeed true that it is impossible to have access to all the factors responsible for stock market movement but the factors like season variance and stable movement of any index can help both new traders as well as active traders in understanding the market and make better informed trades (Rao et al. 2020). These types of problems could be well countered by Time-series analysis to see the pattern movement of the market or even forecast the future values to some extent. Financial markets seem to be unpredictable and sometime even illogical. Due to such nature of the market, it gets really difficult to find the patterns in the ill-structured financial data and that's where the data analytics comes into play. In order to find the hidden patterns and model the ill structured data, tools of data analysis like machine learning and time-series algorithms are required. The highly efficient methodologies in this area of field are machine learning and deep learning which are capable of unearthing hidden patterns within the data, predict the future movement and also capable of finding complex co-relations

between features which further enhances the accuracy of the results (Shahrour and Dekmak 2022).

For this research, I followed the CRISP-DM methodology (Wirth and Hipp 2000). The CRISP-DM stands for cross industry standard process for data mining, is a standard process that operates as a foundation for data science project. CRISP-DM consists of 6 steps –

1. Business Understanding – What is the requirement of the business?
2. Data Understanding – What all the data we already have, what more do we need and from where do we need?
3. Data Preparation – Is the data organized enough for the modelling or it needs some manipulation and wrangling?
4. Modeling – Selecting various algorithms which fits fine with data and serves the purpose of the business goal.
5. Evaluation – How the model created in last step performing with the test data or unforeseen data?
6. Deployment – How the stakeholders can be benefitted from the model results?

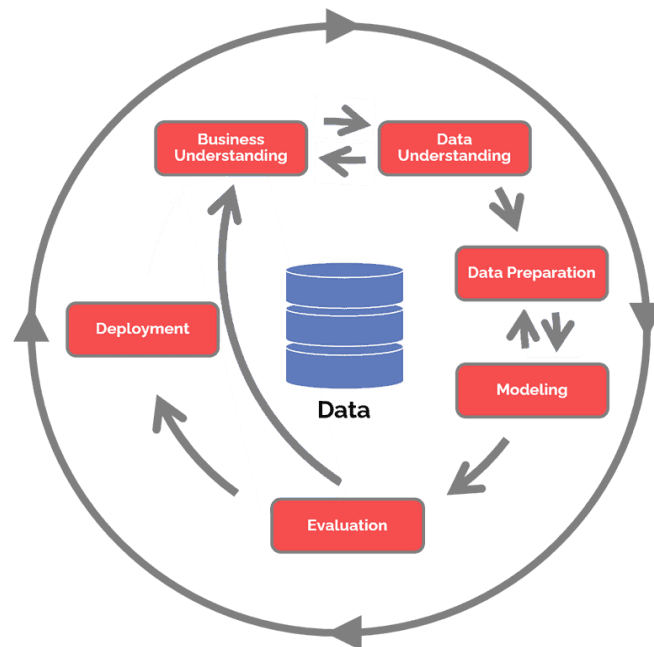


Figure 1:CRISP-DM Diagram. (Source:Wikipedia)

The remaining of this report is structured as follows. “Literature review” that contains relevant work done in the similar field previously and the methodologies used in the project. The “Data and Methods” that describes about data collection

methods, description of the data, data processing done in the project and ethical considerations. The “Results and discussion” describes about preliminary result, data modelling, model evaluation and hyper-tuning. The “Conclusion” clearly states what was being done throughout the project and what could be the future of the project.

2. Literature review

2.1 Related Work

In this part of the report, we have reviewed the previous related work in the field stock price prediction from both the domains- financial and technology.

In 2000, Kim and Han (2000) proposed an approach of genetic algorithm (GA) in combination with artificial neural networks (ANN) with discretization of features to predict the stock price index. The data which they have used for their study have some technical indicators and also the daily change of direction in Korea stock price index (KOSPI). They have made use of 2928 days of trading data for their research starting from January 1989 to December 1998. They have done feature engineering precisely and have performed feature selection methodology. They have applied a technique for optimization of feature discretization which is similar to the feature reduction technique like PCA. The highlight of their work is the introduction of GA to improve the ANN. GA is generally used to improve the learning algorithm itself but here it was also simultaneously utilized for optimizing connection weights between the layers and feature discretization thresholds. There are many limitations to this work due to the evolution in technology sector. For an instance, the learning process of ANN is flawed as the researcher kept the focus only on two factors for optimization. A similar kind of research was conducted in 2016 by Qui and Song in which they have also used ANN to predict the movement of the stock price of the very next day and then they have used GA to enhance the accuracy of the ANN results as an optimization technique (Qiu and Song 2016).

In 2004, Piramuthu carried out detailed research on various features selection methodologies for data extracting applications. For this research, he used diverse dataset that included credit card approval data, traffic on the web data, data of loan defaulters, tam, kiang data and at the end performed a thorough evaluation on how different feature selection methods are enhancing the performance of decision tree. In this study he used several inter-classes and probabilistic distance-based feature selection techniques to study the effectiveness of each technique in processing the input data that would enhance the performance of decision tree(Piramuthu 2004). The aforementioned author used several inter-class and probabilistic distance-based feature selection methodologies to evaluate their result is the highlight for this

research. Although, the algorithm used for evaluation is decision tree which is not capable of handling complex and larger dataset hence we are not sure if the feature selection methodology would perform efficiently when exposed to a larger dataset with a complex model.

In 2005, Hassan and Nath proposed an approach of Hidden Markov Models (HMM) for predicting the stock price of inter-related stock indexes. They have narrowed down the features to 4 i.e., opening price of the stock, closing price of the stock, highest price reached in a day and the lowest price in a day of the stock. They performed this study on the stock prices of four different airlines. The study was conducted on only 2 years of data of stock prices for training and testing (Hassan and Nath 2005). The best part of the HMM approach is that it does not require a prior experience in modelling and is a straightforward simple approach. The study fails to generalize the model and is more sensitive to the airlines data hence we cannot be sure if the same approach would be fine with more complex and bigger data. Another limitation of the study would be the size of the data that they have used. The overall dataset isn't so big hence the size of data over which the evaluation has been performed gets smaller. Such a study could easily be exploited in comparison work with relevant studies in the field of stock price predictions.

In 2018, an integrated approach of Rough set (RS) and Wavelet Neural network (WNN) was introduced to enhance the stock price trend prediction capability. In this research, the author first introduced RS as an optimization technique to reduce the features and later the RS was brought in again to decide the structure of neural network i.e., WNN in this study. They made use of quite diverse dataset consisting of several stock indexes – Dow Jones Index (USA), Nikkei 225 Index (Japan), All Ordinaries Index (Australia), CSI 300 Index (China) and one the last SSE Composite Index (China). Since the author made use of data of various stock markets around the world, the result was good enough with generalization upon evaluation. It was indeed a great idea of using Rough Set as an optimization technique before processing as it brought down the complexity level in computation (Lei 2018). The authors only emphasized on the parameter manipulation and have not discussed anything about the weakness of the study. We observed that since the model was built on the stock market indexes, we cannot be sure if the results would be the same if particular stocks data would be fed in the model.

In 2009, Lee had developed a stock price trend predictive model based on an algorithm Support Vector Machine (SVM). He had used a hybrid feature selection methodology with SVM to predict the stock price trend. The hybrid feature selection technique is named as F-score and Support Sequential Forward Search (F_SSFS), has the capabilities of filter and wrapper techniques to pull out the most subset of most optimum features from the original set of features. In terms evaluation, he compared the performance of the SVM based predictive model in combination with F_SSFS optimization technique with the back propagation neural network (BPNN). He used BPNN for evaluation along with three common feature selection techniques – Information Gain, Correlation based feature selection through paired T-test and Symmetrical Uncertainty. The dataset used for this study is a sample subset of NASDAQ Index from Taiwan Economic Journal Database (TEJD). He performed parameter hyper tuning of kernel function value of SVM by carrying out the grid search methodology which is said to be most expensive function in terms of resources, time complexity and computation power. In the study, the results from the SVM outperform the results from BPNN (Lee 2009). The only limitation found in the study that SVM was only compared with BPNN but not any other machine learning algorithm.

In 2019, Cont and Sirignano proposed a deep learning system built on financial markets' universal feature set. The dataset used for the study contained the records of buying and selling of all the transactions and cancellations of market orders for almost a thousand NASDAQ stock via the database having market order records of the stock exchange. The neural network is comprised of 3 different layers with Long short-term memory (LSTM) units and a feed forward layer with Rectified Linear Units (ReLUs), along with the Stochastic Gradient Descent algorithm (SGD) as an optimization mechanism (Sirignano and Cont 2019). They were able to produce a generalized predictive model which was able to cover the stocks outside the training data as well. They have mentioned a lot of benefits of using the universal model, but it must have been an expensive task to train the model. Due to unclear code of deep learning, it is ambiguous if the irrelevant features were dropped off before feeding the data into the model. Authors have acknowledged a limitation in the paper that it would have been better to carry out a feature selection methodology before training

the data as it would have reduced the resource constraints, space complexity and computational power.

In 2011, Ni et al. proposed stock price trend predictive model that was based on Support Vector Machine (SVM). The main idea of the study was how the optimized feature selection can make the difference in the results of predictive model hence they came up with an approach of fractal feature selection as an optimization technique. The dataset that they used had 19 features as technical indicators from Shanghai Stock Exchange Composite Index (SSECI). Prior training the model with data, they made sure that the data processed and ready for training by carrying out feature selection techniques. They performed grid search for finding the best suited parameters, which is K-Fold cross validation. Moreover, the evaluation of feature selection methodologies is also quite broad. In this study, they kept their main focus on the technical indicators came in the dataset, they did not pay attention to any factors related to financial domain as they also have mentioned in their conclusion (Ni et al. 2011).

In 2018, McNally et al. proposed a recurring neural network (RNN) – Long short-term memory (LSTM) solution with a goal of predicting the direction of bitcoin price in USD. The data used in this study was taken from the Bitcoin Price Index. The dataset included the prices of Bitcoin of each day ranging from 19th August to 2013 to 19th July 2016. They used Boruta Algorithm for a specific purpose of feature engineering as an optimization technique. Boruta algorithm works in a similar way to the random forest classifier. Boruta was originally developed as a wrapper around Random forest classification algorithm with a specific task of feature selection (Kursa and Rudnicki 2010). In this study, they performed hyper-tuning for parameters of LSTM by making use of Bayesian algorithm. They also used various different types of optimization methods to enhance the performance of neural networks (McNally et al. 2018). The downside of their work was overfitting. The problem of bitcoin price prediction is very similar to the problem of stock price prediction. The difference is that bitcoin market is more dynamic and uncertain in nature which makes it more complex when it comes to predicting. The authors couldn't eliminate the noise and levels present in the data. However, the optimization techniques used in the study and feature engineering part of the study are the strengths of this research.

In 2018, Weng et al proposed a short-term stock price prediction system by making use of ensemble methods. They used multiple sets of data for the study, sourced 5 datasets. They pulled the required datasets from 3 open-source APIs and a package from R named TTR. For the study, they used four machine learning approaches – Support Vector Regression Ensemble (SVRE), AdaBoost with unpruned regression trees as a base model (BRT), Random Forest with unpruned regression trees as a base model (RFR) and Neural Network Regression Ensemble (NNRE). A broad study of ensemble methods has been put forward in the aforementioned research with a prime focus on short-term stock price prediction. The author did a comprehensive review of the previous related work and precisely chose 8 technical indicators to perform evaluation on their 5 datasets. The strength of their work is that they developed a system for the investor using R programming in which user need not to feed their own data but rather APIs in place will do the job by pulling the data from the online source (Weng et al. 2018). Nevertheless, there are a few limitations to this work. The system can only be able to predict from 1 to 10 days, it won't work for investors looking for return in 2 weeks or intra-day traders which buy and sell the shares the same day. Also, this system was built only on 20 specific US stocks hence it might fail in generalizing the other market's stocks.

In this literature review, we found many methodologies relevant to our dataset, optimization techniques that can be utilized for feature engineering and various evaluation methodologies. From such a vast survey, it is evident that researchers do not focus much on preprocessing of the data and related mechanisms. Neither the researcher attempted to build such a mechanism that facilitates the preprocessing of the data. From technical domain, researchers are highly likely to concentrate on building predictive model with higher accuracy. When it comes to feature selection, a common trend has been detected where the researchers first list all the features used in the previous works, then apply the optimization algorithm and then based on the results from algorithm, they choose the best voted features. This concludes the summary of feature engineering and predictive modelling from the survey.

2.2 Stock Market Index

The stock market index is the combination of stocks and market index. A market index traces the performance of certain bonds, stocks and all other investments.

These investments are mostly grouped among different types of industries (What Is A Stock Market Index? – Forbes Advisor n.d.). Market index provides a high-level representation of portfolio of investment holdings. And similarly, the stock market index is an index that tracks the performance of certain group of stocks, that allows investors in comparison of current stock price with past prices of the stock to estimate the performance of the market.

There are various stock market indexes available out there and each index has its own way of determining which investments or companies to include in the index. Once the index managers decide the companies to include in the index, then they must decide how these selected companies are represented in the index. This process of determining how the companies is going to be represented in the index is based on a factor called index weighting. The weighting allows the companies included in the index to have an equal impact in the representation of index performance. There are 3 most common weighting models available out there – 1.) Market-Cap Weighted, 2.) Equal Weighted and 3.) Price Weighted.

In the world of finance, there exists thousands of stock market indexes. The most common indexes are listed below-

- **S&P 500 Index:** The Standard and Poor's 500, also known as the S&P 500 is the stock market index responsible for tracing the stock performance of top 500 companies listed in the index in United States of America. This is a market-cap weighted index.
- **Dow Jones Industrial Average:** It is one of the oldest stock market indexes, responsible for tracing the stock performance of top 30 companies listed in the index in United States of America. The scope of the index is quite narrow as only 30 companies are selected for the index. This is a price-weighted market index.
- **Nasdaq 100:** This stock market index is made up of 102 equity securities which are issued by 101 of the huge non-financial companies listed on the index. The most active traded stocks belong to the Nasdaq 100. There are various sectors of companies are included in the index, but it more inclined towards the technology sector. Nasdaq 500 is a market-cap weighted index.
- **NYSE Composite Index:** NYSE Composite Index keeps track of every stock traded on New York Stock Exchange (NYSE) including other assets like foreign

listings, real estate investment trusts and American depository receipts. The index is market-cap weighted index.

- **Russel 2000 Index:** It is a small-cap stock market started by a company named Frank Russel back in 1984. While the other indexes focus on top and big companies, Russel 2000 is held responsible for tracking the performance of 2000 publicly traded domestic companies. It is a market-cap weighted index.
- **Wilshire 5000 Total Market Index:** This stock market index is considered to be manager of entire U.S. stock market. It is responsible for tracking the performance of complete stock market of United States. This is a market-cap weighted index.

For this research, I have made use of the data of stocks belonging to the S&P 500 stock market index as the data was easily accessible and publicly available on various platforms on the internet in the required format.

2.3 Time Series Analysis

If we wish to determine the trend in financial markets or electricity consumption or birth rate in the following year/month/week/day or covid case estimation in the near future, the time plays crucial factor that are required to be considered in the models (Peixero 2019). For context, It would be necessary to know the demand of a particular product in advance in order to manage the inventory of the same product well. For example, it would be a great idea to forecast the peak consumption of the electricity in a day so that pricing and production can be planned well in advance to avoid interruption in supply.

A time series is series of data point collected at a regular interval of time (Australian Bureau of Statistics 2022). For example, number of confirmed covid cases registered every day, power consumption at every hour of the day, price of a stock collected at every minute, sales registered of air conditioners at every month of the year, etc.

Unlike the mainstream machine learning problems where there are several independent variables, time or datetime stamp is often the independent variable in a time-series problem and the goal here is to predict the next value in the series corresponding to the next datetime stamp.

However, there are other aspects of time series that needs to be considered in the analysis. Let's discuss the characteristics of time-series below-

2.3.1 Autocorrelation

In simpler words, autocorrelation refers to the relationship between the series values. Just the way correlation is the way determining the linear relationship between two independent variables, similarly, Autocorrelation is the way to determining the linear relationship between time-series lagged values. For example, r_1 determines the linear relationship between value y_t and the value y_{t-1} of time series, r_2 determines the relationship between y_t and y_{t-2} and this goes on. The relationship r_k can be measured as:

$$r_k = \frac{\sum_{t=k+1}^T (y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^T (y_t - \bar{y})^2}$$

Equation 1: Autocorrelation

In the equation 1 (Hyndman and Athanasopoulos 2018a), T is count of values in the time series.

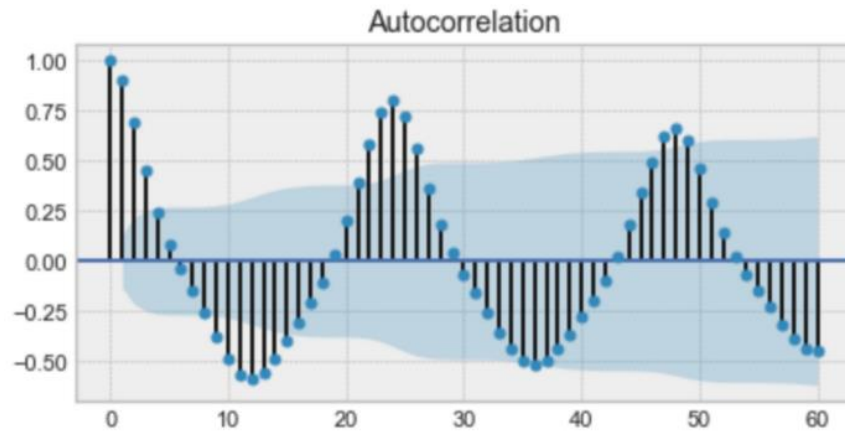


Figure 2: Example of Autocorrelation

In the figure 2, plot of autocorrelation, the very 1st value, the 24th value and the 48th value have the highest positive autocorrelation and in same way, 12th value, 36th value and 60th value have the highest negative autocorrelation. This implies that we'd find the similar value after 24 values, or the same value would be repeated after 24 values.

2.3.2 Seasonality

If the mean of the values of time-series changing at a regular interval of time, then the series is considered to be having seasonality (Kaggle 2022). For example, the electricity consumption is higher in winters than summers in Ireland this means this is the regular pattern of electricity consumption around the year.

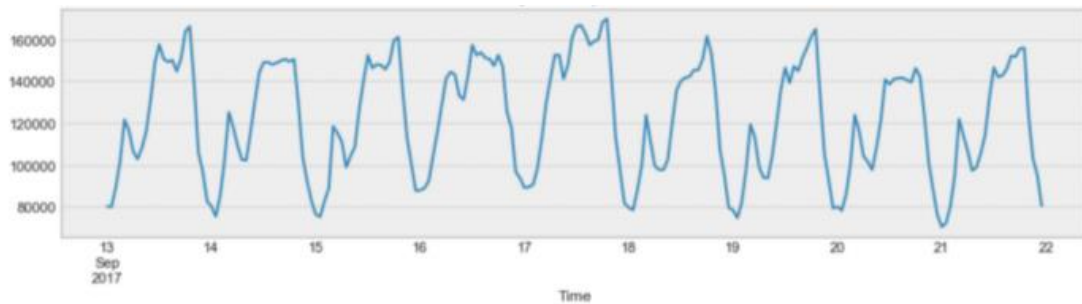


Figure 3: Example of seasonality

In the figure 3, it demonstrated the clear scenario of seasonality of every day. The peak is attained during the evening hours every day, then going down and then getting up back the next day from the morning. Interestingly, seasonality can also be determined by the autocorrelation plot. If the autocorrelation plot is having the sinusoidal shape, then the series exhibits the seasonality.

2.3.3 Stationarity

A series is said to be stationary if its statistical properties including mean, variance, standard deviation do not change over time hence the series with trends can never be stationary but the series with seasonality can be. Stationarity is an important aspect of time-series as many statistical tests and models rely on it (van Delft et al. 2021).

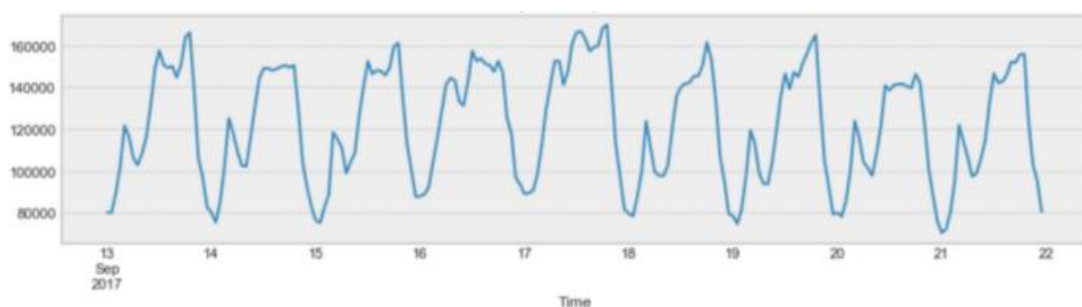


Figure 4: Example of stationarity

As I mentioned above that a series having seasonality can be stationary so let's consider the same example as I did for seasonality here. In figure 4 above, despite having seasonal effect on the series, the statistical properties like mean and variance is constant over time hence the series exhibits stationarity.

Most of the times, it not possible to determine if the series is stationary by having a look at the plot and that when a statistical test comes into play called **Augmented Dickey-Fuller** (ADF) test. Without getting into the mathematical explanation of the test, it is a hypothesis test where the null hypothesis is that series is not stationary and alternate hypothesis is that series is stationary. The implementation of the test is pretty much straightforward in python.

```
#Test for stationarity
from statsmodels.tsa.stattools import adfuller
```

Figure 5:Library for ADF test

Once the above library is successfully imported, run the following chunk of code to obtain a P-value.

```
result = adfuller(df_aapl['close'])
print(f"ADF Statistic: {result[0]}")
print(f"P-value: {result[1]}")

ADF Statistic: -0.6650906914089934
P-value: 0.855585844583895
```

Figure 6:Code to implement ADF test

Once the P-value is obtained, rest of the job is the interpretation of this P-value. If the P-value is less than 0.05 i.e., the significance level, this implies that we fail to accept null hypothesis and the series is stationary and if the P-value is greater than 0.05 then we must accept the null hypothesis which implies that the series is not stationary. In the above example, the P-value is 0.85 which is greater than the significance level, that implies that the series passed as test parameter for ADF test is not stationary.

Ideally, a time-series must be stationary in order to be fed in any algorithm for further analysis or modelling but in the real world, it is almost impossible hence certain transformations are done on the series to make it stationary and ready for further analysis (Shumway and Stoffer 2019).

2.4 Survey on stock price prediction algorithms

Without a doubt the stock market is highly volatile in nature, dynamic and non-linear hence it is almost impossible to predict the stock market accurately. The reason behind it being very challenging is that the market movement depends on many mini

and macro factors such as political factors, company's finances and performance, economic conditions, unexpected occurrences, etc. But the silver lining in stock market is publicly available data and since the nature of it being highly volatile, it produces an enormous amount of data of stock prices every day. The more the data; the more patterns can be found within the data and hence experts from financial and technical domain have always tried to predict the stock market movement based on its historical by leveraging various techniques and methodologies ranging from traditional fundamental analysis to modern era technical analysis.

2.4.1 Auto-Regressive Integrated Moving Average (ARIMA)

ARIMA is a statistical analysis model that makes use of time series to understand and determine the pattern of the series and to forecast the future trends (Contreras et al. 2003). The ARIMA methodology that is used to study the time-series is because of Box and Jenkin Method (Mustafa and Fareed 2020).

There are three pillars of ARIMA model-

- Auto Regressive (AR)
- Moving Average (MA)
- Integrated (I)

Before getting into ARIMA, there are two concepts – a) Autocorrelation, and b) Partial autocorrelation that needs to be understood. Both concepts are derived from correlation. Whenever the value of one variable increase and it causes the increase in value of other or if the value decreases of one variable and so does with other variables then both the variables are said to be having positive correlation whereas if the value of one variable increase and it causes the decrease in value of other variable or vice-versa then those two variables are said to be having negative correlation. For example, the stock price of a company, say on Friday depends on the stock price on Thursday which in turn is dependent on the price on Wednesday and so on. But the price on Friday also depends on the price on Tuesday, as provided in the figure below.

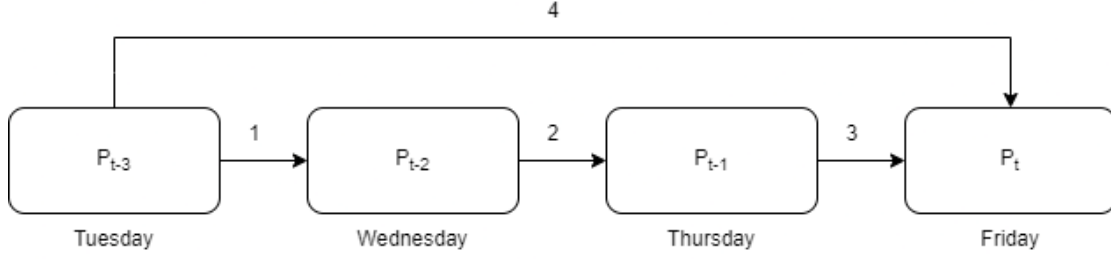


Figure 7: Example of Correlation

In the above figure, Thursday has a direct effect on Friday, Wednesday has a direct effect on Thursday and Tuesday has direct effect on Wednesday and Friday. Also, Wednesday has an indirect effect on Friday.

- **Autocorrelation** considers direct and indirect effect in a time series
- **Partial Autocorrelation** considers only direct effects in a time series.

Auto Regressive (AR): It is one-of-a-kind regression model where the value at current timestamp depends on the values at previous timestamps. It essentially means that the value at the current timestamp is correlated with values at previous timestamps. The correlation in AR model is the partial autocorrelation. The AR model equation can be given as below-

$$Y_t = \beta_1 + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \dots + \phi_p Y_{t-p}$$

Equation 2: Auto Regressive model

ϕ are the weights of the lagged observation which are decided by the correlation between the current value and lagged value. If the correlation is high, then so is the value of the weight and vice-versa. p represents the number of lags in the equation 2.

Moving Average (MA): Moving average model is all about analysing the error that occurred while predicting the future values from the past values to make a better estimation of value at current timestamp. The models deal with the errors from the previous observations. The impact of these error from lagged observations is dependent on the autocorrelation between them. The MA equation can be given as below-

$$Y_t = \beta_2 + \omega_1 \varepsilon_{t-1} + \omega_2 \varepsilon_{t-2} + \dots + \omega_q \varepsilon_{t-q} + \varepsilon_t$$

Equation 3: Moving Average model

Term ϵ are the error observed at a specific lagged value and ω are the weighted that depends on the correlation. q denotes the size of the moving average window that is how many previous values are to include to calculate the average at the current timestamp.

MA model enhances the performance of AR model by taking the error into account resulting in making the prediction's accuracy better. But both AR and MA models are only successful as long as the series of regular timestamp is stationary. They won't hold good if the series isn't stationary hence the stationarity is pre-requisite for AR and MA models.

Integrated (I): In the real-world data, it is next to impossible to encounter a stationary series and that's where the sub-acronym – I in ARIMA comes into play. Integrated determines the differencing order required in the series to make it stationary. Differencing is basically the difference between current and previous value in the series. Differencing order is number of times one need to carry out differencing to make the series stationary. The formula of the differencing can be given as: $Z_t = Y_t - Y_{t-1}$. Integrated has nothing to do with integration in ARIMA. Instead of predicting the series, Integrated (I) factors predicts the difference between two consecutive series values.

Combination of all three segments discussed above makes up the ARIMA model. To sum it up, ARIMA takes three parameters (p, d, q) for implementation where p is the number of previous observations or lag order, d represents the order of differencing and q is the window size of the moving average.

2.4.2 Linear Regression

In linear regression with respect to time series, the basic concept is dependent time series (forecast variable) and independent time series (predictor variable) have a linear relationship with each other (Chapter 5 Time series regression models | Forecasting: Principles and Practice (2nd ed) n.d.). For example, one might need to predict the demand for electricity y which depends on temperature of the day x_1 and the day of the week x_2 . In this scenario, the forecast variable electricity demand would have the linear relationship with predictors or independent variable i.e., day of the week and temperature of the day. Broadly, there are only two types of linear

model: 1) Simple Linear regression and 2) Multiple Linear Regression (5.1 The linear model | Forecasting: Principles and Practice (2nd ed) n.d.).

Simple Linear Regression: In simple linear regression, there is only single predictor variable having a linear relationship with the forecast variable. The equation of the simple linear regression can be given as below:

$$y_t = \beta_0 + \beta_1 x_1 + \varepsilon_t$$

Equation 4: Simple linear regression

In the equation 4, the coefficient β_0 represents the intercept of the time and the coefficient β_1 represent the slope of the line. β_0 is the predicted value when the input is null of the predictor variable and the slope here signifies the average predicted change in the forecast variable resulting from one unit increase in the predictor variable.

Multiple Linear Regression: In multiple linear regression, there are two or more independent or predictor variables having the linear relationship with the forecast variable. The equation of the multiple linear regression can be given as below:

$$y_t = \beta_0 + \beta_1 x_{1,t} + \beta_2 x_{2,t} + \cdots + \beta_k x_{k,t} + \varepsilon_t$$

Equation 5: Multiple linear regression

In the equation 5, $x_1, x_2, \dots x_k$ are the predictor variables. The coefficients β measure the weight of each predictor after having calculated the impact of other predictors on the forecast variable. y is forecast variable in the equation.

2.4.3 Long Short-Term Memory (LSTM)

In the field of deep learning, time-series data cannot be handled with conventional feed-forward networks for training or predicting purposes. A new mechanism was required in place that's capable of holding the historical data and make predictions based on the historical data. Recurrent Neural Network (RNN) is a type of traditional feed-forward artificial neural network that is capable of holding the past information for making predictions and is well suited for handling time-series data (An Introduction To Recurrent Neural Networks And The Math That Powers Them n.d.). RNN is one of a kind artificial neural network specially developed to handle the sequential data or time-series. Traditional feed forward approach in neural networks only deals with the data points which are independent from each other, but the

approach would fail with the sequential data where one data point depends on the previous data point. To arrest such a problem, the neural network was modified in order to incorporate the correlation between the data points. RNNs has a component called ‘memory’ that allows the network to store previous state and information to produce the next series of output.

Long short-term memory (LSTM) is one of the architectures of RNN just like its other architectures- Bidirectional recurrent neural network (BRNN), Gated Recurrent Units (GRU). LSTM was originally developed to handle the problem of vanishing gradient in RNN hence the LSTM is called as the advanced version of RNN (Saxena 2021). For example, while reading a story, we always remember what happened in the last page and that makes us understand the context of the current page, LSTM works in the similar manner, it remembers the information from previous state and use that information in producing the output at the current state. LSTMs are complex, it is quite difficult to understand them in the area of deep learning (Hochreiter and Schmidhuber 1997)

At a very high level, LSTM network consist of three gates – 1) Forget Gate, 2) Input Gate, and 3) Output Gate as shown in the figure below-

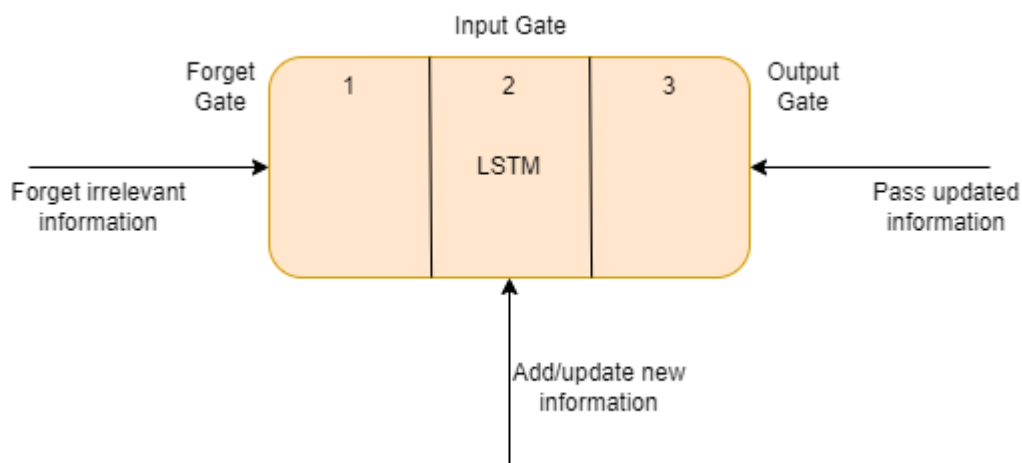


Figure 8: LSTM Cell

The first gate is called the forget cell which determines whether the information coming from previous timestamp is relevant and have to be remembered or irrelevant and can be forgotten. The second gate is the input gate which the cell attempts to learn new information from the input. The third gate is called the output gate in which previous cell pass along the updated information of the current timestamp to the next

one. All the three gates constitute one LSTM cell. All three aforementioned gates have been described in detail in the following subsections. Interestingly, an LSTM cell carries all the information along with the corresponding timestamps.

For example, ‘John is a good man. Jane is an evil person.’ Here I have two sentences separated by a full stop (.). The first sentence states that ‘John is a good man’ and the second states that ‘Jane is an evil person.’ It is evident that I am talking about John in first sentence and as soon as I encountered with the full stop, I started talking about Jane. As one move along from first sentence to the other, the LSTM network should be able to recognize that the subject has been changed from John to Jane and the current subject is Jane. In LSTM, forget gate allows the cell to forget about the John.

2.4.3.1 Forget Gate

The very first step in an LSTM is to decide whether the remember the information or to forget the information from the last timestamp. The equation for the forget gate can be given as –

$$f_t = \sigma (X_t * U_f + H_{t-1} * W_f)$$

Equation 6: Forget gate

In the equation 6, X_t is the input to the current timestamp, U_f signifies the weightage of the input, H_{t-1} is the hidden state from last timestamp and lastly, W_f is the weight matrix that is associated with the hidden state. A sigma function is applied over it which gives the value of f_t anything between 0 to 1. This value of f_t is then multiplied by cell state of the last timestamp and then if the value comes out to be 0, this implies that the network will forget everything or if the value comes out to be 1 then it cannot forget anything. Returning to our example, ideally the network should forget the John as soon as it encounters the full stop.

2.4.3.2 Input Gate

Let me take another example here, ‘John knows how to shoot. He told me that he has served in the army for four years.’ In this example, John is the subject in both the sentences as both are providing information about same person. However, both of the sentences give different information about John. Now if we give it thought, based on the information provided in the first sentence, what would be the important information from the second one. In this example, the fact that he has served in the

army is critical and is something I'd want my model to remember and that is exactly the job of input gate in LSTM.

The Input gate is used for extracting the significant information supplied by the input. The equation of the Input gate can be given as –

$$i_t = \sigma (X_t * U_i + H_{t-1} * W_i)$$

Equation 7: Input gate

In the equation 7, X_t is the input to the current timestamp, U_i is the weight matrix of the input, H_{t-1} is the hidden state from last timestamp and lastly, W_i is the weight matrix that is associated with the hidden state. Once again, a sigma function is applied all over it and the value of i_t comes out to be anything between 0 and 1. The equation of a new information can be given as –

$$N_t = \tanh(X_t * U_c + H_{t-1} * W_c)$$

Equation 8: New information

Now, the new information that is required to be included in the cell state is a function of the hidden state at the last timestamp (t-1) and input (x) at current timestamp (t). The function used for activation is tanh. Due to the activation function used here, N_t will end up between -1 and 1. Now if the value of new information comes out to be negative then the information has to be subtracted from the cell state at current timestamp and if the value of new information comes out to be positive then the new information has to be added to the cell state at current timestamp.

2.4.3.3 Output Gate

Let us consider another example here, “John was a proud soldier and died fighting for his nation. For his priceless contributions, brave ____.” In this example, we are expected to complete the second sentence. Through the visual scan when we read ‘brave’, we know for sure that it is a quality of the person. From the first sentence, we know that our subject name is John who also happens to be a person. Based on the provided input, we are supposed to fill the blank word. The output word I am talking about is the aim of the output gate. The equation of the output gate can be given as –

$$o_t = \sigma (X_t * U_o + H_{t-1} * W_o)$$

Equation 9: Output gate

Each term in the equation 9 remains the same as it explained in the above sections of forget gate and input gate. The value of o_t comes out to be between 0 and 1 due to the activation of the sigma function. The next step would be to calculate the hidden state. To do that we'd need o_t and \tanh of the updated cell state. The equation of the same is given below –

$$H_t = o_t * \tanh(C_t)$$

Equation 10: Hidden state

In the above equation, C_t signifies the long-term memory and o_t is the current output. Here the hidden state is the function of current output and memory. Now the final step is to evaluate the output at the current timestamp which is done by applying the SoftMax activation function on the hidden state (H_t).

$$Output = SoftMax(H_t)$$

Equation 11: SoftMax activation function

In the equation 11, the token with highest score in the output would be the prediction.

3. Data and Methods

3.1 Data Source and Description

The data is sourced from an open-source platform- Kaggle and requires no permission to use or manipulate the data for any purpose. I have taken the data of all the stocks that belongs to a US stock market index – Standard & Poor 500 (S&P 500) starting from 8th February 2013 to 7th February 2018. The data consists of a few features as explained below in detail.

S.No.	Feature	Data Type	Description
1	Date	Date	The date of the day
2	Open	Float	The opening price of the stock for the day
3	High	Float	The highest price of the stock for the day
4	Low	Float	The lowest price of the stock for the day
5	Close	Float	The closing price of the stock for the day
6	Volume	Float	The number of shares traded in the day
7	Name	Object	Ticker name of the stock

Table 1: Data Description

The overall data contains 619040 records including the stock prices of all the companies listed in the stock market index.

At the very first glance of the data, it contains some of the missing values in open, high and low hence those values need to be imputed. There is no way possible that we can have those records removed as we are dealing with a time series problem here and records of each date of the entire date range is mandatory in the dataset.

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
#   Column  Non-Null Count  Dtype
---  -
0   date    619040 non-null    object
1   open    619029 non-null    float64
2   high    619032 non-null    float64
3   low     619032 non-null    float64
4   close   619040 non-null    float64
5   volume  619040 non-null    int64
6   Name    619040 non-null    object
dtypes: float64(4), int64(1), object(2)
memory usage: 33.1+ MB

```

Figure 9:Data Information

In my preliminary analysis, I was under the impression that my data points are irregular. In time-series analysis and forecasting, each data point must be captured at a regular frequency. In my project, I am dealing with stock prices of S&P 500 companies. One thing needed to be kept in mind that stock markets do not work weekends and national holidays and hence do not produce the data all year round. Therefore, one cannot expect 365 values per company per year from the data. The data points are going to be present only for trading days which are actual working days.

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

Figure 10:First 5 records of Dataset

In the figure 10, the very first date is 8th February 2013, and the second date is 11th February 2013. It appears that data is missing for 9th and 10th of February. However, these dates coincide with the weekend when stock market is closed. These missing dates does make the date timestamps irregular, but it does not make the data points irregular. A few options were explored in attempting to rectify this situation –

- First option was interpolation i.e., fill in those missing dates manually and then imputation of corresponding value by the mean of the series. I chose to discard

this strategy as it would be tempering with the original data and disrupt the actual series.

- Second option was to remove all the dates and replace with serial numbers starting from 1. This would keep the series intact but lose the date time factor in the series, which is highly significant for exploration, visualisation and lastly, prediction. I would not be able to tell the date of the predicted stock price hence I discarded this option as well.
- Lastly, to keep the data as it is without doing any change with the real datetime stamps and corresponding series. This would not deteriorate the quality of the data and I would be able to predict the stock price along with the corresponding dates hence, I decided to go ahead with it.

3.2 Data Preparation

The dataset contains the stock price of all the companies listed in the stock market index i.e., S&P 500 in my case. It is not possible to work on the stock prices of all the companies at the same time or even simultaneously hence I chose Apple Inc stock (Ticker name: AAPL) for exploration, experimentation, and modelling as well. Later I chose top 50 companies stocks for generalised result of the project. At start I chose Apple stocks data as shown below-

```
df_aapl = df[df['Name']=='AAPL']
```

```
df_aapl.head()
```

	date	open	high	low	close	volume	Name
1259	2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
1260	2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
1261	2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
1262	2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL
1263	2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL

Figure 11: First 5 rows of Apple Stocks data

Whenever we are dealing with time series, one should make sure if the data type of the dates is accurate and I made sure of that in my project as demonstrated in the code snippet below.

```
df['date'] = pd.to_datetime(df["date"], format="%Y/%m/%d")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 619040 entries, 0 to 619039
Data columns (total 7 columns):
#   Column      Non-Null Count  Dtype
---  ---
0   date        619040 non-null  datetime64[ns]
1   open        619029 non-null  float64
2   high        619032 non-null  float64
3   low         619032 non-null  float64
4   close       619040 non-null  float64
5   volume      619040 non-null  int64
6   Name        619040 non-null  object
dtypes: datetime64[ns](1), float64(4), int64(1), object(1)
memory usage: 33.1+ MB
```

Figure 12: Code snippet to update datatype of column date

Once I made sure that all the data types are correct, I changed the index of the dataframe with dates column. Index of the dataframe defines the sequence of data points and since I am dealing with the time series data, it is a sequential data with respect to its dates and hence this is needed to be done when dealing with time series.

```
df_aapl.head()
```

	open	high	low	close	volume	Name
date						
2013-02-08	67.7142	68.4014	66.8928	67.8542	158168416	AAPL
2013-02-11	68.0714	69.2771	67.6071	68.5614	129029425	AAPL
2013-02-12	68.5014	68.9114	66.8205	66.8428	151829363	AAPL
2013-02-13	66.7442	67.6628	66.1742	66.7156	118721995	AAPL
2013-02-14	66.3599	67.3771	66.2885	66.6556	88809154	AAPL

Figure 13: First 5 rows of Apple stock prices with dates as index

After all the steps mentioned above, I plotted the Apple stock prices to have a visual look at the data and also to derive the insights from visualization.



Figure 14:Plot of Apple Stock prices

Interpretation of Apple Stock prices	<ol style="list-style-type: none"> 1.) There's a significant decline in stock prices is visible between 2015 and mid of 2016 but overall trend of stock prices is positive. 2.) The volume of stocks traded has been decreased with time. Possibly due to high prices of the stocks.
--	--

There are many time series available in the dataset that includes opening price, high price, low price, and closing price but I chose only one of them for the project as the ARIMA model which happens to be base model in this study, does not support multivariate timeseries. Dealing with a single time series and performing analysis on the same is called Univariate Timeseries (Newbold and Granger 1974). When dealing with the stock prices, closing price of day is the most significant factor due many reasons hence I chose the closing price of stocks from the dataset for my analysis.

3.3 Data Exploration

Let us visualize the stock price (Close) fluctuations of a few stocks available in our dataset below-

1. AAL (American Airline Group)



Figure 15: Closing price of American Airline Group

2. CERN (Cerner Corp.)

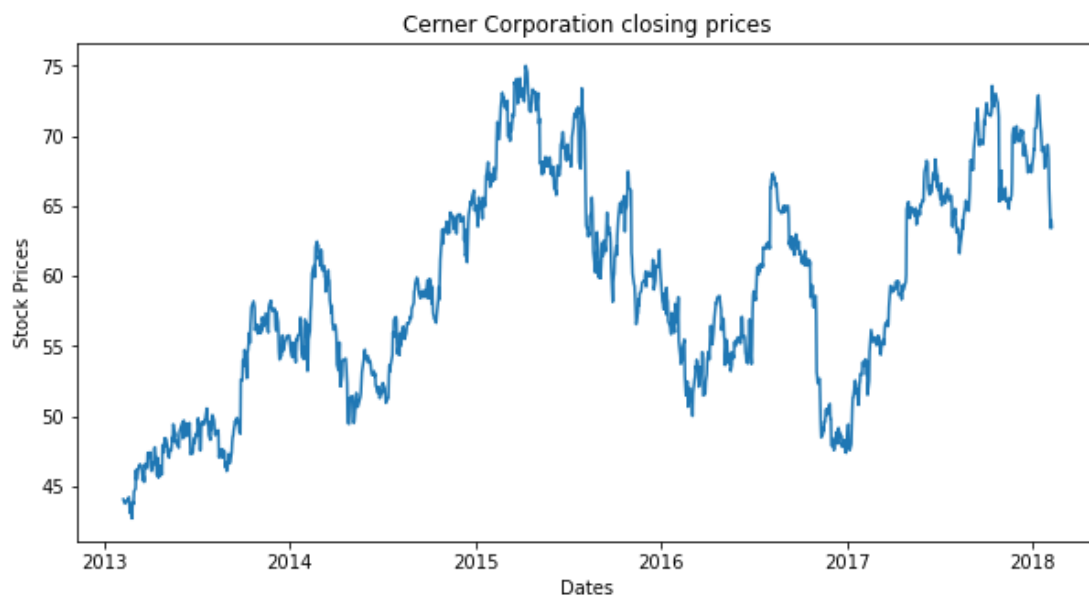


Figure 16: Closing price of Cerner Corporation

3. GOOGL (Alphabet Inc. Class A)

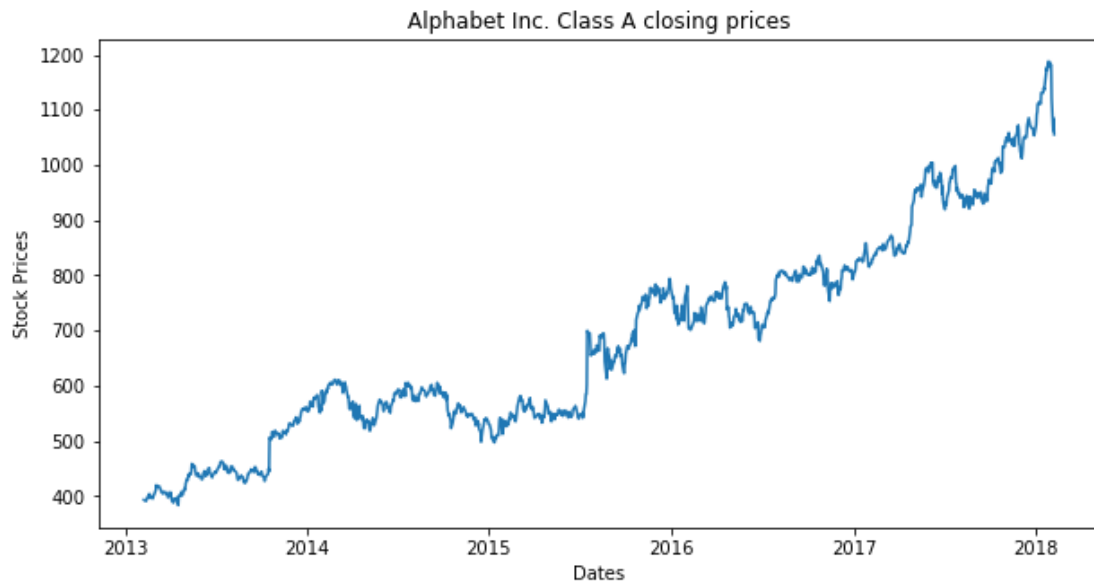


Figure 17: Closing price of Alphabet Inc. (Google)

Through visualisation of all these stocks, it is evident that these stock prices are developing a trend. I had to perform deeper analysis to confirm the presence of seasonality and white noise. Although, through the visualisation itself, it is safe to say that mean and variance is not constant in these series, yet I conducted the required test to confirm the stationarity in the series. A series without stationarity is not fit for my base model, ARIMA. I further did the needful in order to make the series stationary so that the ARIMA model can be implemented.

All the methodologies used during the exploratory data analysis are discussed in the following sub-section.

3.3.1 Resampling

Resampling is an important technique in time-series that allows us to redefine the scope of the data. One can play with the frequency of time-series with resampling. It can be increased from day-to-day data to hourly data points, this is called up-sampling. The frequency can be decreased like from day-to-day to weekly or monthly or quarterly data points, this is called down-sampling (Ho 2021). In up-sampling, you'd have to increase the data points by forging techniques, this decreases the quality and integrity of real data, hence it is never a good practice in data analysis (Ho 2021). I performed down sampling with my data, by choosing the frequency first (yearly, monthly, quarterly) and then choosing the aggregation function performed

on corresponding values (mean, min, max). The reason for performing resampling is to know how the data would react under a different resolution.

I have chosen Apple Inc stocks for all the exploration and experimentation in my project and have performed down sampling on it with 4 different resolutions –

1. Yearly with min and max as aggregation.
2. Quarterly with min and max as aggregation.
3. Monthly with min and max as aggregation.

3.3.2 Smoothing (MA)

In this section, I introduced the moving average smoothing that is a naïve and one of the most effective techniques in time-series for exploration as well as forecasting. Smoothing is used to eliminate the seasonality and fine-grained highs and lows between time steps (Brownlee 2020). Smoothing is another time-series that is built by calculating the average of real time-series of certain window size. Window size refers to the count of values of original series taken to calculate the average in smoothing series. There are two types widely used smoothing techniques –

1. Centred Moving Average: The value of new series at timestamp t with window size of 3 will be calculated as $\text{mean}(\text{obs}(t-1), \text{obs}(t), \text{obs}(t+1))$.
2. Trailing Moving Average: The value of new series at timestamp t with window size of 3 will be calculated as $\text{mean}(\text{obs}(t-2), \text{obs}(t-1), \text{obs}(t))$.

In the project, I have performed the trailing moving average to obtain a smoothened series with window sizes of 5, 10, 15, 20, and 30.

3.3.3 Exponential Weighted Moving Average (EWMA)

Exponentially weighted moving average is a statistical measure used to describe the time-series. Unlike the normal moving average, EWMA provides more weight to the recent data and lesser weight to old data. The weights fall exponentially as the data points gets old. EWMA is meant for capturing the recent trend unlike simple moving average (SMA) which provides a generic trend (Krishnan 2021). The formula of EWMA can be given as –

$$EWMA_t = \alpha * r_t + (1 - \alpha) * EWMA_{t-1}$$

Equation 12: Exponential Weighted Moving Average

In the equation 13, α signifies the weight and r_t is the value in the series at timestamp t . EWMA is widely accepted practice in financial world for the technical analysis and modelling of volatility (CFI 2022).

In the project, I have calculated EWMA of span 20 and 30 with varying the value of α with 0.2, 0.6 and 0.9.

3.3.4 Test for stationarity

Before implementation of the first model of the project that is ARIMA, there are some prerequisites that need to be present for ARIMA and the property of stationary comes in that list of prerequisites. ARIMA is not designed to handle non-stationary series hence the test was conducted to check if the Apple stock prices are stationary or not. The test conducted is called Augmented Dickey Fuller test as explained in section 2.3.3. The inbuilt test function is available in the stats model library. Once the required library was imported, directly series values were fed into the function. The output of the function were two values – 1.) ADF Statistic and 2.) P-value. I was only concerned about P-value as that gives enough information about the stationarity of the series.

3.4 Data Pre-processing

In the project, I have implemented over all 3 models – 1.) Time series ARIMA model, 2.) Machine Learning Linear Regression, and 3.) Deep Learning LSTM Neural Network. All of three have different requirements of data before the actual implementation.

3.4.1 For ARIMA

Following the determination of ARIMA parameters i.e., p , d , q values, I simply ran the model over the entire series without splitting the data. Now, if I had predicted on the same data, there would be very high chances of over-fitting as we passed the same data for training as well and it doesn't make much sense to test the prediction on the same training set. In that case, I confirmed my parameters through Auto ARIMA model and then I split my series in training and testing segment. I chose the 80% of series for training and rest of 20% for testing as shown below in the code snippet.

```
#Pre-processing of data - Train and Test split  
  
n = int(len(df_aapl) * 0.8)  
train = df_aapl['close'][:n]  
test = df_aapl['close'][n:]
```

Figure 18:Code snippet of splitting the data for ARIMA

After the code from figure 18, I had the confirmed parameters from Auto ARIMA module, training series to train the ARIMA model and testing series to check performance of the model.

3.4.2 For Linear Regression

Data preparation for linear regression is pretty much straight forward when dealing with the normal dataset. But when we are dealing with time-series data, the very task is feature engineering using lags. My assumption with stock prices is that the price at current timestamp depends on trading days of an entire before the current timestamp and I know that there are only 5 trading days in a week hence lag value is 5 for feature engineering. This means that we have values of 5 previous days as independent variables or features, and the actual series is dependent variable. I created a function for lag creation of any series so that it can be used again later.

```
def lagit(df, lags):  
    names = list()  
    for i in range(1,lags+1):  
        df['lag_'+str(i)] = df['close'].shift(i)  
        names.append('lag_'+str(i))  
    df.dropna(inplace=True)  
    return names
```

Figure 19:Function for lag creation

I passed my series and lag value 5 from my assumption through the above defined function and it created the lags as required in my existing dataframe as shown below-

	close	lag_1	lag_2	lag_3	lag_4	lag_5
date						
2013-02-15	65.7371	66.6556	66.7156	66.8428	68.5614	67.8542
2013-02-19	65.7128	65.7371	66.6556	66.7156	66.8428	68.5614
2013-02-20	64.1214	65.7128	65.7371	66.6556	66.7156	66.8428
2013-02-21	63.7228	64.1214	65.7128	65.7371	66.6556	66.7156
2013-02-22	64.4014	63.7228	64.1214	65.7128	65.7371	66.6556

Figure 20: Series and its lags in a dataframe

Now that I have my features and predicting variable, I split my data for training and testing. This time, I used in-built function for splitting from Sklearn library. Again, I kept the 80% of data for my training and 20% of my data for testing as shown in the code snippet below-

```
#Preprocessing Data
x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size=0.2, shuffle=False)
```

Figure 21: Creation of training and testing data for linear regression

3.4.3 For LSTM

Unlike ARIMA model and Linear Regression model, LSTM requires a lot of pre-processing before feeding the data into the model. Let us discuss the same in steps.

1. The first step is to calculate the percent change of closing price. The reason for using the percent change instead of the actual prices is the benefit of normalization as we can measure all the variables in a comparable metric. Apart from that, returns have more statistical properties than actual prices like stationarity, as in most of the cases the prices cannot be stationary, but we can have stationary returns (Kenton 2022). It is an in-built function of pandas library.
2. Calculate the log of returns created in step 1. The reason for taking log is normalization (Quantivity 2011). This is one of the prerequisite requirements for deep learning methodology.

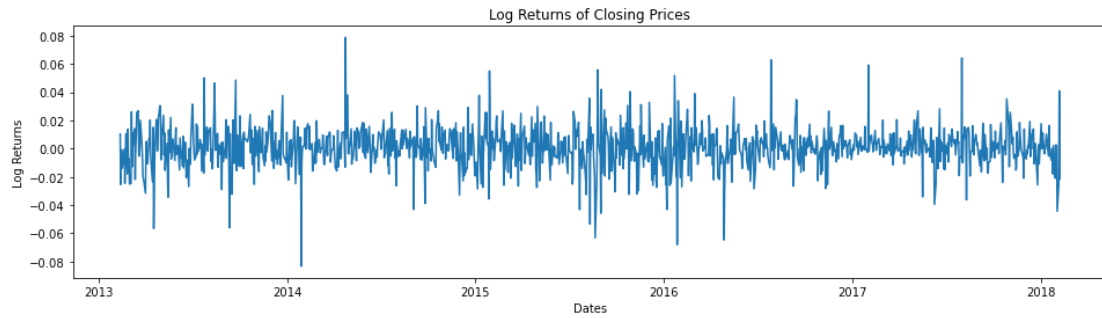


Figure 22: Log returns of closing prices

3. Defining the input variables having values of actual closing prices and log returns of the same.
4. Scaling the values of input variable created in last step between 0 to 1. This was done using MinMax scaler module of Sklearn.
5. Defining output variable having the values of scaled closing price. (Scaling has already been done in last step)
6. Splitting the input value and output value for training and testing. Again, the ratio would be kept the same i.e., 80% for training and 20% for testing.
7. Now the next step is labelling. My goal is to predict the stock price at future ($t+1$) relative to the stock price at t . We know that LSTM has its memory, and it is maintained by setting time step. Time step is basically how many previous steps I want my LSTM network to remember. In this step, I have chosen the time step of 3 and created the output variable at $t+1$. I have again created my training and testing datasets using the variables created in last step. Here, first 3 elements of last training set created in step 6 will be treated first element of new training set and the corresponding output will be the 4th element of the output variable created in step 6 and so on, as shown in the code snippet below-

```
n = 3
Xtrain = list()
Ytrain = list()
Xtest = list()
Ytest = list()
for i in range(n, len(X_train)):
    Xtrain.append(X_train[i-n:i, :X_train.shape[1]])
    Ytrain.append(Y_train[i])
for i in range(n, len(X_test)):
    Xtest.append(X_test[i-n:i, :X_test.shape[1]])
    Ytest.append(Y_test[i])
```

Figure 23: Labelling process for LSTM

8. Last step is reshaping the test and train dataset in the required form created in previous step and then the data is ready to be used modelling.

3.5 Core Technologies and Libraries

My attempt at this project was to make use of every bit I had learned during the tenure of my masters and learned a few more technologies outside the course which were relevant for the project.

Language	Python
Editor	Jupyter Notebook and Spider
Libraries	Pandas, Numpy, Math, Pmdarima, Statsmodels, Sklearn, Warnings, Keras.

Table 2: Core technologies used in project

3.6 Ethical Considerations

There could always be thousands of implications of ethical misconduct. I, as a student of Ethical studies made sure that I don't do anything that would do harm my reputation, my professors and university reputation when I signed up for this project. I made sure of using the publicly available data under the license CC0 1.0 Public domain dedication. This license allows any person to modify, copy, distribute and can perform their own work, even for commercial purposes. However, I have no such intention to put this project on commercial line and is strictly done for study purpose only. Stock market is complex entity whose movement depends upon several mini and macro social, economic, political factors and a lot of confidential and sensitive information available to the company board only. In this project, I attempted to estimate the future stock price-based on only the pattern formed by the historical stock prices. No other information, be it sensitive or publicly available has been used for the project. Since I am not incorporating all the factors responsible for driving stock market hence it would be unethical for me to go for commercial deployment of the project.

Before beginning with the project, I had submitted an application to DKIT ethics committee asking for the ethical clearance for the project. The application contained the aim of project, research question, the dataset to be used for project and a few relevant questions related to scope and effect of the project on humanity. I had

obtained the clearance from the ethics committee and the same can be found in the appendix section.

4. Results and Discussions

4.1 Insights from Data Exploration

As I mentioned earlier as well, I had chosen Apple Inc. stocks for experimentation and exploration and hence all the following results are from Apple stock prices.

I started the exploration with resampling to know more about the underlying data. At first, I set the frequency as yearly and the same has been plotted below.



Figure 24: Yearly minimum and maximum stock price

Interpretation of Figure 24	The minimum and maximum prices per year follows the same pattern. The prices increase dramatically up from 2017.
-----------------------------	--

Next, I set the frequency at quarterly. The same has been plotted below.

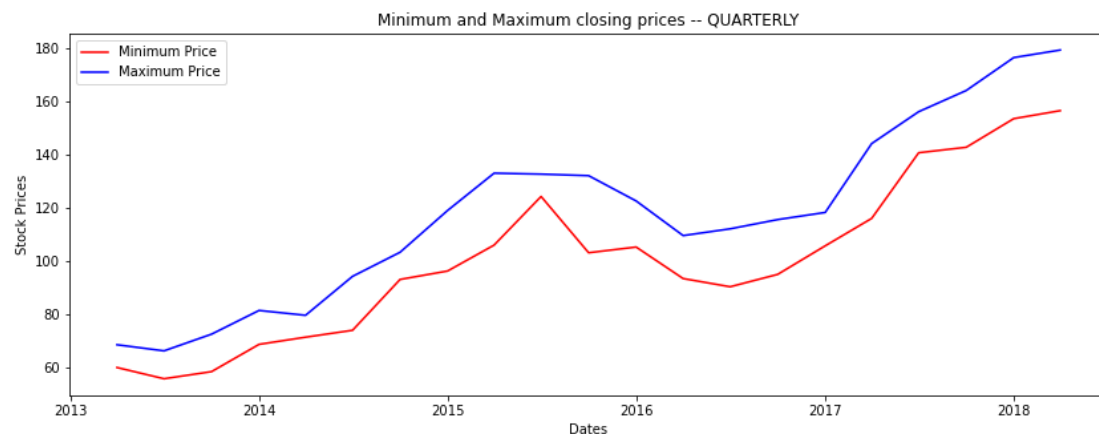


Figure 25: Quarterly minimum and maximum stock price

Interpretation of Figure 25	Prices started to drop from third quarter of 2015 to second quarter of 2016 and then the rise in prices can be seen.
-----------------------------	--

And lastly, I set the frequency at monthly. The same has been plotted below-

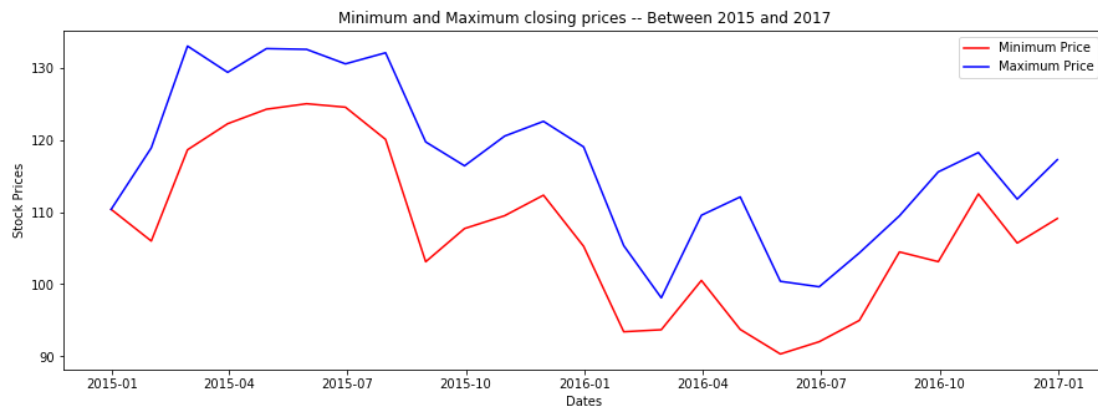


Figure 26: Monthly minimum and maximum stock price

Interpretation of Figure 26	Prices precisely started to go down in 5th month of 2015 and then started to rise back up from 6th month of 2016.
-----------------------------	---

As I narrowed down the frequency, the more accurate results start to surface. So far, I can say about my data that the trend is present including a negative trend from the mid of 2015 to the mid of 2016 but overall is positive of the entire series. Also, there are no signs of any similar pattern at a regular frequency hence no seasonality present in the dataset.

Next in exploration, I moved to the smoothing (SMA) of the series. I tried multiple window sizes to see the effect on the series. The same has been plotted below-

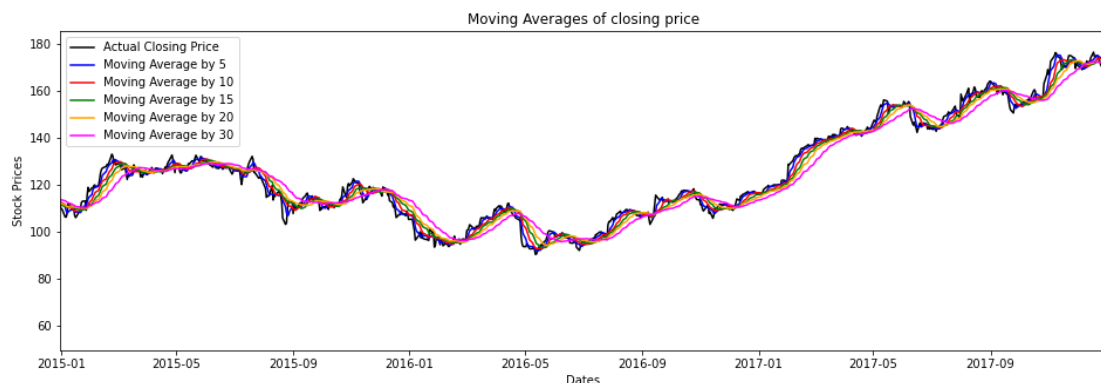


Figure 27: Smoothing by Simple Moving Average

Interpretation of Figure 27	As we increase the window size of the rolling function, the prizes are getting more smoothened out along the duration.
-----------------------------	--

Curiously next I applied exponential weighted moving average to see if it affects the smoothening of the series the same way the SMA did.

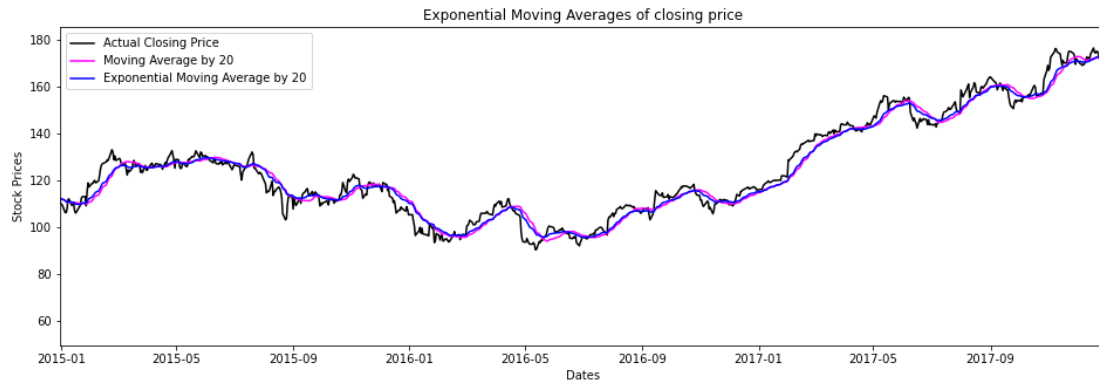


Figure 28: SMA vs EWMA (Window size:20)

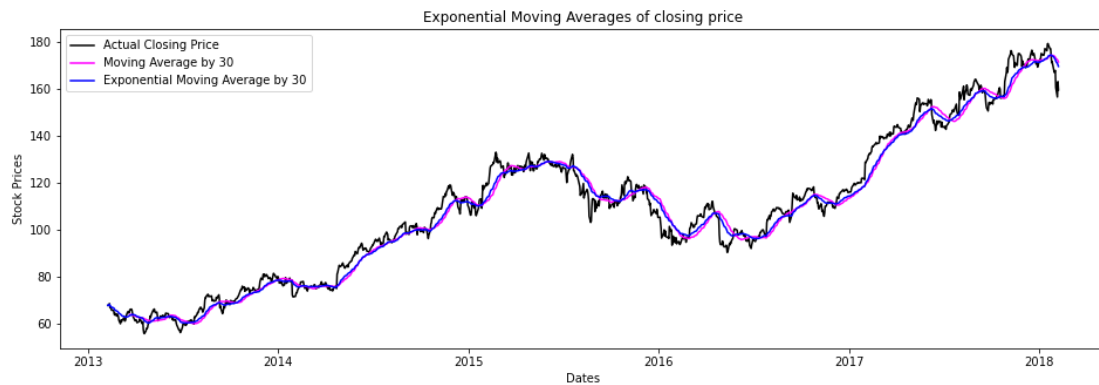


Figure 29: SMA vs EWMA (Window size:30)

Interpretation
of Figure 28
and Figure
29

Apparently, EWMA did not have a different impact on the series as expected. SMA plot and EWMA plot of same window size seems to be overlapping.

With EWMA, I also explored how the value of alpha affects the smoothening of the series.

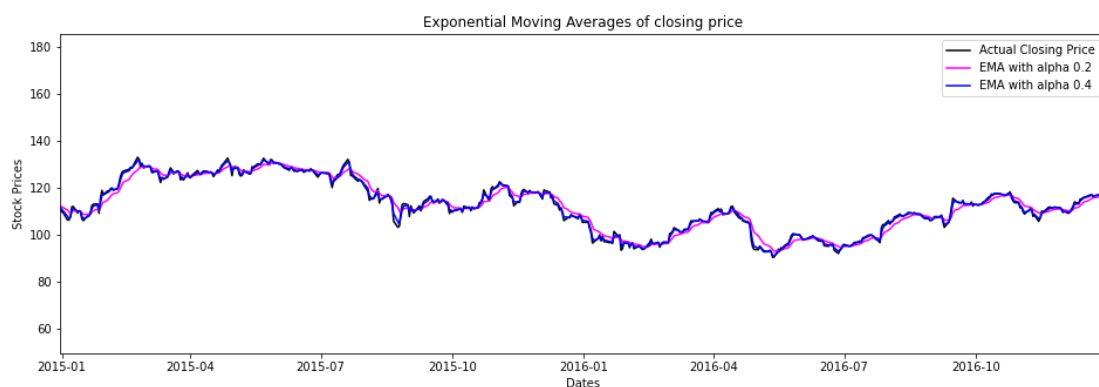


Figure 30:EWMA Plot with different alpha

Interpretation
of Figure 30

The lesser the alpha is, the smoother the series gets. As the value of alpha increases, it takes almost the path of original data.

4.1.1 Why Closing price?

The closing price of the stock market gives important evidence about the generalized trend of the market and also tells the tone of investors. It gives us an idea about the attitude of the big investors towards stock who put a large sum of money into the market for the purpose of managing their assets. Big investors mostly utilize the last few trading hours to position themselves. It is not new that big changes in the prices are witnessed at the end of the day and that movement only takes place because of big capital placements by the big players. They do so in last couple of trading hours so that they can increase buying capacity or selling (Simple Stock Trading 2022). This was more like a financial explanation and could be difficult to grasp for someone who does not understand stocks and market index. From the project point of view, the reason for choosing the closing price is the trend determination. I am dealing with the daily data and as a trader I would always be interested in the daily trend of stock price. Closing price is the last of day before the market closes, if I, as a trader were able to predict the closing price of the day, I would know it would be a great idea to invest or short position or hold. For example, I bought a stock at 20 Euros and my algorithm has predicted that closing price of the stock on that particular day is 50 Euro, this information tells me that the trend is expected to be positive for that particular day and it would be a good idea to hold the money in market and wait for the market to move up. As soon as the price reaches somewhere near to predicted value then the position can be shorted to make profit.

4.2 Models Implementation and Evaluation

4.2.1 ARIMA (Base Model)

To start with the ARIMA modelling, the very first step was to check for stationarity of the series. The series has to be stationary in order to go through the ARIMA model. I performed the Augmented Dickey Fuller test. Following were my null and alternative hypotheses before conducting the test-

H0: The series is non-stationary.

H1: The series is stationary.

```
result = adfuller(df_aapl['close'])
print(f"ADF Statistic: {result[0]}")
print(f"P-value: {result[1]}")
```

ADF Statistic: -0.6650906914089934
P-value: 0.855585844583895

Figure 31: ADF test on closing price

The P-value is 0.8555. Since the P-value is much larger than significance value, I cannot reject null hypothesis. This implies that my series is not stationary, and I need perform differencing to a certain degree in order to make the series stationary.

The next step was to perform the differencing to make the mean constant of the series but before that I plotted the Auto correlation function on the closing prices.

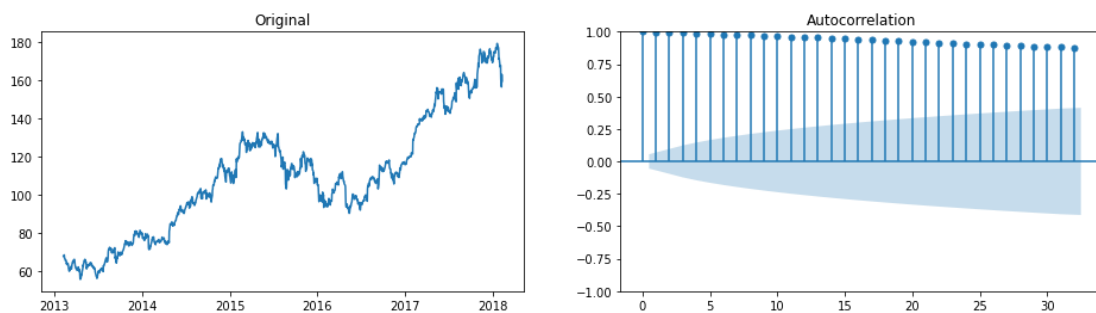


Figure 32: ACF plot of closing prices

I performed the differencing once and twice on closing price and plotted the ACF plot again to see if it did the job.

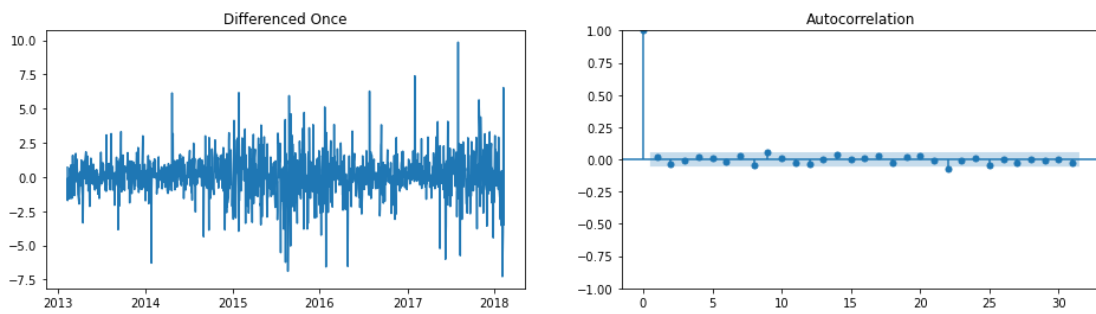


Figure 33: ACF plot of differenced (once) closing prices

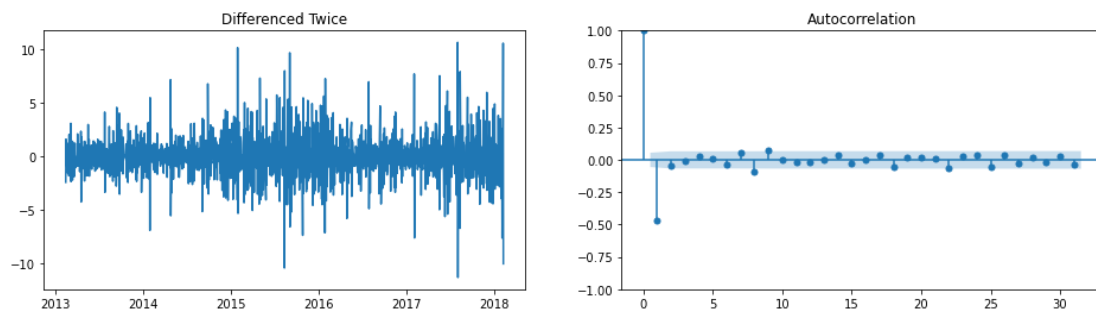


Figure 34: ACF plot of differenced (twice) closing prices

According to me, differencing once did the required job of making the series stationary and differencing twice makes it more negatively differenced. Nevertheless, I confirmed my result using `ndiffs` module from `Pmdarima` library and confirmed my results as shown below-

```
from pmdarima.arima.utils import ndiffs

ndiffs(df_aapl['close'], test="adf")

1
```

Figure 35: Code snippet to implement `ndiffs` function

This exercise gave me the value of d as on the parameter of ARIMA i.e., p, d, q .

Now that my series is finalised from above test i.e., differenced once, next job was to find the optimum value of p using that series. In order to do so, I plotted the PACF on the differenced series as shown below-

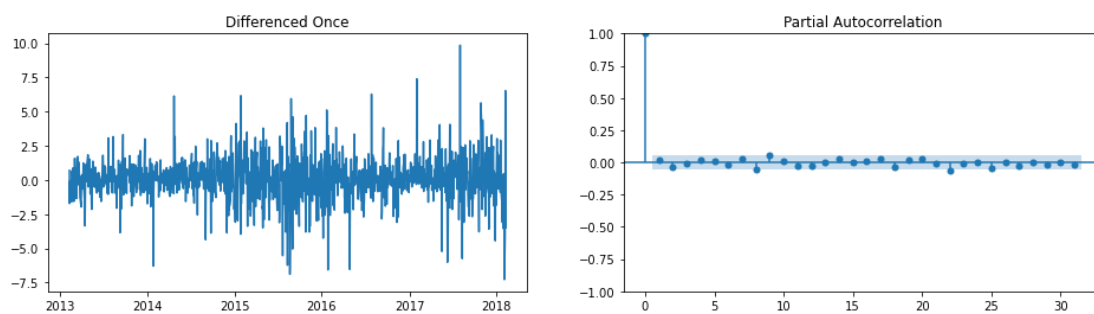


Figure 36: PACF plot of differenced (once) closing prices

In figure 36, the aim is to find the lag number crossing the shaded region, that would be the value of p . In this case, there is no lag crossing the shaded region except for 0 hence the p value is 0 for ARIMA model.

Next, I had to find the optimum value of q and in order to do so I had to analyse the ACF plot of differenced series i.e., figure 33. Similar to p , we have to find the lag number crossing the shaded region in ACF plot and that would be the ideal value of q . Since there are no lag values to be found crossing the shaded region hence the q value would be 0.

At this point, I have manually figured out the required ARIMA parameters i.e. $(p, d, q) = (0, 1, 0)$ and stand ready for modelling. I performed the required pre-processing of the data as discussed in section 3.4.1 and trained the ARIMA model using the training data.

SARIMAX Results

Dep. Variable:	y	No. Observations:	1007			
Model:	ARIMA(0, 1, 0)	Log Likelihood	-1829.083			
Date:	Sat, 27 Aug 2022	AIC	3660.166			
Time:	17:52:49	BIC	3665.080			
Sample:	0	HQIC	3662.033			
	- 1007					
Covariance Type:	opg					
	coef	std err	z	P> z	[0.025	0.975]
sigma2	2.2221	0.060	37.140	0.000	2.105	2.339
Ljung-Box (L1) (Q):	0.51	Jarque-Bera (JB):	519.68			
Prob(Q):	0.48	Prob(JB):	0.00			
Heteroskedasticity (H):	2.10	Skew:	-0.10			
Prob(H) (two-sided):	0.00	Kurtosis:	6.51			

Figure 37: Summary of ARIMA model

Next part was to predict the values using the trained model and then evaluate the model using the test data.



Figure 38: Actual vs Predicted of ARIMA

By having a look at the plot above, ARIMA does not seem to provide promising results to predict the future stock prices. I have calculated error of the model i.e., Root Mean Squared Error (RMSE) value of the model.

```
from math import sqrt
root_mse = sqrt(mean_squared_error(eval_df['Actual'],eval_df['predictions']))
print(root_mse)

27.05075384955778
```

Figure 39: RMSE of ARIMA

The model is as good as the RMSE value closer to 0. This is clearly a larger value which is not a good for a model.

4.2.1.1 Hyper tuning of ARIMA parameters

Since the results did not turn out to be promising with the manual determination of ARIMA parameters. I attempted to take a different path to determine the best combination of parameters to obtain better results from ARIMA model.

GRID SEARCH: The first methodology is grid search in which the process of implementing ARIMA model with different possible combinations of parameters can be automated (How to Grid Search ARIMA Model Hyperparameters with Python n.d.). This approach is divided into 2 parts –

1.) Evaluate the ARIMA model

This function consists of 3 steps-

- 1.) Split the series in training and testing.
- 2.) Run the loop over the length of testing dataset performing the following tasks in the specified sequence
 - i. Train the model.
 - ii. Create one step prediction.
 - iii. Store prediction as well as actual value.
- 3.) Evaluate the RMSE value using test data and predictions.

2.) Evaluate the set of ARIMA parameters

In this part, we have to iterate over all the possible combinations available of ARIMA parameters and then call the function created in the first part. First part of the function will return the RMSE value. As we iterate through all the values of p,d,q, we have to keep updating the best RMSE score and corresponding parameter order as soon as the lower RMSE is encountered.

In the end, this method will return least possible RMSE value and parameters responsible for it. The space and time complexity of method are very high of this method hence it is not preferable.

AUTO ARIMA: Auto Arima is an in-built module of library Pmdarima which accepts series and the range of parameters as its arguments and provides AIC score of multiple combinations of parameters. Based on the lowest AIC value, the corresponding parameters are the best fit for the model. This method is quicker and efficient than grid search methodology hence it is more preferable.

I ran the Auto Arima as well to confirm the parameters which I determined manually as shown in the code snippet below-

```

stepwise_fit = auto_arma(df_aapl['close'], start_p=1, start_q=1,
                        test='adf',
                        max_p=9, max_q=2,
                        m=1,
                        d=1,
                        seasonal=False,
                        start_P=0,
                        D=None,
                        trace=True,
                        error_action='ignore',
                        suppress_warnings=True,
                        stepwise=False)

ARIMA(0,1,0)(0,0,0)[0] intercept : AIC=4716.112, Time=0.05 sec
ARIMA(0,1,1)(0,0,0)[0] intercept : AIC=4717.799, Time=0.14 sec
ARIMA(0,1,2)(0,0,0)[0] intercept : AIC=4718.540, Time=0.24 sec
ARIMA(1,1,0)(0,0,0)[0] intercept : AIC=4717.820, Time=0.11 sec
ARIMA(1,1,1)(0,0,0)[0] intercept : AIC=4718.052, Time=0.50 sec
ARIMA(1,1,2)(0,0,0)[0] intercept : AIC=4720.484, Time=0.37 sec
ARIMA(2,1,0)(0,0,0)[0] intercept : AIC=4718.455, Time=0.18 sec
ARIMA(2,1,1)(0,0,0)[0] intercept : AIC=4720.406, Time=0.44 sec
ARIMA(2,1,2)(0,0,0)[0] intercept : AIC=inf, Time=1.30 sec
ARIMA(3,1,0)(0,0,0)[0] intercept : AIC=4720.332, Time=0.23 sec
ARIMA(3,1,1)(0,0,0)[0] intercept : AIC=4722.194, Time=0.58 sec
ARIMA(3,1,2)(0,0,0)[0] intercept : AIC=inf, Time=1.65 sec
ARIMA(4,1,0)(0,0,0)[0] intercept : AIC=4721.630, Time=0.27 sec
ARIMA(4,1,1)(0,0,0)[0] intercept : AIC=4723.534, Time=0.90 sec
ARIMA(5,1,0)(0,0,0)[0] intercept : AIC=4723.450, Time=0.35 sec

Best model: ARIMA(0,1,0)(0,0,0)[0] intercept
Total fit time: 7.312 seconds

```

Figure 40: Code snippet of Auto ARIMA

From the Auto ARIMA as well, I got the parameters which I had determined through ACF and PACF plots.

4.2.2 Linear Regression

Linear regression is pretty much straightforward after the pre-processing of data and feature engineering that I discussed in section 3.4.2. I trained the model using training features and training labels and then I predicted the value using testing features. Once I put the actual and predicted values together in a dataframe, I plot them to visualize the prediction accuracy of the model.



Figure 41: Actual vs Predicted of Linear regression

Linear Regression (LR) model seems to be doing much better than the ARIMA. I had calculated the RMSE score of LR model using actual and predicted values.

```
from sklearn.metrics import mean_squared_error
from math import sqrt
root_mse = sqrt(mean_squared_error(eval_df['close'],eval_df['Predictions']))
print(root_mse)

1.8895731120065877
```

Figure 42: RMSE of Linear Regression

The RMSE value is very much closer to the 0 that means it is a good model and was able to estimate the future stock prices well.

4.2.3 Long Short-Term Memory (LSTM)

Post the data pre-processing for LSTM as discussed in the section 3.4.3, I imported 3 different modules from Keras library i.e., Sequential, LSTM and Dense. All these modules are required for building LSTM network. LSTM model is unlike any other machine learning model. The model is built in the following steps-

- 1.) Module Sequential is instantiated as the model.
- 2.) LSTM is added to the model along with some required parameters.
- 3.) Dense layer is added to model.
- 4.) Model is compiled using a loss function and an optimizer
- 5.) Model is trained using training and testing data prepared in pre-processing stage.

Once the model is trained, I made predictions using test inputs dataset. Once I put the actual and predicted values together in a dataframe, I plot them to visualize the prediction accuracy of the model.



Figure 43: Actual vs Predicted of LSTM

From the visualisation, LSTM was able to capture the pattern of test labels or actual outputs quite well. Another insight from the visualisation is that the predictions are pretty much accurate in the near future and gap between actual and predicted is widening over time. I had calculated the RMSE score using actual and predicted values for LSTM model.

```
from sklearn.metrics import mean_squared_error
from math import sqrt
root_mse = sqrt(mean_squared_error(eval_df['close'],eval_df['Predictions']))
print(root_mse)

9.950361183772229
```

Figure 44: RMSE of LSTM

The RMSE score is not too high and hence it turned out to be a descent model for stock price forecasting.

4.3 Generalised Results

So far, I had only been using the stock prices of Apple Inc for analysis and modelling and the results that I have obtained stands valid only for Apple Inc. For the generalised results, I filtered out the stock prices of top 50 companies of S&P 500. The idea here was to obtain the RMSE value for every model that I have used for every company (top 50). This would give us an idea which model is better generally for dealing with stock prices. In this segment, I implemented the following in the given sequence-

- 1.) Filtered out the top 50 ticker names from the raw data set in a separate list.
- 2.) Created a list of dataframes of closing price series of top 50 companies.
- 3.) Created a user defined function for ARIMA model that includes the pre-processing of the data functionality to modelling and providing RMSE value in return.
- 4.) Created another user defined function for LR that includes the pre-processing of the data functionality to modelling and providing RMSE value in return.
- 5.) Created last user defined function for LSTM that includes the pre-processing of the data functionality to modelling and providing RMSE value in return.
- 6.) Created 3 empty lists to store RMSE score of each model.
- 7.) I ran the loop over the list of dataframe of top 50 companies to perform the following tasks-
 - a. Invoke ARIMA function and append the returned value to empty list 1.
 - b. Invoke LR function and append the returned value to empty list 2.
 - c. Invoke LSTM function and append the returned value to empty list 3.

- 8.) Concatenated the 4 lists into one dataframe – Top 50 company, empty list 1 as RMSE for ARIMA, empty list 2 as RMSE for LR and empty list 3 as RMSE for LSTM.
- 9.) Exported the final dataframe into csv format and average of each column would give us the generalised performance result of each model.

Ticks	ARIMA_error	LR_error	LSTM_error
PCLN	269.1172497	26.43080193	45.16849295
GOOG	115.026549	11.87115031	16.28374643
GOOGL	166.0940997	11.15880778	55.92993981
AZO	117.0125536	10.71557102	10.97464483
AMZN	250.7477814	14.95108532	35.61193656
CMG	75.9581734	7.270943222	10.43073171
REGN	81.04042412	7.944734119	8.290654714
MTD	138.3103775	6.505132157	46.40636092
BLK	85.064721	5.01025856	20.19324018
BIIB	42.28077989	4.573669351	5.037188951
EQIX	61.28779194	5.061299967	23.13752725
SHW	66.39051164	3.630149562	17.58398907
ADS	20.99109208	3.973168808	4.242291
GWW	60.25372056	4.648095943	7.289645289
AGN	33.58549996	2.82764753	3.361665028
TDG	31.2985497	3.852840223	4.139371575
CHTR	32.57589702	5.370146779	7.864950865
ESS	28.35331197	2.3086948	3.625923753
LMT	46.22773417	2.58608593	8.586117895
ORLY	40.36645378	4.539514555	4.911103039
ISRG	107.1004102	4.863744679	31.90714279
PSA	10.59846863	2.351746855	2.357756821
GS	14.58513351	3.071508904	4.637131983
RE	19.3171138	3.341178908	8.590652959
MHK	40.77866451	2.330980153	7.018248384
AMG	23.85223001	2.209735725	2.274327619
SPG	16.75879973	2.094753144	2.074580111
NOC	48.7917491	2.69170727	12.43853606
ROP	49.56188699	2.35030941	26.83690432
ULTA	40.05258918	4.490374464	5.408506602
MCK	16.02446574	2.430044989	2.884017135
IBM	24.47716167	1.68937301	1.777592563
AYI	32.00491462	4.162731331	4.661839976
COO	43.30385513	3.214239758	10.29052014
PXD	22.33727409	3.026721574	2.988556428

HUM	47.48510141	3.092361634	12.23896187
FDX	35.69311746	2.401710021	8.716597763
WHR	9.658771386	2.515252841	2.766580207
AVB	11.17024179	1.561578229	2.239912338
MMM	41.02926908	2.136881455	8.986987857
ILMN	38.46947235	3.356353268	4.042529989
MLM	14.39416482	3.638116011	3.59904767
BA	83.23323113	3.850827929	30.5147794
BDX	28.19477927	2.285918805	9.41270421
AMGN	11.40197028	2.060452128	2.423162936
ALXN	11.8311611	2.360801965	2.491094061
COST	12.31510422	2.176702768	3.110484523
BRK.B	21.81951114	1.649728114	4.010892358
GD	18.39584099	2.049383379	7.500436817
SNA	12.4622961	1.928862277	2.054983747
Average	53.38164043	4.532277578	11.38649983

Table 3:RMSE of Top 50 companies

According to the analysis conducted by the stock prices of Top 50 companies of S&P 500, I would say that LR emerged as a better model among all of them. However, from the previous exploration and visualisation, LSTM is a good fit for prediction in near future and lastly, ARIMA is not a good fit for predicting stock prices.

5. Conclusion and Future Work

This work is majorly consisting of 4 parts – data exploration, data processing, data modelling and hyper tuning. I collected and structured 5 years of stock prices of a stock market index i.e., S&P 500. I explored the data thoroughly using time-series data exploration methodologies. I performed pre-processing on data depending on the requirement of the algorithm. I did the feature engineering for the facilitate the machine learning methodology with the time series data. I trained three different types of models and was able to evaluate the results with each other. I was able to develop the program that would automate the ARIMA parameters determination process. At the end, I attempted to generalize the results from different models. At the completion of the project, I was successfully able to produce the answers to my research questions.

Research Questions	Potential Answers
What would be the best suited parameters to implement ARIMA efficiently?	Best parameters can be obtained from Auto ARIMA library or Grid Search Methodology.
Which power transformation method would be appropriate for the data?	Logarithmic transformation facilitates the normalization of data.
Does the data follow any seasonality and trend?	No evidence found for seasonality, but data does contain trends.
How would the data help to develop neural network-based prediction model?	Data was manipulated in such a way where the input variable contained the list of previous 3 values as single item in an array and 4 th value is corresponding label
Which model would have the higher accuracy?	Despite having more complex architecture of LSTM, LR outperformed all other algorithms used in the project
What would be criteria of choosing the best model?	As less as the Root mean squared error value is, the better would be the model.

Table 4: Potential answers to research questions

Given the scope of stock market, I'd like to incorporate the social media information into my feature engineering process. Updated news and tweets are one of the main drivers of stock market and social media is the only place where the news or any update regarding anything appears even before getting on television. Something like

this could definitely be used as a predictor in the model. As of now, I only attempted at making predictions based on the historical patterns hence the project would be way too risky to be deployed in the industry. But if more valuable factors like social media, news, geo-political factors can be incorporated into the project then it might have a small chance of being industry fit.

6. Appendices

6.1 Exploratory Data Analysis (EDA) and Base Model

Description: The following code is responsible for performing EDA, determination of parameters for ARIMA model and implementation of ARIMA model.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:

import pandas as pd
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt
import datetime

# In[2]:

df = pd.read_csv('all_stocks_5yr.csv')

# In[3]:

df.head()

# In[4]:

df.tail()

# In[5]:

df.shape

# In[6]:

df.info()
```

```
# In[7]:

df['date'] = pd.to_datetime(df["date"], format="%Y/%m/%d")

# In[8]:

df.info()

# In[9]:

stocks = list(df.Name.unique())

# In[10]:

type(stocks)

# In[11]:

#Number of stock available in the data
len(stocks)

# In[12]:

print(stocks)

# In[13]:

#Lets pick one stock at this time and try to work around it

# In[14]:

#AAPL

df_aapl = df[df['Name']=='AAPL']
```

```
# In[15]:

df_aapl.head()

# In[16]:

df_aapl.info()

# In[17]:

#Description of apple stock prices
df_aapl.describe()

# In[18]:

fig, (axis1, axis2) = plt.subplots(2, 1, figsize=(18,15))

axis1.plot(df_aapl['date'],df_aapl['open'], color = 'black', label =
'open')
axis1.plot(df_aapl['date'],df_aapl['high'], color = 'green', label =
'high')
axis1.plot(df_aapl['date'],df_aapl['low'], color = 'red', label =
'low')
axis1.plot(df_aapl['date'],df_aapl['close'], color = 'blue', label =
'close')
axis1.set_xlabel('Dates')
axis1.set_ylabel('Stock Prices')
axis1.set_title('Stock prices history')
axis1.legend()

axis2.plot(df_aapl['date'],df_aapl['volume'], color = 'orange', label =
'volume')
axis2.set_xlabel('Dates')
axis2.set_ylabel('Stocks Volume')
axis2.set_title('Volume traded overtime')
axis2.legend()

plt.show()
```

There's a significant decline in stock prices is visible between 2015 and mid of 2016 but overall trend of stock prices is positive.

The volume of `stocks` traded has been decreased `with` time. Possibly due to high prices of the stocks.

```
# In[19]:
```

```
df_aapl.index
```

```
# In[20]:
```

```
#Reset Index  
df_aapl.reset_index()
```

```
# In[21]:
```

```
#Set date as index in dataframe  
df_aapl = df_aapl.set_index('date', drop=True)
```

```
# In[22]:
```

```
df_aapl.head()
```

```
# In[23]:
```

```
#check if the datatype is correct  
df_aapl.index
```

```
# In[24]:
```

```
#Insights by resampling  
df_aapl.resample(rule = 'A').min()
```

```
# In[25]:
```

```
df_aapl.resample(rule = 'A').max()
```

```
# In[26]:
```

```
#Yearly insights
fig, ax = plt.subplots(figsize = (10,5))

ax.plot(df_aapl.resample(rule = 'A').min()['close'], color='red', label
= 'Minimum Price')
ax.plot(df_aapl.resample(rule = 'A').max()['close'], color='blue',
label = 'Maximum Price')
ax.set_xlabel('Dates')
ax.set_ylabel('Stock Prices')
ax.set_title('Minimum and Maximum closing prices -- YEARLY')
ax.legend()
```

```
plt.show()
```

The minimum and maximum prices per year follows the same pattern. The prices shoots up from 2017.

```
# In[27]:
```

```
df_aapl.resample(rule = 'Q').min()
```

```
# In[28]:
```

```
#Quarterly insights
fig, ax = plt.subplots(figsize = (14,5))

ax.plot(df_aapl.resample(rule = 'Q').min()['close'], color='red', label
= 'Minimum Price')
ax.plot(df_aapl.resample(rule = 'Q').max()['close'], color='blue',
label = 'Maximum Price')
ax.set_xlabel('Dates')
ax.set_ylabel('Stock Prices')
ax.set_title('Minimum and Maximum closing prices -- QUARTERLY')
ax.legend()
```

```
plt.show()
```

Prices started to drop from third quarter of 2015 to second quarter of 2016 and then the rise in prices can be seen.

```
# In[29]:
```

```
#Quarterly insights
fig, ax = plt.subplots(figsize = (15,5))
```



```
ax.plot(df_aapl.resample(rule = 'M').min()['close'], color='red', label
= 'Minimum Price')
ax.plot(df_aapl.resample(rule = 'M').max()['close'], color='blue',
label = 'Maximum Price')
ax.set_xlabel('Dates')
ax.set_ylabel('Stock Prices')
ax.set_title('Minimum and Maximum closing prices -- MONTHLY')
ax.legend()
```

```
plt.show()
```

```
# In[30]:
```

```
samp = df_aapl.resample(rule = 'M').min()
print(samp.index)
```

```
# In[31]:
```

```
#Closer insights between 2015 and 2017 insights
df_aapl_15_17 = df_aapl["2014-12-31":"2016-12-31"]
```

```
fig, ax = plt.subplots(figsize = (15,5))
```

```
ax.plot(df_aapl_15_17.resample(rule = 'M').min()['close'], color='red',
label = 'Minimum Price')
ax.plot(df_aapl_15_17.resample(rule = 'M').max()['close'],
color='blue', label = 'Maximum Price')
ax.set_xlabel('Dates')
ax.set_ylabel('Stock Prices')
ax.set_title('Minimum and Maximum closing prices -- Between 2015 and
2017')
ax.legend()
```

```
plt.show()
```

Prices precisely started to go down in 5th month of 2015 and then started to rise back up from 6th month of 2016.

```
# In[32]:
```

```
#Smoothing
df_aapl_ma = pd.DataFrame(df_aapl['close'].copy())
```

```
# In[33]:

type(df_aapl_ma)

# In[34]:

df_aapl_ma.head()

# In[35]:

df_aapl_ma['ma_5'] = df_aapl_ma['close'].rolling(5).mean()
df_aapl_ma['ma_10'] = df_aapl_ma['close'].rolling(10).mean()
df_aapl_ma['ma_15'] = df_aapl_ma['close'].rolling(15).mean()
df_aapl_ma['ma_20'] = df_aapl_ma['close'].rolling(20).mean()
df_aapl_ma['ma_30'] = df_aapl_ma['close'].rolling(30).mean()

# In[36]:

df_aapl_ma.head()

# In[37]:

fig, ax = plt.subplots(figsize = (16,5))

ax.plot(df_aapl_ma['close'], color='black', label = 'Actual Closing
Price')
ax.plot(df_aapl_ma['ma_5'], color='blue', label = 'Moving Average by
5')
ax.plot(df_aapl_ma['ma_10'], color='red', label = 'Moving Average by
10')
ax.plot(df_aapl_ma['ma_15'], color='green', label = 'Moving Average by
15')
ax.plot(df_aapl_ma['ma_20'], color='orange', label = 'Moving Average by
20')
ax.plot(df_aapl_ma['ma_30'], color='magenta', label = 'Moving Average
by 30')
ax.set_xlabel('Dates')
ax.set_ylabel('Stock Prices')
ax.set_xlim(datetime.date(2014,12,31),datetime.date(2017,12,31))
ax.set_title('Moving Averages of closing price')
ax.legend()
```

```
plt.show()
```

As we increase the window size of the rolling **function**, the prizes are getting more smoothened out along the duration.

```
# In[38]:
```

```
#Exponential weighted moving average (EWMA)
```

```
df_aapl_ma['ema_20'] = df_aapl_ma['close'].ewm(span=20).mean()
```

```
df_aapl_ma['ema_30'] = df_aapl_ma['close'].ewm(span=30).mean()
```

```
df_aapl_ma.head()
```

```
# In[39]:
```

```
fig, ax = plt.subplots(figsize = (16,5))
```

```
ax.plot(df_aapl_ma['close'], color='black', label = 'Actual Closing Price')
```

```
ax.plot(df_aapl_ma['ma_20'], color='magenta', label = 'Moving Average by 20')
```

```
ax.plot(df_aapl_ma['ema_20'], color='blue', label = 'Exponential Moving Average by 20')
```

```
ax.set_xlabel('Dates')
```

```
ax.set_ylabel('Stock Prices')
```

```
ax.set_xlim(datetime.date(2014,12,31),datetime.date(2017,12,31))
```

```
ax.set_title('Exponential Moving Averages of closing price')
```

```
ax.legend()
```

```
plt.show()
```

```
# In[40]:
```

```
fig, ax = plt.subplots(figsize = (16,5))
```

```
ax.plot(df_aapl_ma['close'], color='black', label = 'Actual Closing Price')
```

```
ax.plot(df_aapl_ma['ma_30'], color='magenta', label = 'Moving Average by 30')
```

```
ax.plot(df_aapl_ma['ema_30'], color='blue', label = 'Exponential Moving Average by 30')
```

```
ax.set_xlabel('Dates')
```

```

ax.set_ylabel('Stock Prices')
#ax.set_xlim(datetime.date(2014,12,31),datetime.date(2017,12,31))
ax.set_title('Exponential Moving Averages of closing price')
ax.legend()

plt.show()

# In[41]:

df_aapl_ma['ema_0.2'] = df_aapl_ma['close'].ewm(alpha = 0.2).mean()
df_aapl_ma['ema_0.6'] = df_aapl_ma['close'].ewm(alpha = 0.6).mean()
df_aapl_ma['ema_0.9'] = df_aapl_ma['close'].ewm(alpha = 0.9).mean()

df_aapl_ma.head()

# In[120]:

fig, ax = plt.subplots(figsize = (16,5))

ax.plot(df_aapl_ma['close'], color='black', label = 'Actual Closing Price')
ax.plot(df_aapl_ma['ema_0.2'], color='magenta', label = 'EMA with alpha 0.2')
ax.plot(df_aapl_ma['ema_0.6'], color='blue', label = 'EMA with alpha 0.4')
#ax.plot(df_aapl_ma['ema_0.9'], color='red', label = 'EMA with alpha 0.9')

ax.set_xlabel('Dates')
ax.set_ylabel('Stock Prices')
ax.set_xlim(datetime.date(2014,12,31),datetime.date(2016,12,31))
ax.set_title('Exponential Moving Averages of closing price')
ax.legend()

plt.show()

```

The lesser the alpha is, the more smooth the data gets. As the value of alpha increases, it takes almost the path of original data. From the performed EDA, we were able to derive the following insights from the Apple stock-

1. The stock prices contains white noise and follows a random walk.
2. Data shows the positive trend in the stock prices.

3. Not enough evidence found to guarantee the presence of seasonality in the stock closing prices. In order to perform the **ARIMA**, the series should be stationary.

In[43]:

#Test for stationarity

```
from statsmodels.tsa.stattools import adfuller
```

I am conducting Augmented Dickey Fuller test to check if the closing price series is stationary or not.

H0: The series is non-stationary

H1: The series is stationary

if the P-value is less than significance level (0.05) then we would reject the null hypothesis.

In[44]:

```
result = adfuller(df_aapl['close'])
print(f"ADF Statistic: {result[0]}")
print(f"P-value: {result[1]}")
```

Since the P-value is much larger than significance value, I cannot reject null hypothesis. This implies that my series is not stationary and I need perform differencing to a certain degree in order to make the series stationary.

In[45]:

#Auto Correlation Function (ACF)

```
from statsmodels.graphics.tsaplots import plot_acf
```

In[46]:

```
fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))
```

```
ax1.plot(df_aapl['close'])
ax1.set_title('Original')
plot_acf(df_aapl['close'], ax=ax2);
```

In[47]:

#Differencing (d)

```
diff_1 = df_aapl['close'].diff().dropna()

fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))

ax1.plot(diff_1)
ax1.set_title('Differenced Once')
plot_acf(diff_1, ax=ax2);
```

In[48]:

```
diff_2 = df_aapl['close'].diff().diff().dropna()

fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))

ax1.plot(diff_2)
ax1.set_title('Differenced Twice')
plot_acf(diff_2, ax=ax2);
```

It looks like Differencing (d) 1 is good enough to fit ARIMA. Let us confirm the same below using pmdarima package.

In[49]:

```
from pmdarima.arima.utils import ndiffs
```

In[50]:

```
ndiffs(df_aapl['close'], test="adf")
```

This confirms the value of differencing (d = 1) for implementing ARIMA.

In[51]:

#Lets find out the optimal value of p for ARIMA model

```
from statsmodels.graphics.tsaplots import plot_pacf
```

In[52]:

```
fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))

ax1.plot(diff_1)
ax1.set_title('Differenced Once')
plot_pacf(diff_1, ax=ax2);
```

From the above plot, I can take the value of P as 8 or 9.

```
# In[53]:
```

```
#Lets find out the optimal value of q for ARIMA model
```

```
fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))
```

```
ax1.plot(diff_1)
```

```
ax1.set_title('Differenced Once')
```

```
plot_acf(diff_1, ax=ax2);
```

From the above plot, I can take the value of Q as 2.

```
# In[54]:
```

```
#Fitting the ARIMA model
```

```
from statsmodels.tsa.arima.model import ARIMA
```

```
# In[55]:
```

```
model = ARIMA(df_aapl['close'], order=(9,1,2))
```

```
result_1 = model.fit()
```

```
# In[56]:
```

```
print(result_1.summary())
```

```
# In[57]:
```

```
model_2 = ARIMA(df_aapl['close'], order=(8,1,2))
```

```
result_2 = model_2.fit()
```

```
print(result_2.summary())
```

```
# In[58]:
```

```
#result_2.forecast()
```

```
# In[59]:
```

```
#plot the residuals

residual_1 = pd.DataFrame(result_1.resid)

fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))

ax1.plot(residual_1)
ax2.hist(residual_1, density=True)

plt.show()

# In[60]:

residual_2 = pd.DataFrame(result_2.resid)

fig, (ax1,ax2) = plt.subplots(1,2, figsize=(16,4))

ax1.plot(residual_2)
ax2.hist(residual_2, density=True)

plt.show()

Let us find out the best model through Auto ARIMA
# In[61]:

from pmdarima import auto_arima
from sklearn.metrics import mean_squared_error
import warnings
warnings.filterwarnings("ignore")

# In[62]:

stepwise_fit = auto_arima(df_aapl['close'], start_p=1, start_q=1,
                          test='adf',
                          max_p=9, max_q=2,
                          m=1,
                          d=1,
                          seasonal=False,
                          start_P=0,
                          D=None,
                          trace=True,
                          error_action='ignore',
```



```
suppress_warnings=True,
stepwise=False)
```

Now that we have the order for best possible model, let's finalize the ARIMA model with the order and evaluate the results.

```
# In[63]:
```

```
order = stepwise_fit.get_params()['order']
```

```
# In[64]:
```

```
#Pre-processing of data - Train and Test split
```

```
n = int(len(df_aapl) * 0.8)
train = df_aapl['close'][:n]
test = df_aapl['close'][n:]
```

```
# In[79]:
```

```
print(df_aapl['close'].head())
print('*****')
print(df_aapl['close'].tail())
print('*****')
```

```
# In[84]:
```

```
print(train.head())
print('*****')
print(test.tail())
print('*****')
```

```
# In[65]:
```

```
print(len(train))
print(len(test))
```

```
# In[66]:
```

```
len(df_aapl) == len(train) + len(test)
```

```
# In[67]:

train.head()

# In[68]:

model = ARIMA(train.values, order = order)
result = model.fit()
result.summary()

# In[69]:

res = pd.Series(result.forecast(len(test)))

# In[70]:

result.forecast()

# In[71]:

type(res)

# In[72]:

res_df = res.to_frame(name='predictions')
res_df = res_df.reset_index(drop=True)

# In[73]:

test_df = test.to_frame(name='Actual')
test_df = test_df.reset_index()

# In[74]:
```

```

eval_df = pd.concat([test_df, res_df], axis=1)
eval_df = eval_df.set_index('date', drop=True)

# In[75]:

eval_df

# In[76]:

fig, (axis1, axis2) = plt.subplots(2, 1, figsize=(18,15))

axis1.plot(train, color = 'blue', label = 'Train')
axis1.plot(test, color = 'green', label = 'Test')
axis1.plot(eval_df.predictions, color = 'orange', label = 'Predicted',
linestyle = '--')
axis1.set_xlabel('Dates')
axis1.set_ylabel('Stock Prices')
axis1.set_title('Stock Price prediction')
axis1.legend()

axis2.plot(test, color = 'green', label = 'Actual')
axis2.plot(eval_df.predictions, color = 'orange', label = 'Predicted',
linestyle = '--')
axis2.set_xlabel('Dates')
axis2.set_ylabel('Stock Prices')
axis2.set_title('Stock Price prediction')
axis2.legend()

plt.show()

# In[103]:

'''start = len(train)
end = len(train) + len(test) - 1
pred = result.get_prediction(start=start, end=end)
print(pred.predicted_mean)'''

# In[108]:

#Evaluation of the model

test.values.mean()

```

```
# In[107]:
```

```
from math import sqrt
root_mse =
sqrt(mean_squared_error(eval_df['Actual'],eval_df['predictions']))
print(root_mse)
```

Interpretation of the `model`: The error is much higher with the ARIMA model and it should be as less as possible. The more the error is closer to 0, the better is the model.

6.2 Hyper-tuning of ARIMA parameters

Description: The following code is the implementation of Grid Search to determine the best set of parameters to model ARIMA.

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
import warnings
from math import sqrt
import pandas as pd
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
warnings.filterwarnings("ignore")
```

```
# In[3]:
```

```
def evaluate_arima_model(X, arima_order):
    #prepare training dataset
    train_size = int(len(X) * 0.80)
    train, test = X[0:train_size], X[train_size:]
    history = [x for x in train]
    # make predictions
    predictions = list()
    for t in range(len(test)):
        model = ARIMA(history, order=arima_order)
        model_fit = model.fit()
        yhat = model_fit.forecast()[0]
```

```

        predictions.append(yhat)
        history.append(test[t])
    # calculate out of sample error
    rmse = sqrt(mean_squared_error(test, predictions))
    return rmse

# In[4]:

# evaluate combinations of p, d and q values for an ARIMA model
def evaluate_models(dataset, p_values, d_values, q_values):
    dataset = dataset.astype('float32')
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    rmse = evaluate_arima_model(dataset, order)
                    if rmse < best_score:
                        best_score, best_cfg = rmse, order
                    print('ARIMA%s RMSE=%.3f' % (order,rmse))
                except:
                    continue
    print('Best ARIMA%s RMSE=%.3f' % (best_cfg, best_score))

# In[2]:

#Load the data
df = pd.read_csv('all_stocks_5yr.csv',parse_dates=True)

# In[3]:

#Pick one stock
df = df[df['Name']=='AAL']

# In[4]:

df.head()

# In[16]:

```

```
p_values = range(0,10)
d_values = range(0, 3)
q_values = range(0, 3)
warnings.filterwarnings("ignore")
evaluate_models(df.close.values, p_values, d_values, q_values)

# In[ ]:
```

6.3 Linear Regression Model

Description: The following code is responsible for pre-processing the data for LR, modelling and evaluation of LR.

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:

import pandas as pd
import numpy as np

# In[2]:

df = pd.read_csv('all_stocks_5yr.csv', parse_dates=True,
index_col='date')

# In[3]:

df.head()

# In[4]:

type(df.index)

# In[9]:
```

```
#Stock Selection
df = df[df['Name']=='AAPL']

# In[10]:

df.info()

# In[11]:

df.describe()

# In[14]:

df = df[['close']]

# In[16]:

df.head()

# In[27]:

def lagit(df, lags):
    names = list()
    for i in range(1,lags+1):
        df['lag_'+str(i)] = df['close'].shift(i)
        names.append('lag_'+str(i))
    df.dropna(inplace=True)
    return names

# In[28]:

lagnames = lagit(df,5)

# In[29]:

lagnames
```

```
# In[30]:

df.head()

# In[37]:

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# In[32]:

X = np.array(df[lagnames])

# In[35]:

Y = np.array(df['close'])

# In[40]:

#Preprocessing Data
x_train,x_test,y_train,y_test = train_test_split(X,Y, test_size=0.2,
shuffle=False)

# In[52]:

print(len(x_train))
print(len(x_test))

# In[41]:

#Model creation and training
lr = LinearRegression().fit(x_train,y_train)

# In[46]:

pred = lr.predict(x_test)
```



```
print(pred.shape)

# In[76]:

len(y_test)

# In[49]:

type(pred)

# In[64]:

eval_df = pd.DataFrame(df['close'][-251:])
eval_df = eval_df.reset_index()

# In[68]:

pred_df = pd.Series(pred).to_frame(name='Predictions')
pred_df = pred_df.reset_index(drop=True)

# In[70]:

eval_df = pd.concat([eval_df, pred_df], axis=1)
eval_df = eval_df.set_index('date', drop=True)

# In[71]:

eval_df

# In[51]:

import matplotlib.pyplot as plt

# In[73]:
```

```

fig, (axis1, axis2) = plt.subplots(2, 1, figsize=(18,15))

axis1.plot(df['close'][:-252], color = 'blue', label = 'Train')
axis1.plot(df['close'][-251:], color = 'green', label = 'Test')
axis1.plot(eval_df['Predictions'], color = 'orange', label =
'Predicted', linestyle = '--')
axis1.set_xlabel('Dates')
axis1.set_ylabel('Stock Prices')
axis1.set_title('Stock Price prediction')
axis1.legend()

axis2.plot(eval_df['close'], color = 'green', label = 'Actual')
axis2.plot(eval_df['Predictions'], color = 'orange', label =
'Predicted', linestyle = '--')
axis2.set_xlabel('Dates')
axis2.set_ylabel('Stock Prices')
axis2.set_title('Stock Price prediction')
axis2.legend()

plt.show()

# In[74]:

from sklearn.metrics import mean_squared_error
from math import sqrt
root_mse =
sqrt(mean_squared_error(eval_df['close'],eval_df['Predictions']))
print(root_mse)

```

The error is comparatively very low which indicates that the model is good.

6.4 LSTM Model

Description: The following code is responsible for pre-processing the data for LSTM, modelling, and evaluation of LSTM.

```

#!/usr/bin/env python
# coding: utf-8

# In[1]:

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

```

```
# In[2]:

df = pd.read_csv('all_stocks_5yr.csv', parse_dates=True,
index_col='date')

# In[3]:

df.head()

# In[4]:

type(df.index)

# In[5]:

df = df[df['Name']=='AAPL']

# In[6]:

df.head()

# In[7]:

df.info()

# In[9]:

df = df[['close']]

# In[10]:

df.describe()

Calculate Percent change
```

The reason for using the percent change instead of the actual prices is the benefit of normalization as we can measure all the variables in a comparable metric. Apart from that, returns have more statistical properties than actual prices like stationarity, as in most of the cases the prices cannot be stationary but we can have stationary returns.

A stationary time series is one where statistical properties such as mean, variance, standard deviation, correlation remains constant over time.

```
# In[11]:
```

```
df['returns'] = df.close.pct_change()
```

```
# In[12]:
```

```
df.head()
```

```
# In[13]:
```

```
df['log_returns'] = np.log(1 + df['returns'])
```

```
# In[14]:
```

```
df.head()
```

```
# In[21]:
```

```
plt.figure(figsize = (16,4))
plt.plot(df.log_returns)
plt.title('Log Returns of Closing Prices')
plt.xlabel('Dates')
plt.ylabel('Log Returns')
plt.show()
```

Our mean is constant throughout the time here.

```
# In[22]:
```

```
df.dropna(inplace = True)
```

```
# In[23]:

df.head()

# In[24]:

#Input variable
X = df[['close','log_returns']].values

# In[25]:

X

# In[26]:

from sklearn.preprocessing import MinMaxScaler

# In[27]:

scaler = MinMaxScaler(feature_range=(0,1)).fit(X)
X_scaled = scaler.transform(X)

# In[32]:

X_scaled[:5]

# In[29]:

#Output Variable
Y = [x[0] for x in X_scaled]

# In[31]:

Y[:5]
```

```
# In[34]:

#Train and test Split
split = int(len(X_scaled) * 0.8)
print(split)

# In[35]:

X_train = X_scaled[:split]
X_test = X_scaled[split:len(X_scaled)]
Y_train = Y[:split]
Y_test = Y[split:len(Y)]

# In[40]:

assert len(X_train) == len(Y_train)
assert len(X_test) == len(Y_test)

# In[102]:

len(X_test)

# In[50]:

n = 3
Xtrain = list()
Ytrain = list()
Xtest = list()
Ytest = list()
for i in range(n, len(X_train)):
    Xtrain.append(X_train[i-n:i, :X_train.shape[1]])
    Ytrain.append(Y_train[i])
for i in range(n, len(X_test)):
    Xtest.append(X_test[i-n:i, :X_test.shape[1]])
    Ytest.append(Y_test[i])

# In[71]:

Xtrain , Ytrain = (np.array(Xtrain), np.array(Ytrain))
```

```
Xtrain = np.reshape(Xtrain, (Xtrain.shape[0], Xtrain.shape[1],
Xtrain.shape[2]))

Xtest, Ytest = (np.array(Xtest), np.array(Ytest))
Xtest = np.reshape(Xtest, (Xtest.shape[0], Xtest.shape[1],
Xtest.shape[2]))

# In[75]:

print(Xtrain.shape)
print(Ytrain.shape)
print('---'*20)
print(Xtest.shape)
print(Ytest.shape)

# In[81]:

#LSTM Model
from keras.models import Sequential
from keras.layers import LSTM, Dense

# In[82]:

model = Sequential()
model.add(LSTM(4, input_shape=(Xtrain.shape[1], Xtrain.shape[2])))
model.add(Dense(1))
model.compile(loss="mean_squared_error", optimizer="adam")
model.fit(Xtrain, Ytrain, epochs=100, validation_data=(Xtest, Ytest),
batch_size = 16, verbose=1)

# In[83]:

model.summary()

# In[84]:

trainPredict = model.predict(Xtrain)
testPredict = model.predict(Xtest)
```

```
# In[103]:

len(testPredict)

# In[106]:

len(Ytest)

# In[114]:

trainPredict = np.c_[trainPredict, np.zeros(trainPredict.shape)]
testPredict = np.c_[testPredict, np.zeros(testPredict.shape)]

# In[116]:

trainPredict = scaler.inverse_transform(trainPredict)
trainPredict = [x[0] for x in trainPredict]

testPredict = scaler.inverse_transform(testPredict)
testPredict = [x[0] for x in testPredict]

# In[132]:

print(len(testPredict))
print(type(testPredict))

# In[118]:

df.close.tail()

# In[137]:

eval_df = pd.DataFrame(df['close'][-249:])
eval_df = eval_df.reset_index()

# In[138]:
```



```

pred_df = pd.Series(testPredict).to_frame(name='Predictions')
pred_df = pred_df.reset_index(drop=True)

# In[139]:

eval_df = pd.concat([eval_df, pred_df], axis=1)
eval_df = eval_df.set_index('date', drop=True)

# In[140]:

eval_df

# In[143]:

fig, (axis1, axis2) = plt.subplots(2, 1, figsize=(18,15))

axis1.plot(df['close'][:-249], color = 'blue', label = 'Train')
axis1.plot(df['close'][-249:], color = 'green', label = 'Test')
axis1.plot(eval_df['Predictions'], color = 'orange', label =
'Predicted', linestyle = '--')
axis1.set_xlabel('Dates')
axis1.set_ylabel('Stock Prices')
axis1.set_title('Stock Price prediction using LSTM')
axis1.legend()

axis2.plot(eval_df['close'], color = 'green', label = 'Actual')
axis2.plot(eval_df['Predictions'], color = 'orange', label =
'Predicted', linestyle = '--')
axis2.set_xlabel('Dates')
axis2.set_ylabel('Stock Prices')
axis2.legend()

plt.show()

# In[142]:

from sklearn.metrics import mean_squared_error
from math import sqrt
root_mse =
sqrt(mean_squared_error(eval_df['close'], eval_df['Predictions']))
print(root_mse)

```

If we consider the rmse value, it is a fine `model`. But analysing the above plot, the prediction is quite accurate in the near future and error is increasing over time.

```
# In[ ]:
```

6.5 Top 50 stocks

Description: The following code is responsible for implementation and evaluation of ARIMA, LR and LSTM on top 50 stocks of S&P 500 index.

```
#!/usr/bin/env python
```

```
# coding: utf-8
```

```
# In[1]:
```

```
import pandas as pd
```

```
import numpy as np
```

```
# In[2]:
```

```
df = pd.read_csv('all_stocks_5yr.csv', parse_dates=True,  
index_col='date')
```

```
# In[3]:
```

```
df.head()
```

```
# In[5]:
```

```
df = df[['Name', 'close']]
```

```
# In[21]:
```

```
df_grp = df.groupby('Name').mean()
```

```
# In[22]:
```

```
df_grp = df_grp.reset_index()

# In[23]:

df_grp = df_grp.sort_values(by=['close'],ascending = False)
df_grp = df_grp.reset_index(drop=True)

# In[83]:

df_grp.head()

# In[27]:

top_50 = list(df_grp['Name'][:50])

# In[80]:

top_50[:5]

# In[29]:

len(top_50)

# In[30]:

temp = [df, df_grp]

# In[36]:

top_50_dfnames = list()
for x in top_50:
    top_50_dfnames.append('df_'+x)

# In[37]:
```

```
top_50_dfnames[:5]

# In[53]:

#List of dataframes of top 50 stocks of S&P 500
top_50_dfs = list()
for i in range(50):
    top_50_dfnames[i] = df[df['Name']==top_50[i]]
    top_50_dfs.append(top_50_dfnames[i])

# In[54]:

len(top_50_dfs)

# In[69]:

type(top_50_dfs[2])

# In[74]:

from pmdarima import auto_arima
from math import sqrt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import warnings
warnings.filterwarnings("ignore")

# In[89]:

#ARIMA Model

def ARIMA_model(df):
    stepwise_fit = auto_arima(df['close'], start_p=1, start_q=1,
                              test='adf',
                              max_p=9, max_q=3,
                              m=1,
                              d=1, max_d=3,
                              seasonal=False,
```

```

        start_P=0,
        D=None,
        trace=False,
        error_action='ignore',
        suppress_warnings=True,
        stepwise=False)
order = stepwise_fit.get_params()['order']
n = int(len(df) * 0.8)
train = df['close'][:n]
test = df['close'][n:]
model = ARIMA(train.values, order = order)
result = model.fit()
res = pd.Series(result.forecast(len(test)))
res_df = res.to_frame(name='predictions')
res_df = res_df.reset_index(drop=True)
test_df = test.to_frame(name='Actual')
test_df = test_df.reset_index()
eval_df = pd.concat([test_df, res_df], axis=1)
eval_df = eval_df.set_index('date', drop=True)
root_mse =
sqrt(mean_squared_error(eval_df['Actual'], eval_df['predictions']))

return root_mse

# In[91]:

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

# In[92]:

#Linear Regression
def Linear_model(df):
    def lagit(df, lags):
        names = list()
        for i in range(1, lags+1):
            df['lag_'+str(i)] = df['close'].shift(i)
            names.append('lag_'+str(i))
        df.dropna(inplace=True)
        return names

    lagnames = lagit(df, 5)
    X = np.array(df[lagnames])
    Y = np.array(df['close'])

```

```

    x_train,x_test,y_train,y_test = train_test_split(X,Y,
test_size=0.2, shuffle=False)
    lr = LinearRegression().fit(x_train,y_train)
    pred = lr.predict(x_test)
    n = len(y_test)
    eval_df = pd.DataFrame(df['close'][-n:])
    eval_df = eval_df.reset_index()
    pred_df = pd.Series(pred).to_frame(name='Predictions')
    pred_df = pred_df.reset_index(drop=True)
    eval_df = pd.concat([eval_df,pred_df],axis=1)
    eval_df = eval_df.set_index('date', drop=True)
    root_mse =
sqrt(mean_squared_error(eval_df['close'],eval_df['Predictions']))

    return root_mse

```

In[97]:

```

from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import LSTM, Dense

```

In[98]:

#LSTM

```

def LSTM_model(df):
    df['returns'] = df.close.pct_change()
    df['log_returns'] = np.log(1 + df['returns'])
    df.dropna(inplace = True)
    X = df[['close', 'log_returns']].values
    scaler = MinMaxScaler(feature_range=(0,1)).fit(X)
    X_scaled = scaler.transform(X)
    Y = [x[0] for x in X_scaled]
    split = int(len(X_scaled) * 0.8)
    X_train = X_scaled[:split]
    X_test = X_scaled[split:len(X_scaled)]
    Y_train = Y[:split]
    Y_test = Y[split:len(Y)]
    n = 3
    Xtrain = list()
    Ytrain = list()
    Xtest = list()
    Ytest = list()
    for i in range(n,len(X_train)):
        Xtrain.append(X_train[i-n:i,:X_train.shape[1]])

```

```

        Ytrain.append(Y_train[i])
    for i in range(n, len(X_test)):
        Xtest.append(X_test[i-n:i, :X_test.shape[1]])
        Ytest.append(Y_test[i])
    Xtrain, Ytrain = (np.array(Xtrain), np.array(Ytrain))
    Xtrain = np.reshape(Xtrain, (Xtrain.shape[0], Xtrain.shape[1],
Xtrain.shape[2]))

    Xtest, Ytest = (np.array(Xtest), np.array(Ytest))
    Xtest = np.reshape(Xtest, (Xtest.shape[0], Xtest.shape[1],
Xtest.shape[2]))
    model = Sequential()
    model.add(LSTM(4, input_shape=(Xtrain.shape[1], Xtrain.shape[2])))
    model.add(Dense(1))
    model.compile(loss="mean_squared_error", optimizer="adam")
    model.fit(Xtrain, Ytrain, epochs=100, validation_data=(Xtest,
Ytest), batch_size = 16, verbose=1)
    trainPredict = model.predict(Xtrain)
    testPredict = model.predict(Xtest)
    trainPredict = np.c_[trainPredict, np.zeros(trainPredict.shape)]
    testPredict = np.c_[testPredict, np.zeros(testPredict.shape)]
    trainPredict = scaler.inverse_transform(trainPredict)
    trainPredict = [x[0] for x in trainPredict]

    testPredict = scaler.inverse_transform(testPredict)
    testPredict = [x[0] for x in testPredict]
    z = len(Ytest)
    eval_df = pd.DataFrame(df['close'][-z:])
    eval_df = eval_df.reset_index()
    pred_df = pd.Series(testPredict).to_frame(name='Predictions')
    pred_df = pred_df.reset_index(drop=True)
    eval_df = pd.concat([eval_df, pred_df], axis=1)
    eval_df = eval_df.set_index('date', drop=True)
    root_mse =
sqrt(mean_squared_error(eval_df['close'], eval_df['Predictions']))

    return root_mse

# In[99]:

temp = LSTM_model(top_50_dfs[2])

# In[100]:

temp

```

```
# In[101]:

ARIMA_error = list()
LR_error = list()

for x in top_50_dfs:
    ar_err = ARIMA_model(x)
    ARIMA_error.append(ar_err)
    lr_err = Linear_model(x)
    LR_error.append(lr_err)

# In[106]:

LSTM_error = list()

for x in top_50_dfs:
    lstm_err = LSTM_model(x)
    LSTM_error.append(lstm_err)

# In[107]:

len(LSTM_error)

# In[108]:

final_result = pd.DataFrame({'Ticks':top_50,
                             'ARIMA_error':ARIMA_error,
                             'LR_error':LR_error,
                             'LSTM_error':LSTM_error})

# In[109]:

final_result.head()

# In[110]:

final_result.to_csv('top_50.csv', index=False)
```



```
# In[ ]:
```

6.6 Ethical clearance documents

Context: The application was submitted to ethical committee of the university with all the data set and proposal of the project. The same application was returned with a decision if the researcher can proceed with the idea and selected data.



DUNDALK INSTITUTE OF TECHNOLOGY
School of Informatics & Creative Arts
Ethical Approval Form for Research Projects

Researcher Name: **Nishant Verma** Year: **2022** Course: **MSc. In Data Analytics**

Title of project: **Analysis and Prediction of stock prices**

Name of supervisor/s: **David O'Keeffe** Date: **25-Mar-22**
 (if applicable)

This application is to be completed by the researcher and where appropriate, in conjunction with the project supervisor. The lead researcher/supervisor is responsible for submitting the completed form to the appropriate Research Ethics Committee (details below)

Please note: If your submission is incomplete or unclear, your application will be returned to you and your project may be delayed.

Section 1

Type of Researcher

Please tick the appropriate box below to indicate the type of researcher you are:

Undergraduate ☐
 (proceed to section 2)

Completed Ethical Approval forms for undergraduate research should be submitted to the relevant Departmental Research Ethics Committee (DREC).

Drec.dcam@dkit.ie – Department of Creative Arts Media and Music
Drec.dcm@dkit.ie – Department of Computing Science and Mathematics
Drec.dvhcc@dkit.ie – Department of Visual and Human Centred Computing

Postgraduate ☒
 (proceed to section 3)

Completed Ethical Approval forms for Postgraduate research should be submitted directly to the School Research Ethics Committee (SREC).

Srec.ica@dkit.ie

Staff ☐
 (proceed to section 3)

Completed Ethical Approval forms for Staff research should be submitted directly to the School Research Ethics Committee (SREC).

Srec.ica@dkit.ie

ICA Research Ethics Approval Application Leathanach - Page 1

Section 2

Please complete questions 1-4 listed below.

	Human and / or Animal Research	YES	NO
1	<p>Does your research involve human participants other than the following¹?</p> <ul style="list-style-type: none"> • Research using exclusively secondary sources. • Research using materials legally accessible to the public that have legal protection, e.g., record of court judgements, data archives. • Research using materials that are publicly accessible and where there is no reasonable expectation for privacy, e.g., books, published third party interviews. • Observations of human behaviour in public where (i) those being observed have no reasonable expectation of privacy, (ii) there is no intervention on the part of the researcher nor any interaction between the researcher and those observed, and (iii) individuals are not identifiable in the results. <p>If 'YES', please complete B and C below.</p>		
2	<p>Does your project involve working with animals?</p> <p>– If 'YES', please complete B and C below. – Please note that for ethical consideration: 'Animals' are classed as vertebrate animals including cyclostomes and cephalopods (DIRECTIVE 2010/63/EU)</p>		
3	<p>Does your project involve working with participants from any of the following categories?</p> <ul style="list-style-type: none"> • Minors (under 18 years of age) • People with learning or communication difficulties • Patients • People in custody • People engaged in illegal activities <p>If 'YES', please complete D below.</p>		
4	<p>Does your project have any possible ethical implications other than those outlined in questions 1, 2 and 3?</p> <p>If 'YES', please complete E below.</p>		

A. I consider that this project has no significant ethical implications² to be brought through the ICA School Ethics Review Process
Please complete Section 4

☐

¹ If there is any doubt, researchers should contact the Chair of the SREC.

² In determining significant implications, please consider all potential risks attached to this project. If there is any doubt, researchers should contact the Chair of the SREC.

- B.**
I. Is this study part of a larger project that already has ethical clearance?

YES / NO

If **YES** please answer question B II.
 If **NO** please answer question C.

- II. If this study is part of a larger project that already has ethical clearance, are you proposing any changes to the operational plan already ethically approved?**

YES / NO

If **YES**, please complete *Sections 3 and 4*.
 If **NO**, then please provide the project details below and complete *Section 4*.

Title of project with ethical clearance: _____

- C. Could this project have ethical implications that should be brought before the appropriate ICA Departmental Ethics Review Committee as it will be carried out with human participants?**

YES / NO

If **YES**, please complete *Sections 3 and 4*.
 If **NO**, please complete *Section 4*.

- D. I consider that this project may have ethical implications that should be brought before the School Research Ethics Committee as it will be carried out with human participants in a "vulnerable" category.**
Please complete Sections 3 and 4 ☐

All research carried out with human participants in a vulnerable category must be referred by the Departmental Research Ethics Committee to the School Research Ethics Committee for approval.

- E. Could this project have ethical implications, other than those previously outlined, that should be brought before the appropriate ICA Departmental Ethics Review Committee?**

YES / NO

If **YES**, please complete *Sections 3 and 4*.
 If **NO**, please complete *Section 4*.

Section 3

3.1 Application Form Checklist

Please complete Section 3 and provide additional information as attachments.

My application includes the following documentation:	INCLUDED (mark as YES)	NOT APPLICABLE (mark as N/A)
Recruitment advertisement		✓
Participant Information Leaflet		✓
Participant Informed Consent form		✓
Questionnaire/Survey		✓
Interview/Focus Group Questions		✓
Debriefing material		✓
Evidence of approval to gain access to off-site location		✓
Ethical approval from external organizations. If ethical approval from external organizations is pending give details below		✓
Details		

3.2 Project Details

a) Lay description (Maximum 200 words)

Please outline, in terms that any non-expert would understand, what your research project is about, including what participants will be required to do. Please explain any technical terms or discipline-specific phrases.

In this research project, the objective is to build a system that would give recommendation to an investor whether to invest in a particular stock or not. The system would be based on the analysis of previous data of stock prices. The data to be used for the research are stock prices and related information that publicly available to all either through stock exchanges or social media. The project involves statistical analysis on the acquired data and application of machine learning models to obtain the desired outcome.

b) Research objectives (Maximum 150 words)

Please summarise briefly the objectives of the research.

The objective is to analyse the patterns in the stock price fluctuation, the seasonality effect underneath the stock price pattern and if possible, how the global events affect the stock prices. Based on the analysis, the main objective is to find the right stock for investing or if a particular stock is right for investing.

c) Research location and duration

Location(s)/Population*	Dundalk, Ireland
Research start date	4-Mar-22
Research end date	30-Aug-22
Approximate duration	6 months

* If location/Population other than DkIT campus/population, provide details of the approval to gain access to that location/population as an appendix.

3.3 Participants

		YES	NO	N/A
Do participants fall into any of the following special groups?	Minors (under 18 years of age)			✓
	People with learning or communication difficulties			✓
	Patients			✓
	People in custody			✓
	People engaged in illegal activities (e.g. drug-taking)			✓
Have you given due consideration to the need for satisfactory Garda clearance?				✓

3.4 Sample Details

Approximate number	N/A
Where will participants be recruited from?	N/A
Inclusion Criteria	N/A
Exclusion Criteria	N/A
Will participants be remunerated, and if so in what form?	
N/A	

Justification for proposed sample size and for selecting a specific gender, age, or any other group if this is done in your research.

N/A

3.5 Risk to Participants

- Please describe any risks to participants that may arise due to the research. Such risks could include physical stress, emotional distress, perceived coercion e.g. lecturer interviewing own students. Detail the measures and considerations you have put in place to minimize these risks – N/A
- What will you communicate to participants about any identified risks? Will any information be withheld from them about the research purpose or procedure? If so, please justify this decision. – N/A

3.6 Informed Consent

	YES	NO	N/A
Will you obtain active consent for participation?			✓
Will you describe the main experimental procedures to participants in advance?			✓
Will you inform the participants that their participation is voluntary and may be withdrawn at any point?			✓
If the research is observational, will you ask for their consent to being observed?			✓

With questionnaires, will you give participants the option of omitting questions they do not want to answer?			✓
Will you tell participants that their data will be treated with full confidentiality and that, if published, it will not be identifiable as theirs?			✓
Will the data be anonymous?			✓
Will you debrief participants at the end of their participation?			✓
Will your project involve deliberately misleading participants in any way, or will information be withheld? If you answer yes, give details and justification for doing this below.			✓
N/A			

a) Please outline your approach to ensuring the confidentiality of data (that is, that the data will only be accessible to agreed upon parties and the safeguarding mechanisms you will put in place to achieve this.) You should include details on how and where the data will be stored, and who will have access to it.

N/A

b) Please outline how long the data will be retained for, if it will be destroyed and how it will be destroyed.

1. Storage – N/A
2. Access – N/A
3. Communication – N/A
4. Digital Platform Usage – N/A
5. Data maintenance – N/A

--

Section 4

Researcher I have read and I understand the DkIT Ethics Policy available from:
<https://www.dkit.ie/assets/uploads/documents/Research/Policies/DkIT%20Research%20Ethics%20Policy.pdf>

Signed:  Print Name: **Nishant Verma** Date: **25-Mar-22**
 (Researcher)

Supervisor: Applications for Ethical Approval of Undergraduate projects are forwarded to the Departmental Research Ethics Committee for approval or referral to the School Research Ethics Committee. Applications for Ethical Approval of Postgraduate and Staff projects are sent to the School Research Ethics Committee for Approval.

I have read and approved this form & information:

Signed: *David O'Keeffe* Print Name: David O'Keeffe Date: 25/03/22

(Supervisor/Head of Department/ Research Centre Director/ Head of School)

There is an obligation on the researcher and/or supervisor to bring to the attention of the Departmental/School Research Ethics Committee(s): (a) Any issues with ethical implications not clearly covered by this form (b) Any ethical issues which may arise during the carrying out of the research; (c) Any ethically significant change made to the project after approval.

Section 5 (For office use only)

STATEMENT OF ETHICAL APPROVAL (FOR UNDERGRADUATE PROJECTS ONLY)

This project has been considered using agreed department procedures and is now:

Approved:

☐

Referred to the School Ethics Committee:

☐

Signed: _____ Print Name: _____ Date: _____
(Chair of Departmental Research Ethics Committee/Head of Department)

STATEMENT OF ETHICAL APPROVAL

This project has been considered using agreed School procedures and is now:

Approved:

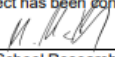
☒

Rejected (further information sought):

☐

Chair of School Research Ethics Committee

This project has been considered by the Ethics Committee and ethical approval is granted.

Signed:  _____ Print Name: Martin Mc Hugh Date: 5-5-22
Chair of School Research Ethics Committee

7. References

- Australian Bureau of Statistics. (2022). *Statistics C* [online].
- Brownlee, J. (2020). *Moving Average Smoothing for Data Preparation and Time Series Forecasting in Python* [online]. Available from: <https://machinelearningmastery.com/moving-average-smoothing-for-time-series-forecasting-python/> [accessed 4 September 2022].
- CFI. (2022). *Exponentially Weighted Moving Average (EWMA)* [online]. Available from: <https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/exponentially-weighted-moving-average-ewma/> [accessed 4 September 2022].
- Contreras, J., Espínola, R., Nogales, F.J. and Conejo, A.J. (2003). ARIMA models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3), pp.1014–1020.
- van Delft, A., Characiejus, V. and Dette, H. (2021). A nonparametric test for stationarity in functional time series. *Statistica Sinica* [online], 31(3), pp.1375–1395.
- Eapen, J., Bein, D. and Verma, A. (2019). Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In: *2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019*.
- Gupta, R., Garg, N. and Singh, S. (2013). Stock market prediction accuracy analysis using kappa measure. In: *Proceedings - 2013 International Conference on Communication Systems and Network Technologies, CSNT 2013*.
- Hassan, M.R. and Nath, B. (2005). Stock market forecasting using Hidden Markov Model: A new approach. In: *Proceedings - 5th International Conference on Intelligent Systems Design and Applications 2005, ISDA '05*.
- Ho, J. (2021). *Time Series Data Analysis — Resample* [online]. Available from: <https://towardsdatascience.com/time-series-data-analysis-resample-1ff2224edec9> [accessed 4 September 2022].
- Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation* [online], 9(8), pp.1735–1780.

How to Grid Search ARIMA Model Hyperparameters with Python. Available from: <https://machinelearningmastery.com/grid-search-arima-hyperparameters-with-python/> [accessed 5 September 2022].

Hyndman, R. and Athanasopoulos, G. (2018a). Autocorrelation. In: *Forecasting: Principles and Practice*. Available from: <https://otexts.com/fpp2/autocorrelation.html> [accessed 1 September 2022].

Hyndman, R. and Athanasopoulos, G. (2018b). The linear model. In: *Forecasting: Principles and Practice*. Available from: <https://otexts.com/fpp2/regression-intro.html> [accessed 2 September 2022].

Hyndman, R. and Athanasopoulos, G. (2018c). Time series regression models. In: *Forecasting: Principles and Practice*. Available from: <https://otexts.com/fpp2/regression.html> [accessed 2 September 2022].

Ince, H. and Trafalis, T.B. (2008). Short term forecasting with support vector machines and application to stock price prediction. *International Journal of General Systems*, 37(6).

Inthachot, M., Boonjing, V. and Intakosum, S. (2016). Artificial Neural Network and Genetic Algorithm Hybrid Intelligence for Predicting Thai Stock Price Index Trend. *Computational Intelligence and Neuroscience*, 2016.

Kaggle. (2022). *Seasonality* [online]. Available from: <https://www.kaggle.com/code/ryanholbrook/seasonality> [accessed 1 September 2022].

Kenton, W. (2022). *Percentage Changes and How to Calculate Them* [online]. Available from: <https://www.investopedia.com/terms/p/percentage-change.asp> [accessed 4 September 2022].

Kim, K.J. and Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Applications*, 19(2).

Krishnan, S. (2021). *Simple Moving Average and Exponentially Weighted Moving Average with Pandas* [online]. Available from: <https://medium.com/codex/simple-moving-average-and-exponentially-weighted-moving-average-with-pandas-57d4a457d363> [accessed 4 September 2022].

- Kursa, M.B. and Rudnicki, W.R. (2010). Feature Selection with the Boruta Package. *JSS Journal of Statistical Software* [online], 36. Available from: <http://www.jstatsoft.org/> [accessed 7 September 2022].
- Lee, M.C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications*, 36(8).
- Lei, L. (2018). Wavelet Neural Network Prediction Method of Stock Price Trend Based on Rough Set Attribute Reduction. *Applied Soft Computing Journal*, 62.
- Lin, X., Yang, Z. and Song, Y. (2009). Short-term stock price prediction based on echo state networks. *Expert Systems with Applications* [online], 36(3), pp.7313–7317.
- Liu, G. and Wang, X. (2019). A new metric for individual stock trend prediction. *Engineering Applications of Artificial Intelligence*, 82.
- Liu, S., Zhang, C. and Ma, J. (2017). CNN-LSTM Neural Network Model for Quantitative Strategy Analysis in Stock Markets. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*.
- McNally, S., Roche, J. and Caton, S. (2018). Predicting the Price of Bitcoin Using Machine Learning. In: *Proceedings - 26th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, PDP 2018*.
- Mustafa, H.I. and Fareed, N.Y. (2020). COVID-19 Cases in Iraq; Forecasting Incidents Using Box-Jenkins ARIMA Model. *Proceedings - 2nd Al-Noor International Conference for Science and Technology, NICST 2020* [online], 28 August 2020, pp.22–26.
- Newbold, P. and Granger, C.W.J. (1974). *Experience with Forecasting Univariate Time Series and the Combination of Forecasts*.
- Ni, L.P., Ni, Z.W. and Gao, Y.Z. (2011). Stock trend prediction based on fractal feature selection and support vector machine. *Expert Systems with Applications*, 38(5).
- Peixero, M. (2019). *The Complete Guide to Time Series Analysis and Forecasting* [online]. Available from: <https://towardsdatascience.com/the-complete-guide-to-time-series-analysis-and-forecasting-70d476bfe775> [accessed 1 September 2022].

- Piramuthu, S. (2004). Evaluating feature selection methods for learning in data mining applications. *European Journal of Operational Research*, 156(2).
- Qiu, M. and Song, Y. (2016). Predicting the direction of stock market index movement using an optimized artificial neural network model. *PLoS ONE*, 11(5).
- Quantivity. (2011). *Why Log Returns* [online]. Available from: <https://quantivity.wordpress.com/2011/02/21/why-log-returns/> [accessed 4 September 2022].
- Rao, P.S., Srinivas, K. and Mohan, A.K. (2020). A Survey on Stock Market Prediction Using Machine Learning Techniques. *Lecture Notes in Electrical Engineering* [online], 601, pp.923–931.
- Rouf, N., Bashir Malik, M., Arif, T., Sharma, S., Singh, S., Aich, S. and Kim, H.-C. (2021). electronics Stock Market Prediction Using Machine Learning Techniques: A Decade Survey on Methodologies, Recent Developments, and Future Directions. , 2021. Available from: <https://doi.org/10.3390/electronics10212717>.
- Saeed, M. (2021). *An Introduction To Recurrent Neural Networks And The Math That Powers Them* [online]. Available from: <https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/> [accessed 3 September 2022].
- Saxena, S. (2021). *Introduction to Long Short Term Memory (LSTM)* [online]. Available from: <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/> [accessed 3 September 2022].
- Shahrour, M.H. and Dekmak, M. (2022). Intelligent Stock Prediction: A Neural Network Approach. *International Journal of Financial Engineering* [online], 20 July 2022. Available from: https://www.researchgate.net/publication/362147561_Intelligent_Stock_Prediction_A_Neural_Network_Approach [accessed 7 September 2022].
- Shih, D.H., Hsu, H.L. and Shih, P.Y. (2019). A study of early warning system in volume burst risk assessment of stock with big data platform. In: *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analytics, ICCCBDA 2019*.

- Shumway, R.H. and Stoffer, D.S. (2019). Correlation and Stationary Time Series. *Time Series: A Data Analysis Approach Using R* [online], 2 July 2019, pp.17–35. Available from: https://www.researchgate.net/publication/334179455_Correlation_and_Stationary_Time_Series [accessed 7 September 2022].
- Simple Stock Trading. (2022). *Why stock market close is so important for traders* [online]. Available from: <https://www.simple-stock-trading.com/why-stock-market-closeis-so-important-for-traders/> [accessed 4 September 2022].
- Sirignano, J. and Cont, R. (2019). Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9).
- Tan, T.Z., Quek, C. and Ng, G.S. (2007). Biological brain-inspired genetic complementary learning for stock market and bank failure prediction. *Computational Intelligence*, 23(2).
- Tretina, K. and Schmidt, J. (2022). *What Is A Stock Market Index?* [online]. Available from: <https://www.forbes.com/advisor/investing/stock-market-index/> [accessed 1 September 2022].
- Upadhyay, A. and Bandyopadhyay, G. (2012). Forecasting Stock Performance in Indian Market using Multinomial Logistic Regression. *Journal of Business Studies Quarterly*, 3(3).
- Wang, X., Phua, P.K.H. and Lin, W. (2003). Stock Market Prediction Using Neural Networks: Does Trading Volume Help in Short-term Prediction?. In: *Proceedings of the International Joint Conference on Neural Networks*.
- Weng, B., Lu, L., Wang, X., Megahed, F.M. and Martinez, W. (2018). Predicting short-term stock prices using ensemble methods and online data sources. *Expert Systems with Applications*, 112.
- Wikipedia. (2013). *File:CRISP-DM Process Diagram.png* [online]. Available from: https://en.wikipedia.org/wiki/File:CRISP-DM_Process_Diagram.png [accessed 1 September 2022].
- Wirth, R. and Hipp, J. (2000). CRISP-DM: Towards a Standard Process Model for Data Mining. , 2000.
- Yaes, R.J. (1989). The Efficient Market Hypothesis. *Science*, 244(4911).