A Project report on

ALGORITHMIC TRADING

A Dissertation submitted in partial fulfilment of the academic requirements for the award of the degree.

Bachelor of Technology

In

Computer Science and Engineering

Submitted by

L. Bharadwaj

18H51A0517

M.M. Prathyush

18H51A0518

G. Vineetkumar

18H51A0596

Under the esteemed guidance of

DR. Sarat Chandra Nayak

Assoc. Professor, Dept. of CSE

Department of Computer Science and Engineering

CMR College of Engineering & Technology

(An Autonomous Institution under UGC & JNTUH, Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NBA.)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2018- 2022

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE ANDENGINEERING

CERTIFICATE

This is to certify that the Major Project phase - 2 report entitled **" Algorithmic Trading "** being submitted by **L. Bharadwaj (18H51A0517), M. M. Prathyush (18H51A0518), and G. Vineetkumar (18H51A0596)** in partial fulfilment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out his/her under my guidance and supervision.

The results embodies in this project report have not been submitted to anyother University or Institute for the award of any Degree.

Dr. Sarat Chandra Nayak
Associate Professor
Dept. of CSE

Dr. K. Vijaya Kumar
Professor and HOD
Dept. of CSE

Submitted for viva voice Examination held on _____

External Examiner

Acknowledgment

With great pleasure I want to take this opportunity to express my heartfelt gratitudeto all the people who helped in making this project work a grand success.

I acknowledge my special thanks to **Dr. Sarat Chandra Nayak (Associate Professor)** Department of Computer Science and Engineering for their consent encouragement, valuable guidance and help in the successful completion of the project work.

I would like to thank **Dr. K. Vijaya Kumar**, Head of the Department of ComputerScience and Engineering, for his moral support throughout the period of my study in CMRCET.

I am highly indebted to **Dr. V. A. Narayana**, Principal CMRCET for giving permission to conduct this project work in a successful and fruitful way.

I would like to thank the Teaching & Non- teaching staff of Department of Computer Science and Engineering for their co-operation

Finally, I express my sincere thanks to **Mr. Ch. Gopal Reddy**, Secretary, CMR Groupof Institutions, for his continuous care. I sincerely acknowledge and thank all those who gave support directly and indirectly in completion of this project work.

L. BHARADWAJ

(18H51A0517)

M. M. PRATHYUSH

(18H51A0518)

G. VINEETKUMAR

(18H51A0596)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Algorithmic trading is a method of executing orders using automated pre-programmed trading instructions for the things such as time and price. This type of trading attempts to boost the speed and computational resources of computers compared to human traders. These encompass a variety of trading strategies, which are based on formulas and results from mathematical finance, and often rely on specialized software.

Stock market is a probabilistic trading where we can gain and have possibility of loss. Investing in stock market trading might be a risky task. So, we can use algorithmic trading which uses the earlier trends of a particular stock and help us predicting investing in the stock. Algorithmic trading using machine learning techniques to increase the probability of profit because it uses the technical analysis of stock, price action strategies, seasonal trends and help us to predict which time is better to invest in stocks. So, we will not fall in debts anymore. We are developing an algorithm using machine learning and deep learning techniques like SMA (Simple Moving Average), Arima, RNN (Recurrent Neural Network) and LSTM (Long Short-Term Memory) for predictions.

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities, and derivatives over virtual platforms. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. This market has given investors the chance of gaining money and having a prosperous life through investing small initial amounts of money, minimal risk compared to the risk of opening new business or the need of high salary career. Stock markets are affected by many factors causing the uncertainty and high volatility in the market. Although humans can take orders and submit them to the market, automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in submitting orders than any human. However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are incorporated and considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analysed.

Time-series prediction is a common technique widely used in many real-world applications such as weather forecasting and financial market prediction. It uses the continuous data in a period to predict the result in the next time unit. Many timeseries prediction algorithms have shown their effectiveness in practice. The most common algorithms now are based on Recurrent Neural Networks (RNN), as well as its special type Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU). Stock market is a typical area that presents time-series data and many researchers' studies on it and proposed various models. In this project, LSTM model is used to predict the stock price.

1.1 MOTIVATION FOR WORK

Stock price prediction is an attractive area where we gain more profits to develop the business. With a successful model for stock prediction, we can gain insight about market behaviour over time. Machine learning technique will be an efficient method to solve the problem of predicting stock movements.

1.2 OBJECTIVE

- To design efficient system for predicting the stock market movement using ML.
- Specifically, use LSTM method to Predict stock price based on the historical data.
- Compare existing methods to show the superiority of LSTM.

1.3 PROBLEM STATEMENT

Time Series forecasting & modelling plays a significant role in data analysis. Time series analysis is a specialized branch of statistics used extensively in fields such as Econometrics & Operation Research. Time Series is being widely used in analytics & data science. Stock prices are volatile in nature and price depends on a range of factors. The main aim of this project is to predict stock prices using Long short-term memory (LSTM).

P/E * EPS = PRICE

If we can predict stocks future P/E and EPS, we will know accurate PRICE

1.4 SCOPE & LIMITATIONS

1.We predict the stock market up to one month.

2.It is hard to predict in the pandemic situations

CHAPTER 2

BACKGROUND WORK

2.0 DOMAIN INTRODUCTION

- ***TRADING***: Trade is a basic economic concept involving the buying and selling of goods and services, with compensation paid by a buyer to a seller, or the exchange of goods or services between parties. Trade can take place within an economy between producers and consumers.
- ***PREDICTION***: “Prediction” refers to the output of an algorithm after it have been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether a customer will churn in 30 days.
- ***STOCK***: Stocks are securities that stand for an ownership share in a company. For companies, issuing stock is a way to raise money to grow and invest in their business. For investors, stocks are a way to grow their money and outpace inflation over time.
- ***STOCK EXCHANGE***: The stock exchange is a virtual market, here buyers and sellers trade in existing securities. It is a market hosted by an institute or any such government body where shares, stocks, debentures, bonds, futures, and options are traded. A stock exchange is a meeting place for buyers and sellers. Examples of stock exchanges are New York Stock Exchange (NYSE), NASDAQ, Tokyo Stock Exchange (JPX), Bombay Stock Exchange (BSE), National Stock Exchange (NSE)
- ***BROKER***: A stockbroker is a financial professional who executes orders in the market on behalf of clients. A stockbroker may also be known as a registered representative (RR) or an investment advisor.

2.1 MACHINE LEARNING ALGORITHMS

Machine learning is a concept that provides systems the ability to learn automatically and improve from experience. Machine learning concepts focuses on the development of computer programs that can access data and use it learn for themselves. The learning process begins with observing the data, such as examples, direct experience, or instruction, to look for patterns in data and make better decisions in the future based on the sample data that we provide. The primary goal is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Machine learning algorithms are of unsupervised and supervised.

2.2 UNSUPERVISED LEARNING ALGORITHM

In unsupervised learning, the training of machine happens using information which is neither classified nor labelled and allowing the algorithm to work on that information without guidance. The goal of machine is to group unsorted information according to similarities, patterns, and differences without any prior training of data. Different from supervised learning, we will not train machine. Machine will find the hidden structure in unlabelled data by it-self.

Unsupervised learning classified into two categories of algorithms:

1 Clustering: A clustering problem is where we want to discover the inherent groupings in the data, such as grouping customers by buying behavior.
2 Association: An association learning problem is where we want to discover rules that describe huge portions of your data, such as people that buy X also tend to buy Y.

2.3 SUPERVISED LEARNING ALGORITHM

In Supervised learning, as the name indicates there is a supervisor as teacher. Learning in which we train the machine using data which is well label is known as Supervised learning. Next, we provide machine with new set of examples (data) so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labelled data. Supervised learning algorithms increases consistently with the data. It is a type of inductive learning.

Supervised learning classified into two categories of algorithms:

1 Classification: A classification problem is where the output variable is a category, such as “Red” or “blue” or “disease” and “no disease.”
2 Regression: A regression problem is where the output variable is a real value, such as “dollars” or “weight.”

2.4 PERFORMANCE MATRIX

We calculate the model performance using the formulas given below:

MSE = $\frac{1}{n}\sum_{i=1}^{n}(yi - pi)^2$

MSE = mean squared error

n = number of data points

Yi = observed values

Pi = predicted values

RMSE =$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(yi - pi)^2}$

RMSD = root mean square deviation

N = no. of missing data points

Yi = observed values

Pi = predicted values

$\boldsymbol{R^2}$ = 1-RSS/TSS

R^2 = coefficient of determination

RSS = sum of squares of residuals

TSS = total sum of squares

2.5 SIMPLE MOVING AVERAGE (SMA) [1]

A simple moving average (SMA) is an arithmetic moving average calculated by adding recent prices and then dividing that figure by the number of time periods in the calculation average.

For example, one could add the closing price of a security for several time periods and then divide this total by that same number of periods. Short-term averages respond quickly to changes in the price of the underlying security, while long-term averages are slower to react. There are other types of moving averages, including the exponential moving average (EMA) and the weighted moving average (WMA).

Formula:

$SMA = (A1+A2+\cdots+An)/n$ eq (1)

Where:

An= The price of an asset

n = The number of total periods

For example, this is how you would calculate the simple moving average of a security with the following closing prices over a 15-day period.

Week One (5 days): 20, 22, 24, 25, 23

Week Two (5 days): 26, 28, 26, 29, 27

Week Three (5 days): 28, 30, 27, 29, 28

A 10-day moving average would average out the closing prices for the first 10 days as the first data point. The next data point would drop the earliest price, add the price on day eleven, and then take the average. Likewise, a 50-day moving average would accumulate enough data to average fifty consecutive days of data on a rolling basis.

A simple moving average smooths out volatility and makes it easier to view the price trend of a security. If the simple moving average points up, this means that the security's price is increasing. If it is pointing down, it means that the security's price is decreasing. The longer the period for the moving average, the smoother the simple moving average.

A shorter-term moving average is more volatile, but its reading is closer to the source data. One of the most popular simple moving averages is the 200-day SMA. However, there is a danger to following the crowd. As the Wall Street Journal explains, since thousands of traders base their strategies around the 200-day SMA, there is a chance that these predictions could become self-fulfilling and limit price growth.

Special Considerations

Analytical Significance Moving averages are an important analytical tool used to identify current price trends and the potential for a change in an established trend. The simplest use of an SMA in technical analysis is using it to quickly determine if an asset is in an uptrend or downtrend. Another popular, albeit slightly more complex, analytical use is to compare a pair of simple moving averages with each covering different time frames. If a shorter-term simple moving average is above a longer-term average, here we except an uptrend. On the other hand, if the long-term average is above a shorter-term average, then a downtrend might be the expected outcome.

Popular Trading Patterns

Two popular trading patterns that use simple moving averages include the death cross and a golden cross. A death cross occurs when the 50-day SMA crosses below the 200-day SMA. This is a bearish signal, indicating that further losses are in store. The golden cross occurs when a short-term SMA breaks above a long-term SMA. Reinforced by high trading volumes, this can signal further gains are in store.

2.5.1 IMPLEMENTATION RESULTS

The platform we used is Google colab. We have implemented SMA for the dates between 08-2018 and 07-2019. We have tested SMA for different stocks datasets like google, amazon, apple, tesla

Fig 2.1: SMA Google stock

Fig 2.2: SMA Amazon stock

Fig 2.3: SMA Apple stock

Fig 2.4: SMA Tesla stock

2.5.2 PERFORMANCE ANALYSIS:

From the above graphs, we have noticed the difference between the predicted values and the actual values is extremely high. So, this model is not preferable for stock prediction

Calculated values of SMA:

Stock Name	MSE	RMSE	R^2
TSLA	34719.06597189837	186.33052882417945	-1.7888224221496438
GOOG	19929.376115409774	141.17144227998017	-0.07731787220738173
AAPL	172.4774882448837	13.133068500730653	-1.2322162402500383
AMZN	61294.70551009148	247.57767571025357	-0.1405148889654626

Table 2.1: SMA performance metrics

MSE and RMSE values are extremely high and R^2 is too low, it denotes that this model is not preferable for satisfactory results

2.6 AUTO REGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA) [2]

This acronym is descriptive, capturing the key aspects of the model itself.

Briefly, they are:

- AR: Autoregression. A model that uses the dependent relationship between an observation and number of lagged observations.
- I: Integrated. The use of differencing of raw observations (e.g., subtracting an observation from an observation at the previous time step) to make the time series stationary.
- MA: Moving Average. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components specifies in the model as a parameter. A standard notation ARIMA (p, d, q) where the parameters are substitutes with integer values to quickly indicate the specific ARIMA model. The parameters of the ARIMA model are as follows:

- p: The number of lag observations included in the model, also called the lag order.
- d: The number of times that the raw observations differenced, also called the degree of differencing.
- q: The size of the moving average window, also called the order of moving average.

ARIMA HIGHLIGHTS

- ARIMA models are known to be robust and efficient in financial time series forecasting
- It constantly outperformed complex structural models in short-term prediction
- But they work only for a particular time series data
- It is a linear model that is it cannot manage nonlinear behavior of stock market

2.6.1 IMPLEMENTATION RESULTS

The platform we used is Google Colab. We have implemented ARIMA for the dates between 11-2021 and 03-2022. We evaluated ARIMA for different stocks datasets like google, amazon, apple, tesla

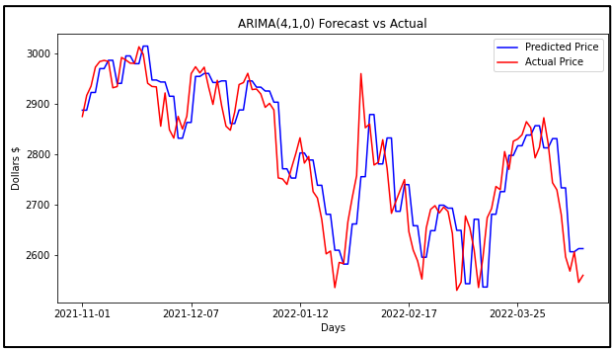

Fig 2.5: ARIMA Google Stock

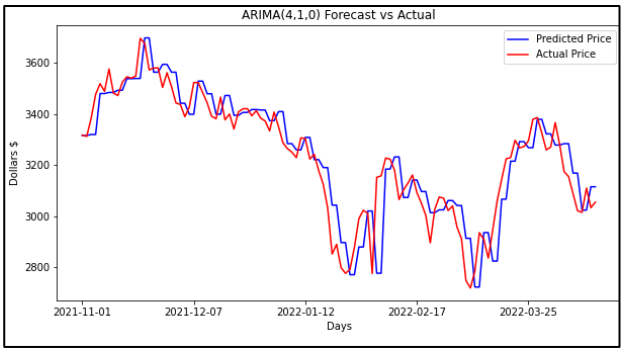

Fig 2.6: ARIMA Amazon Stock

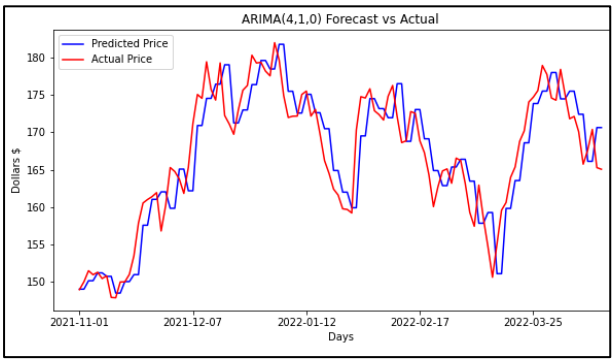

Fig 2.7: ARIMA Apple Stock

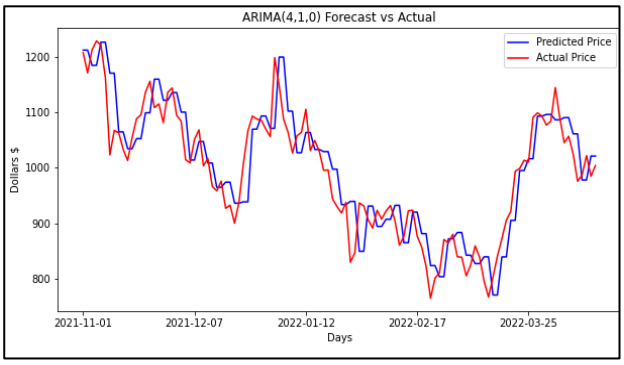

Fig 2.8: ARIMA Tesla Stock

2.6.2 PERFORMANCE ANALYSIS:

From the above graphs, we have noticed the difference between the predicted values and the actual values is low. So, this model is preferable for stock prediction

Calculated values of ARIMA:

Stock	MSE	RMSE	R^2
GOOG	3773.9785847266844	61.43271591527339	0.7959908752231214
TSLA	2398.8965348454335	48.978531366767555	0.80730770665867
AAPL	14.24036165715729	3.7736403719958913	0.815700200173321
AMZN	9812.722059667907	99.0591846305425	0.8174139916784733

Table 2.2: ARIMA performance metrics

MSE and RMSE values are extremely high but R^2 is high, it denotes that this model is slightly preferable for satisfactory results

2.7 RECURRENT NEURAL NETWORKS (RNN) [3]

Recurrent neural networks (RNN) are a class of neural networks that are helpful in modelling sequence data. Derived from feedforward networks, exhibit similar behaviour to how human brains function. Simply put recurrent neural networks produce predictive results in sequential data that other algorithm cannot. RNNs are a powerful and robust type of neural network and belong to the most promising algorithms in use because it is the only one with an internal memory.

Like other deep learning algorithms, recurrent neural networks are old. They were initially created in the 1980's, but only in recent years have we seen their true potential. An increase in computational power along with the massive amounts of data that we now must work with, and the invention of long short-term memory (LSTM) in the 1990s, has really brought RNNs to the foreground.

Because of their internal memory, RNN's can remember important things about the input they received, which allows them to be very precise in predicting what is coming next. Therefore, they are the preferred algorithm for sequential data like time series, speech, text, financial data, audio, video, weather and much more. Recurrent neural networks can form a much deeper understanding of a sequence and its context compared to other algorithms.

In a feed-forward neural network, the information only moves in one direction from the input layer, through the hidden layers, to the output layer. The information moves straight through the network and never touches a node twice.

Feed-forward neural networks have no memory of the input they receive and are bad at predicting what is coming next. Because a feed-forward network only considers the current input, it has no notion of order in time. It simply cannot remember anything about what happened in the past except its training

In a RNN the information cycles through a loop. When it decides, it considers the current input and what it has learned from the inputs it received previously

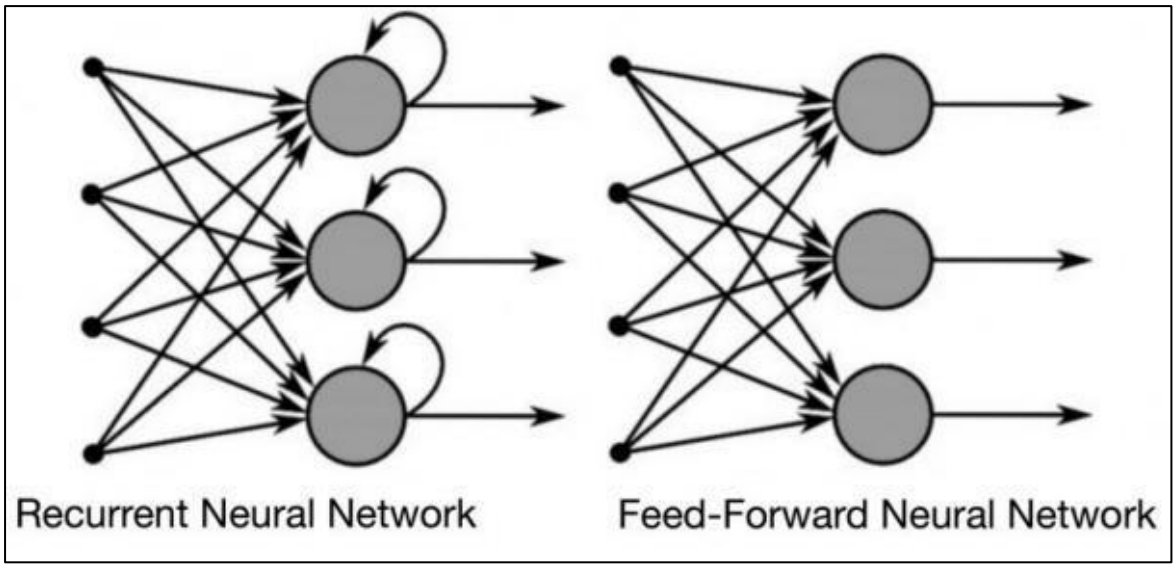

Fig 2.9: Difference between Feed Forward and RNN [3]

Also note that while feed-forward neural networks map one input to one output, RNNs can map one to many, many to many (translation) and many to one (classifying a voice)

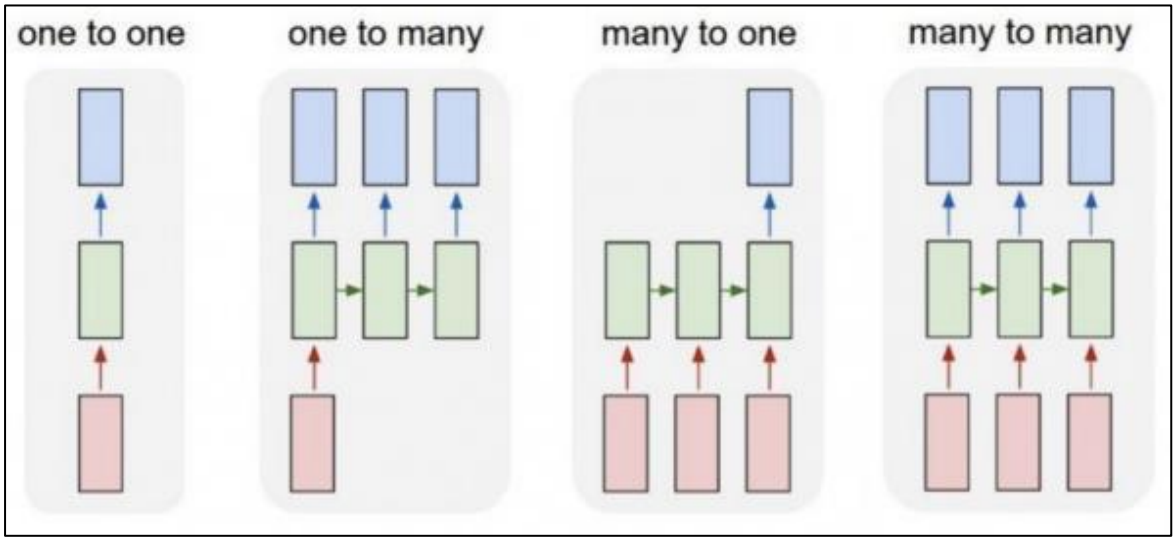

Fig 2.10: RNN mapping different type of inputs to outputs [3]

2.7.1 IMPLEMENTATION RESULTS

The platform we used is Google Colab. We have implemented RNN for the dates between 08-2018 and 07-2019. We have evaluated RNN for different stocks datasets like google, amazon, apple, tesla

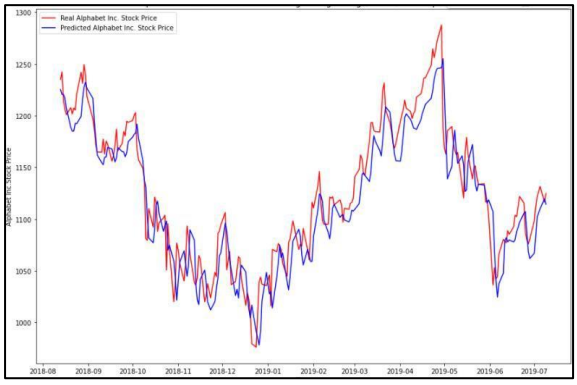

Fig 2.11: RNN Google stock

Fig 2.12: RNN Amazon stock

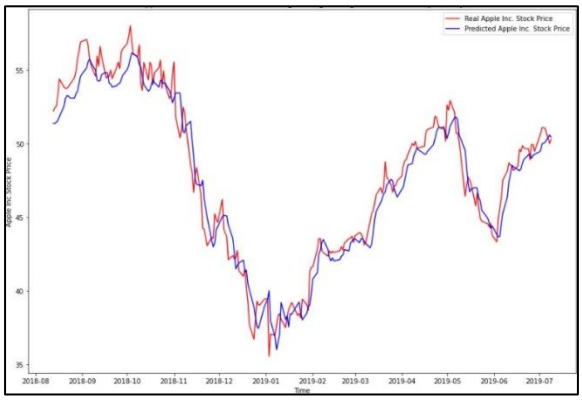

Fig 2.13: RNN Apple stock

Fig 2.14: RNN Tesla stock

2.7.2 PERFORMANCE ANALYSIS:

From the above graphs, we have noticed the difference between the predicted values and the actual values is extremely low. So, this model is preferable for stock prediction

Calculated values of RNN:

Stock	MSE	RMSE	R^2
TSLA	2804.6098714823484	52.95856749839773	0.7747186257457797
GOOG	5022.300541395257	70.86819132301358	0.728510611596199
AAPL	18.03839872750767	4.247163609693847	0.7665457272285929
AMZN	8791.871955275536	93.76498256425762	0.8364090212454195

Table 2.3: RNN performance metrics

we noticed that the MSE and RMSE values are slightly low and R^2 is near to one, it denotes that this model is preferable for satisfactory results

CHAPTER 3
PROPOSED SYSTEM

3.0 LONG SHORT-TERM MEMORY (LSTM) [5]

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network can process not only single data points, but also entire sequences of data.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell.

LSTM networks are well-suited to classifying, processing, and making predictions based on time series data, since there can be lags of unknown duration between notable events in a time series. LSTM deal with the vanishing gradient problem that encounters when training traditional RNNs.

RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem with RNNs is computational, when training a vanilla RNN using backpropagation, the long-term gradients which are back-propagated can "vanish" (that is, they can tend to zero) or "explode" (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged.

3.1 VANISHING GRADIENT PROBLEM

- Ø It occurs when training artificial neural networks with gradient-based learning methods
- Ø In such methods, during each iteration of training each of the neural network's weights receives an update
- Ø In some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value.
- Ø In the worst case, this may completely stop the neural network from further training.

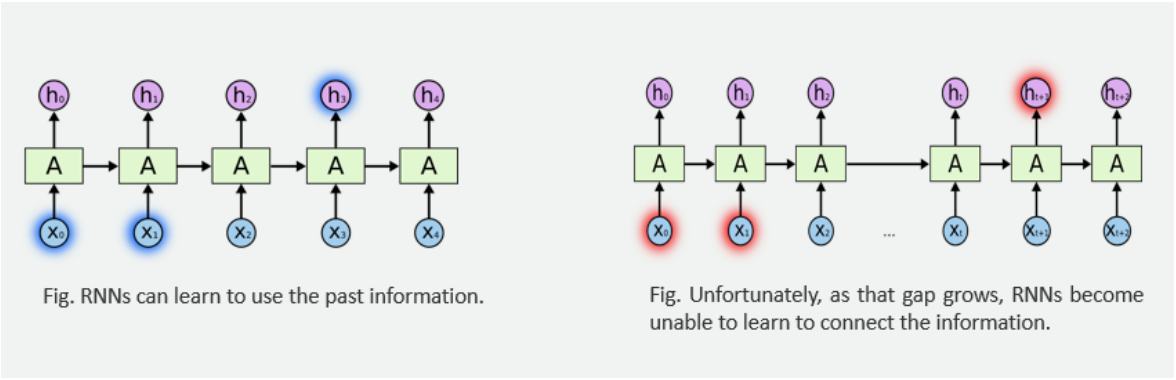

Fig 3.1: The Problem of Long-Term Dependencies [5]

3.2 LONG SHORT-TERM MEMORY (LSTM) HIGHLIGHTS

- LSTM is an advanced type of RNN
- LSTM is proficient in learning about long-term dependencies.
- It solves the vanishing gradient problem
- LSTM is useful on time-series data for classification, processing, and making predictions.
- It does not just use the previous prediction but retains data for longer-term
- LSTMs are good at processing sequences of data such as text, speech, and general time-series

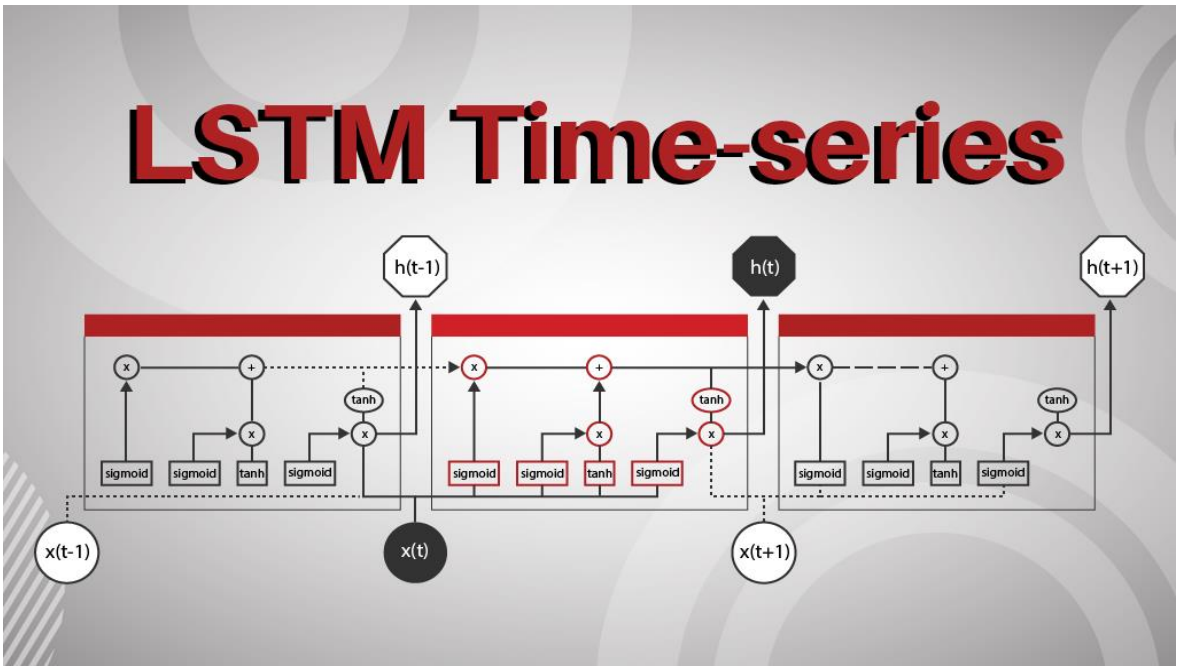

Fig 3.2: LSTM TIME SERIES

3.3 LSTM MODEL

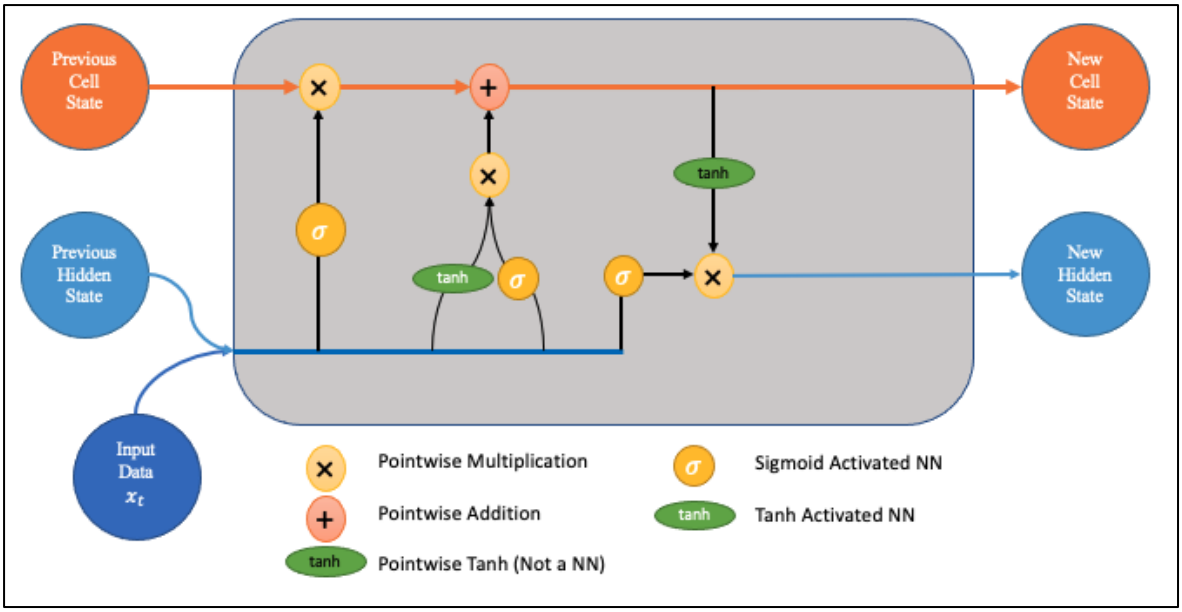

Fig 3.3: LSTM MODEL [5]

The output of an LSTM at a particular point in time is dependent on three things:

- The current long-term memory of the network known as the cell state
- The output at the previous point in time known as the previous hidden state
- The input data at the current time step

LSTMs use a series of ‘gates’ which control the information in a sequence of data.

There are three gates in a typical LSTM.

- forget gate,
- input gate and
- output gate.

3.3.1 FORGET GATE

Here we will decide which bits of the cell state (long term memory of the network) are useful given both the previous hidden state and new input data.

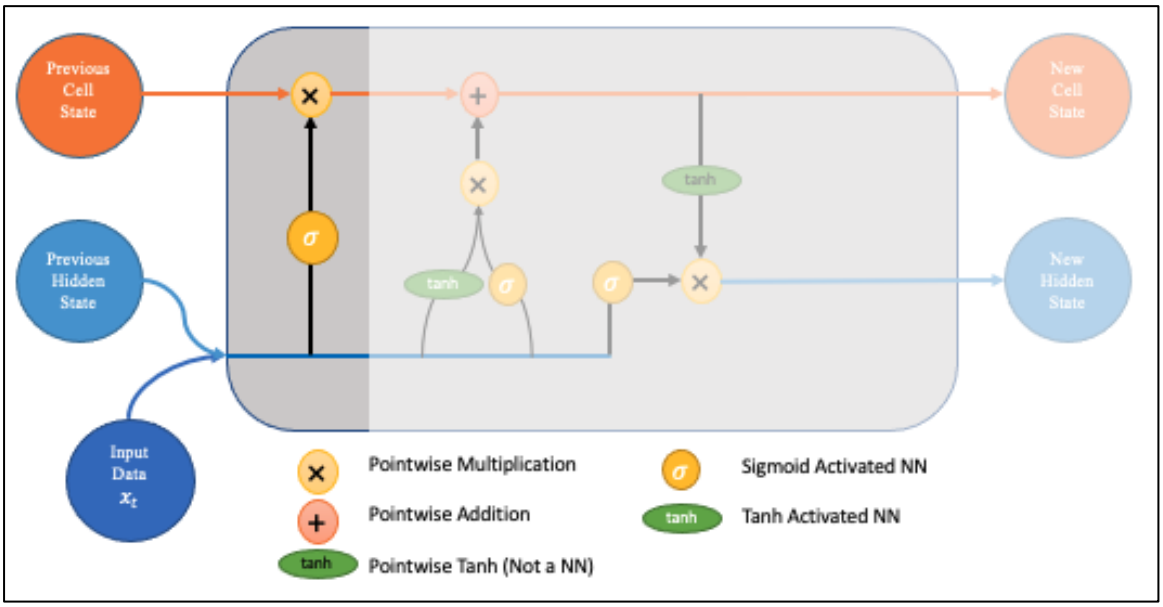

Fig 3.4: FORGET GATE [5]

The previous hidden state and the new input data are fed into a neural network. This network generates a vector where each element is in the interval [0,1] (ensured by using the sigmoid activation). This network (within the forget gate) is trained so that it outputs close to zero when a component of the input is deemed irrelevant and closer to 1 when relevant. It is useful to think of each element of this vector as a sort of filter/sieve which allows more information through as the value gets closer to one.

These output values have sent up and pointwise multiplied with the previous cell state. This pointwise multiplication means that components of the cell state which is irrelevant for the forget gate network will multiplies by a number close to zero and thus will have less influence on the following steps.

In summary, the forget gate decides which pieces of the long-term memory should now be forgotten (have less weight) given the previous hidden state and the new data point in the sequence.

3.3.2 INPUT GATE

The next step involves the new memory network and the input gate. The goal of this step is to determine what added information should add to the networks long-term memory (cell state), given the previous hidden state and new input data.

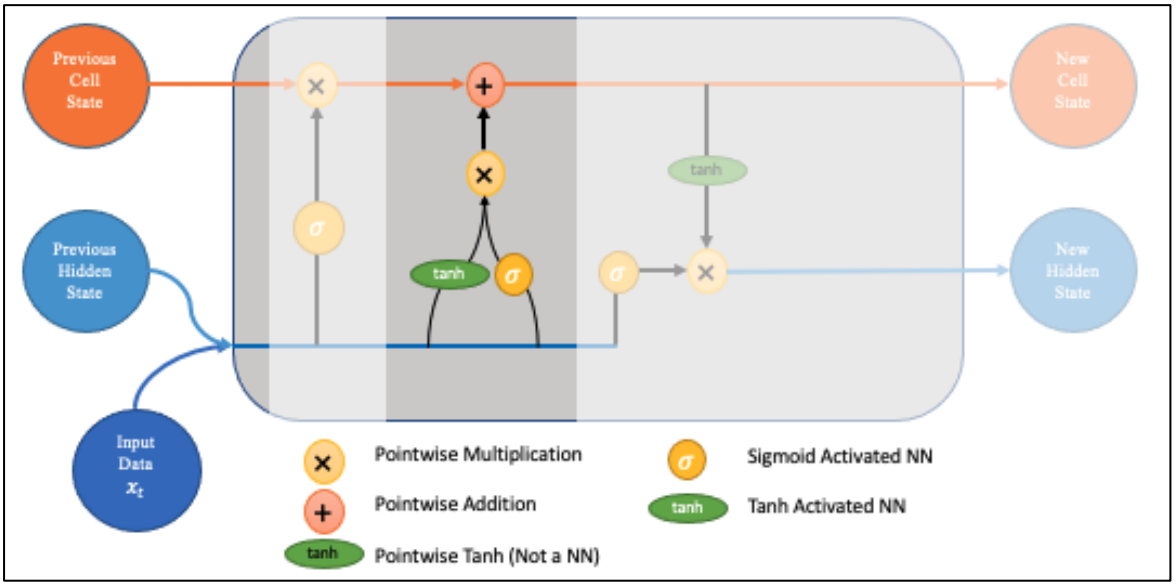

Fig 3.5: INPUT GATE [5]

Both the new memory network and the input gate are neural networks in themselves, and both take the same inputs, the previous hidden state, and the new input data.

1. The new memory network is a tanh activated neural network which has learned how to combine the previous hidden state and new input data to generate a 'new memory update vector'. This vector contains information from the new input data given the context from the previous hidden state. This vector tells us how much to update each component of the long-term memory (cell state) of the network given the new data. We use a tanh here because its values lie in [-1,1] and so can be negative. The possibility of negative values here is necessary if we wish to reduce the impact of a component in the cell state.

2. However, in part 1 above, where we generate the new memory vector, there is a big problem, it does not check if the new input data is even worth remembering. This is where the input gate comes in. The input gate is a sigmoid activated network which acts as a filter, identifying which components of the 'new memory vector' are worth retaining. This network

will output a vector of values in [0,1] (due to the sigmoid activation), allowing it to function as a filter through pointwise multiplication. Like what we saw in the forget gate, an output near zero is telling us we do not want to update that element of the cell state.

3. The output of parts 1 and 2 are pointwise multiplied. This causes the magnitude of added information we decided on in part 2 to be regulated and set to 0 if necessary. The resulting combined vector is then *added* to the cell state, resulting in the long-term memory of the network being updated.

3.3.3 OUTPUT GATE

Now that our updates to the long-term memory of the network are complete, we can move to the last step, the output gate, deciding the new hidden state. We will use three things: the newly updated cell state, the previous hidden state, and the new input data.

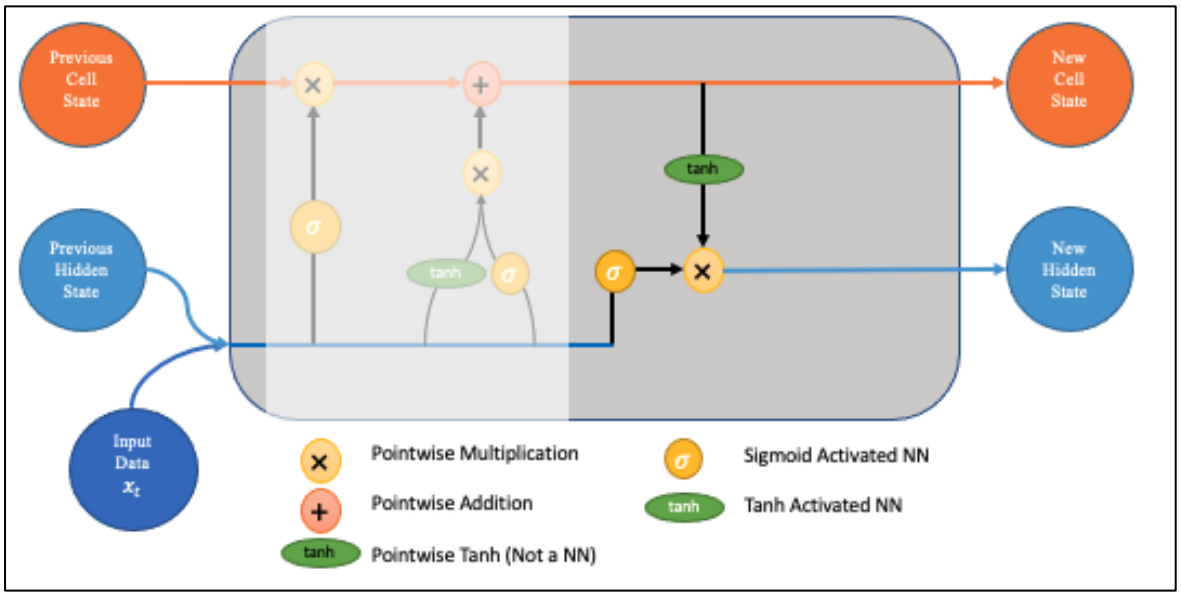

Fig 3.6: OUTPUT GATE [5]

This ensures that only necessary information will be an output (saved to the new hidden state). However, before applying the filter, we pass the cell state through a tanh to force the values into the interval [-1,1].

- Apply the tanh function to the current cell state pointwise to obtain the squished cell state, which now lies in [-1,1].
- Pass the previous hidden state and current input data through the sigmoid activated neural network to obtain the filter vector.

- Apply this filter vector to the squished cell state by pointwise multiplication.
- Output the new hidden state!

3.4 Applications of LSTM [5]

1. Robot control
2. Time series prediction
3. Speech recognition
4. Rhythm learning
5. Music composition
6. Grammar learning
7. Handwriting recognition
8. Human action recognition
9. Sign language translation
10. Protein homology detection
11. Predicting subcellular localization of proteins
12. Time series anomaly detection
13. Prediction tasks in business process management
14. Prediction in medical care pathways
15. Semantic parsing
16. Object co-segmentation
17. Airport passenger management
18. Short-term traffic forecast
19. Drug design
20. Market prediction

CHAPTER 4

DESIGNING

4.0 DESIGN OF LSTM MODEL IMPLEMENTATION

- We reimplemented LSTM model to predict the stock market movement.
- We designed the LSTM model that it predicts the real-world stock instead of dummy data or manipulated data
- We designed our project to evaluate against the different real world stock market datasets.
- The model evaluates against different combinations of parameters to know the better combinations of parameters with which the model can predict the best stock movement
- We designed the project to help the investors to trade according to the stock movement predicted by the LSTM model
- Client can observe the future one-month stock and do better trading

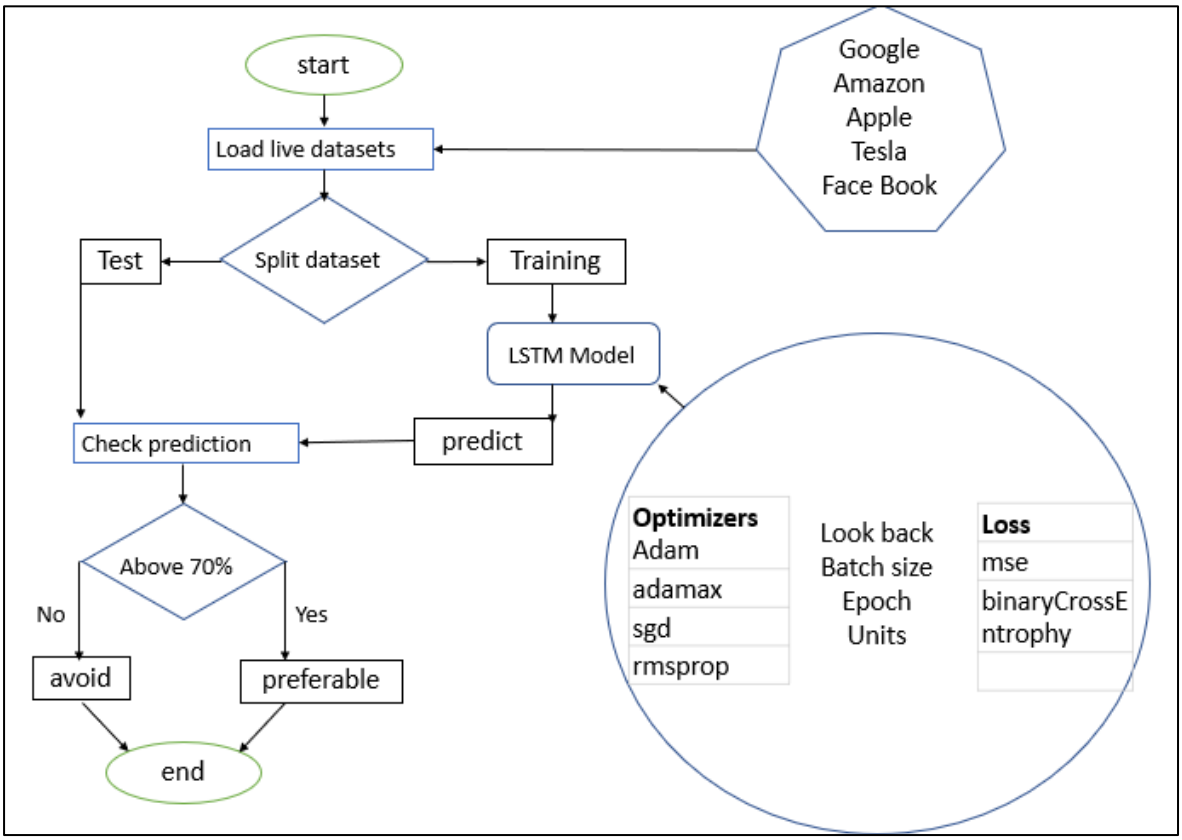

Fig 4.1: Project design

4.1 DESIGN DESCRIPTION

Load Live Datasets:

we are loading real live datasets from yahoo website

```
stock_name = "GOOG"
dataset=pdr.get_data_yahoo(stock_name, start='2015-01-01', end='2018-03-08').reset_index()
dataset.index = dataset['Date']

print("Number of rows and columns:", dataset.shape)
dataset.head(5)
```

Fig 4.2: Snippet of loading live data

we have taken Google, Amazon, Apple, Tesla, Face Book datasets.

Split Dataset:

We have divided the dataset into two datasets

1. Training data set - 70%

2. Test data set - 30%

```
split_percent = 0.70
split = int(split_percent*len(close_data))

close_train = close_data[:split]
close_test = close_data[split:]

date_train = dataset['Date'][:split]
date_test = dataset['Date'][split:]
```

Fig 4.3: Snippet of splitting data

Model parameters

We have considered different parameters to evaluate the LSTM model performance

- Look back
- Batch size
- Epoch
- Units
- **Different optimizers:**
  adam
  adamax
  sgd
  rmsprop
- **Different loss measures:**
  mse
  binaryCrossEntrophy

Model prediction

We get prediction data as an output from the model

Check model performance

Compare the predicted data set with the test data set

Calculate the mode performance using performance matrix formulas

Result

The model with above 70% accurate performance is a preferable model

We should observe future one-month predicted data and do better trading

The model with below 70% accurate performance is a non-preferable model

CHAPTER 5

RESULTS AND DISCUSSION

5.0 REIMPLEMENTED RESULTS OF LSTM

1. LSTM is an existing model, but we have reimplemented the model to increase the performance of the model by changing different combinations of parameters.
2. check the model performance against different datasets.
3. To show how the LSTM model is superior to the other existing models in terms of performance matrix.

5.1 FIND BEST PARAMETERS COMBINATIONS

We will figure out best optimizer, loss measure, batch size, look back, epoch

5.1.1 OPTIMIZERS

1. AdamDelta

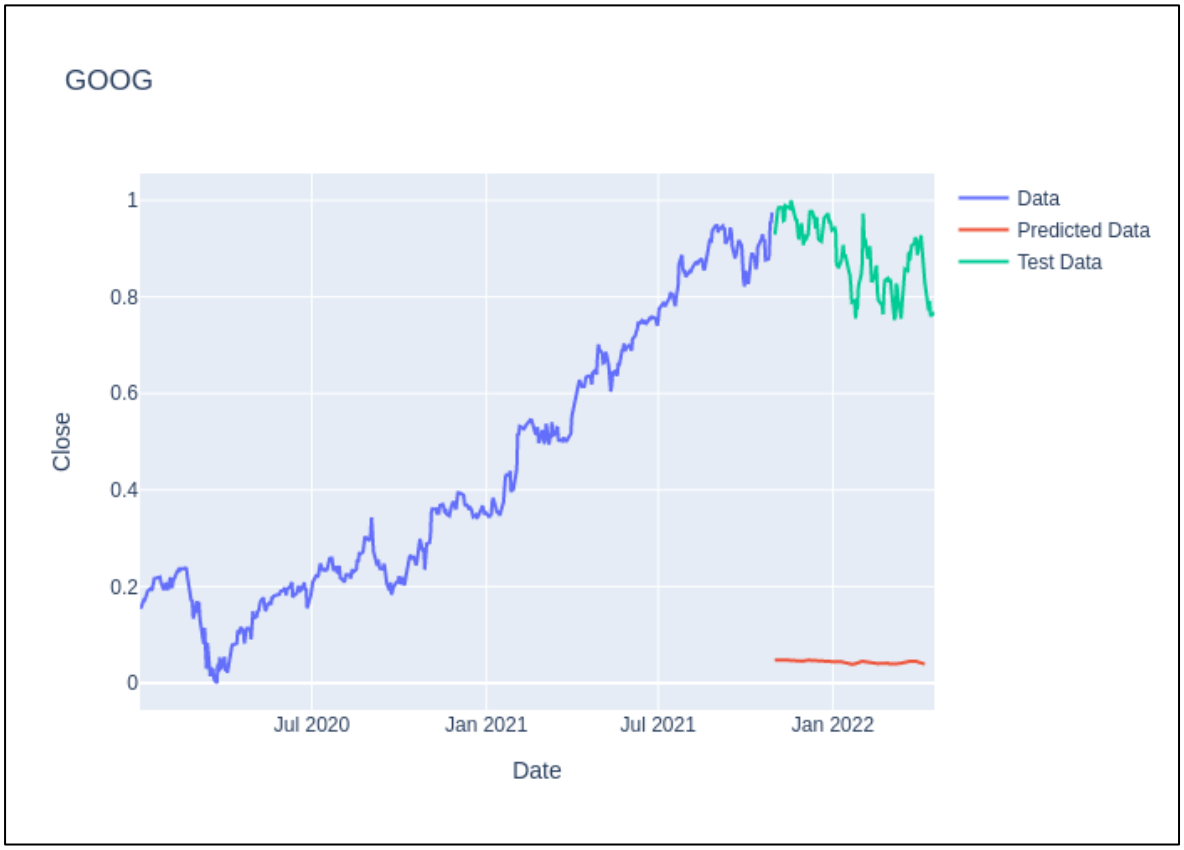

Fig 5.1: AdamDelta optimizer performance for Google dataset

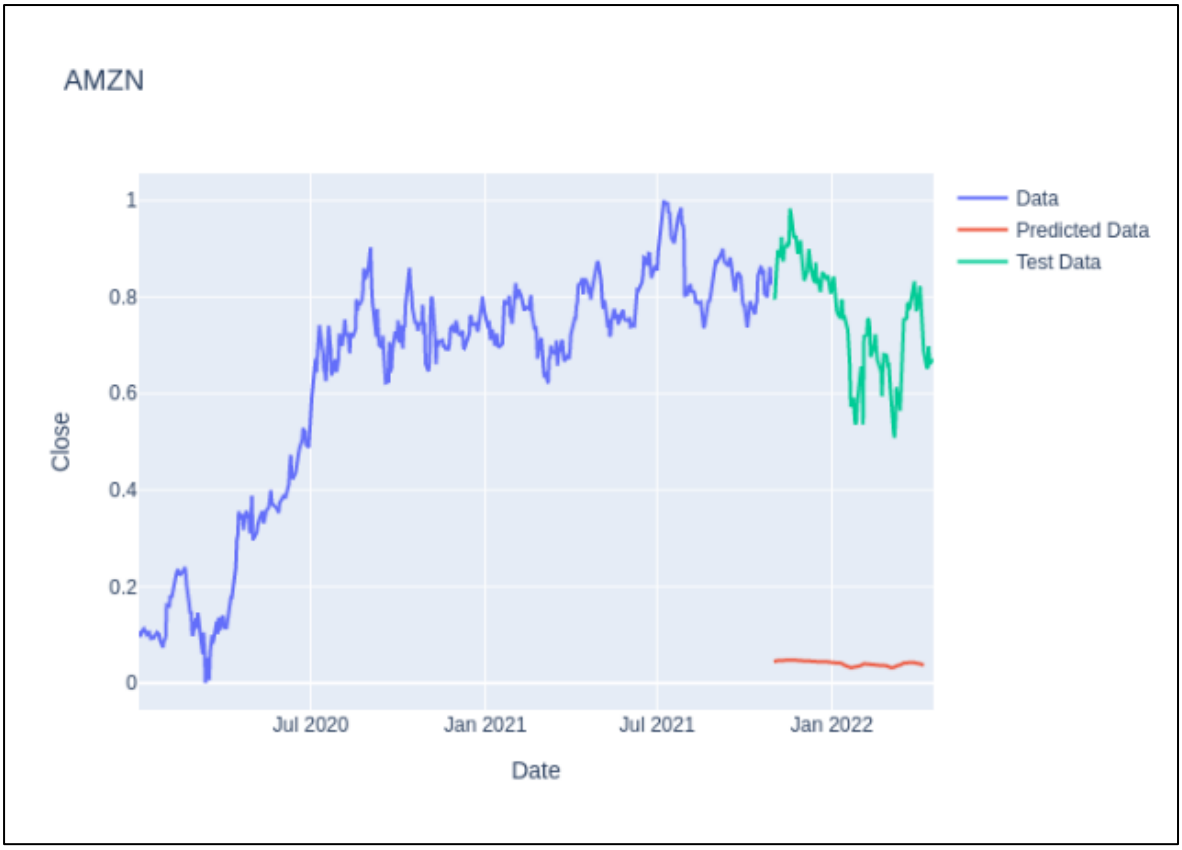

Fig 5.2: AdamDelta optimizer performance for Amazon dataset

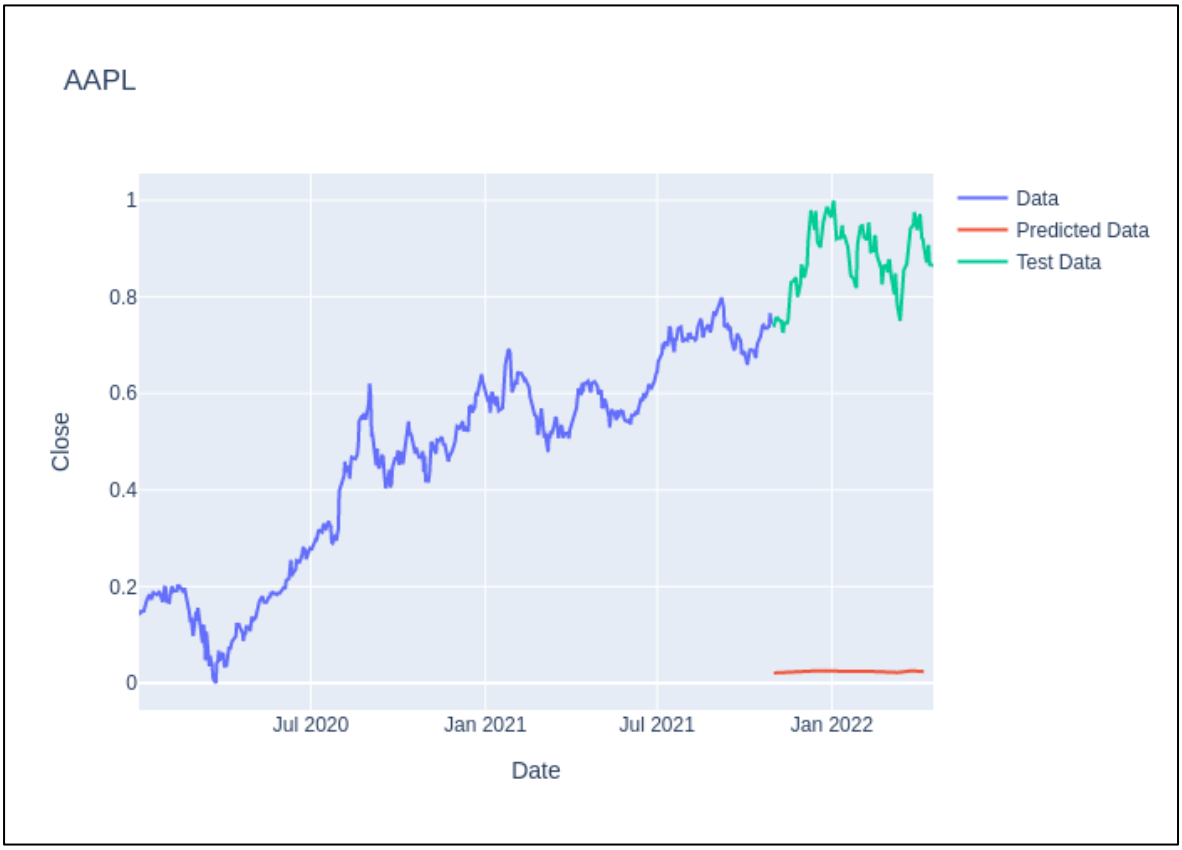

Fig 5.3: AdamDelta optimizer performance for Apple dataset

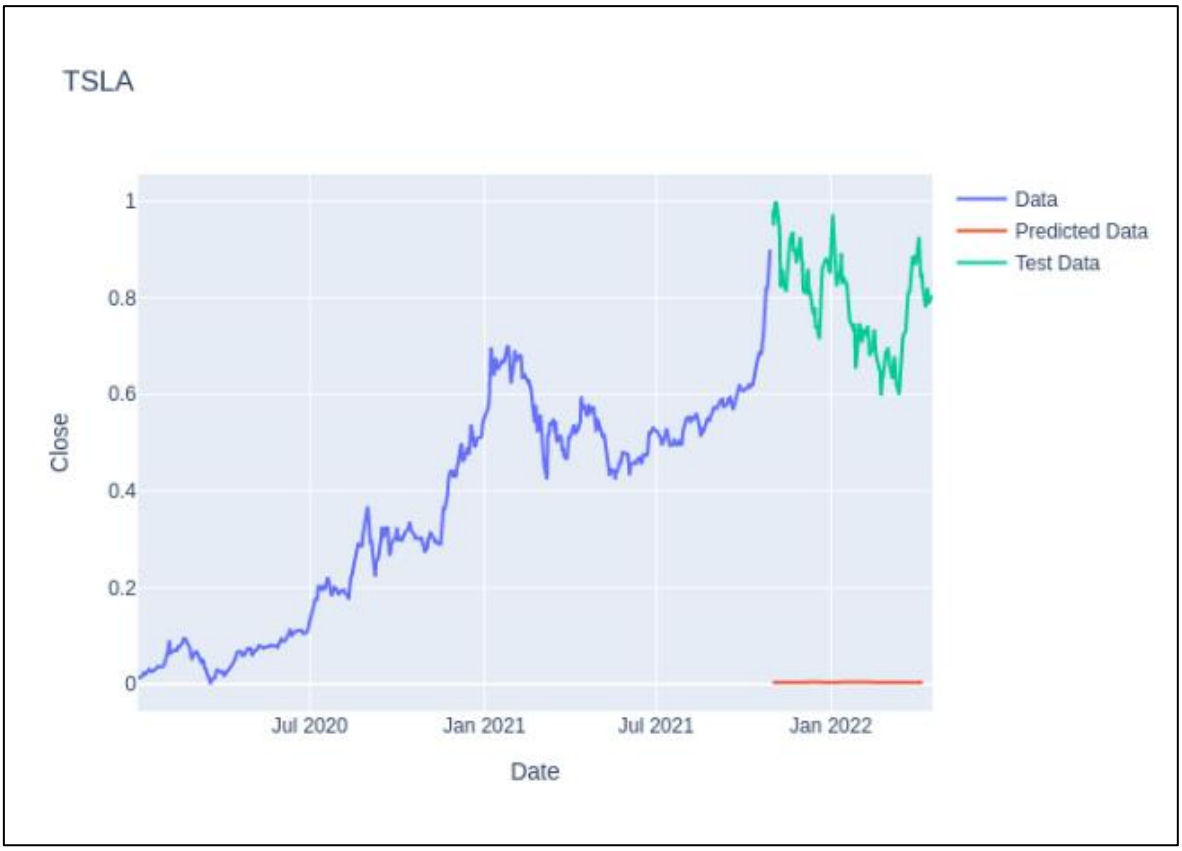

Fig 5.4: AdamDelta optimizer performance for Tesla dataset

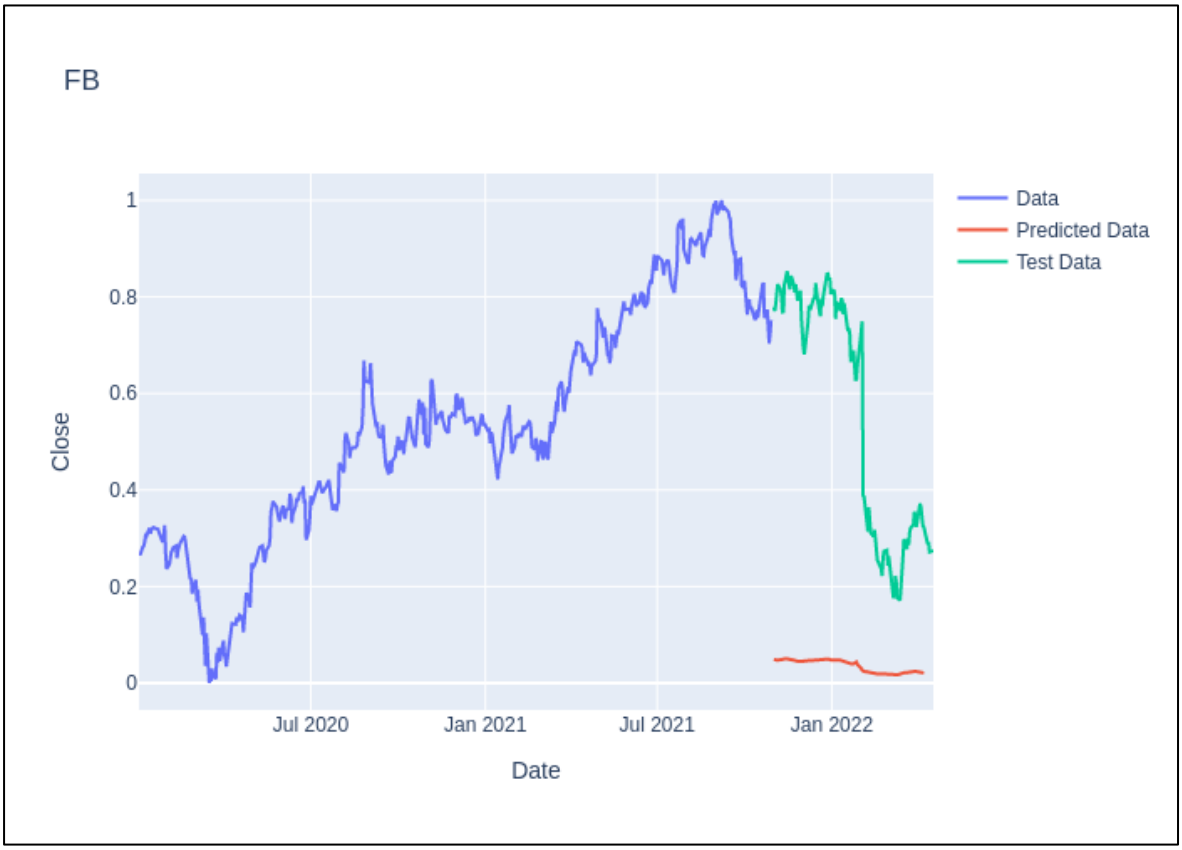

Fig 5.5: AdamDelta optimizer performance for Facebook dataset

Stock	MSE	RMSE	R^2
GOOG	0.7242315188510888	0.85101793098094	-161.85612254035095
TSLA	0.6376065173504686	0.79850267209976	-64.74287934878706
AAPL	0.7381159935887406	0.85913677234113	-144.25799615087402
AMZN	0.5324242087692773	0.72967404282273	-40.27379915035574
FB	0.3424143418992762	0.58516180830542	-4.617525515906984

Table 5.1: AdamDelta optimizer performance metrics

Note: MSE and RMSE values are low but R^2 is too low, it denotes that AdamDelta optimizer is not preferable for satisfactory results

2. Adam optimizer

Fig 5.6: Adam optimizer performance for Google dataset

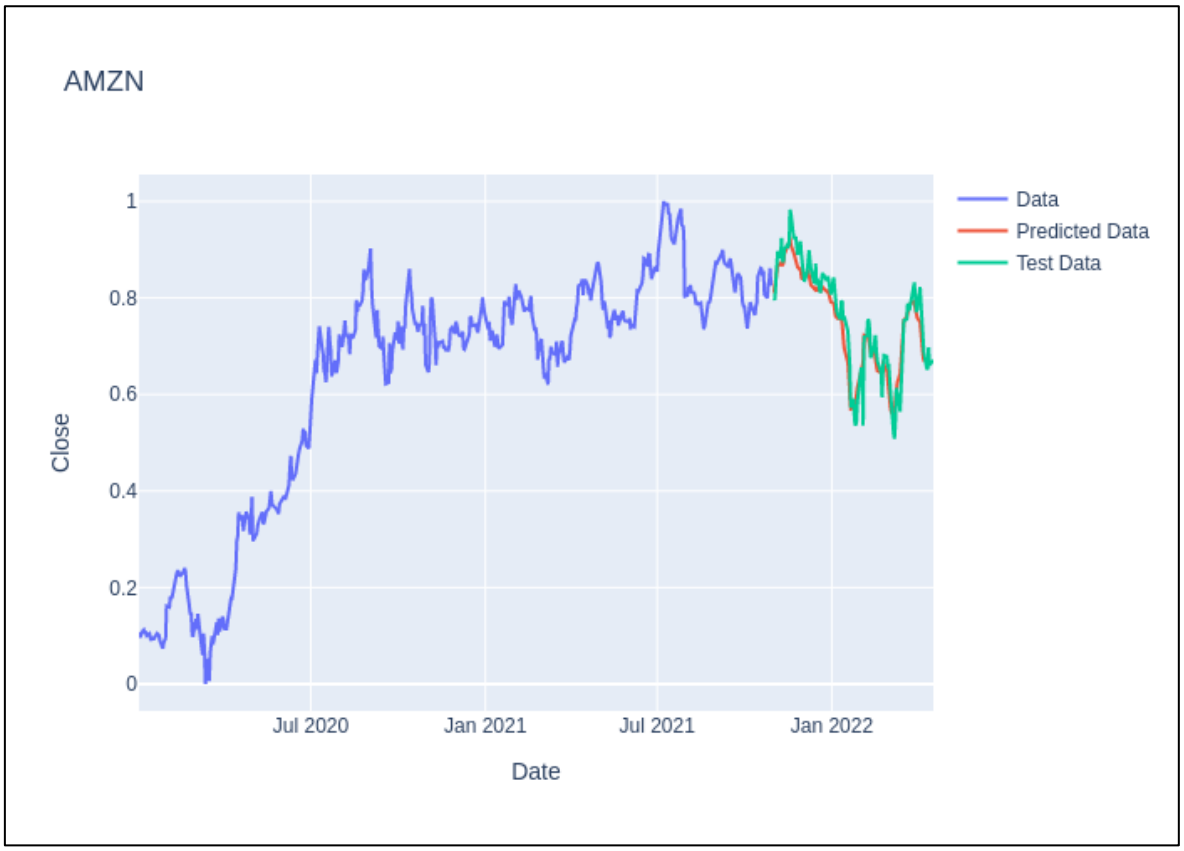

Fig 5.7: Adam optimizer performance for Amazon dataset

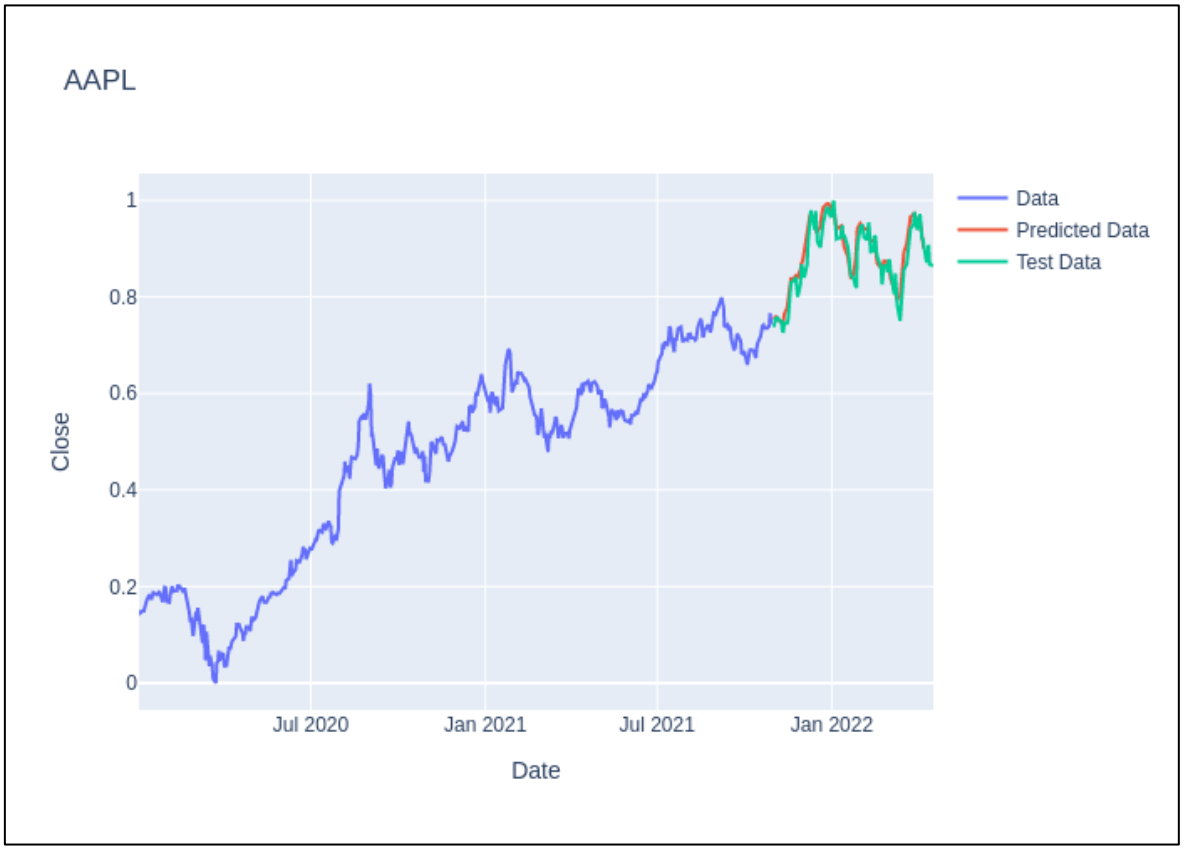

Fig 5.8: Adam optimizer performance for Apple dataset

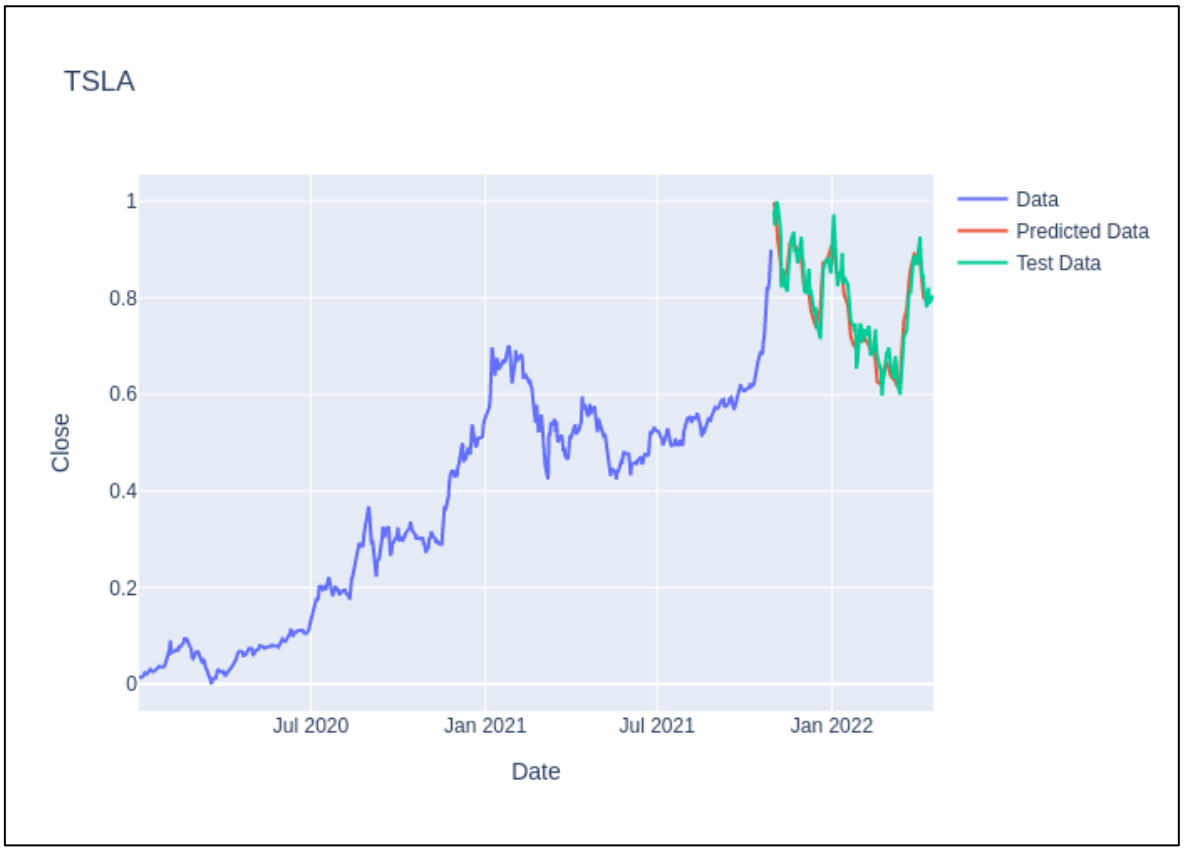

Fig 5.9: Adam optimizer performance for Tesla dataset

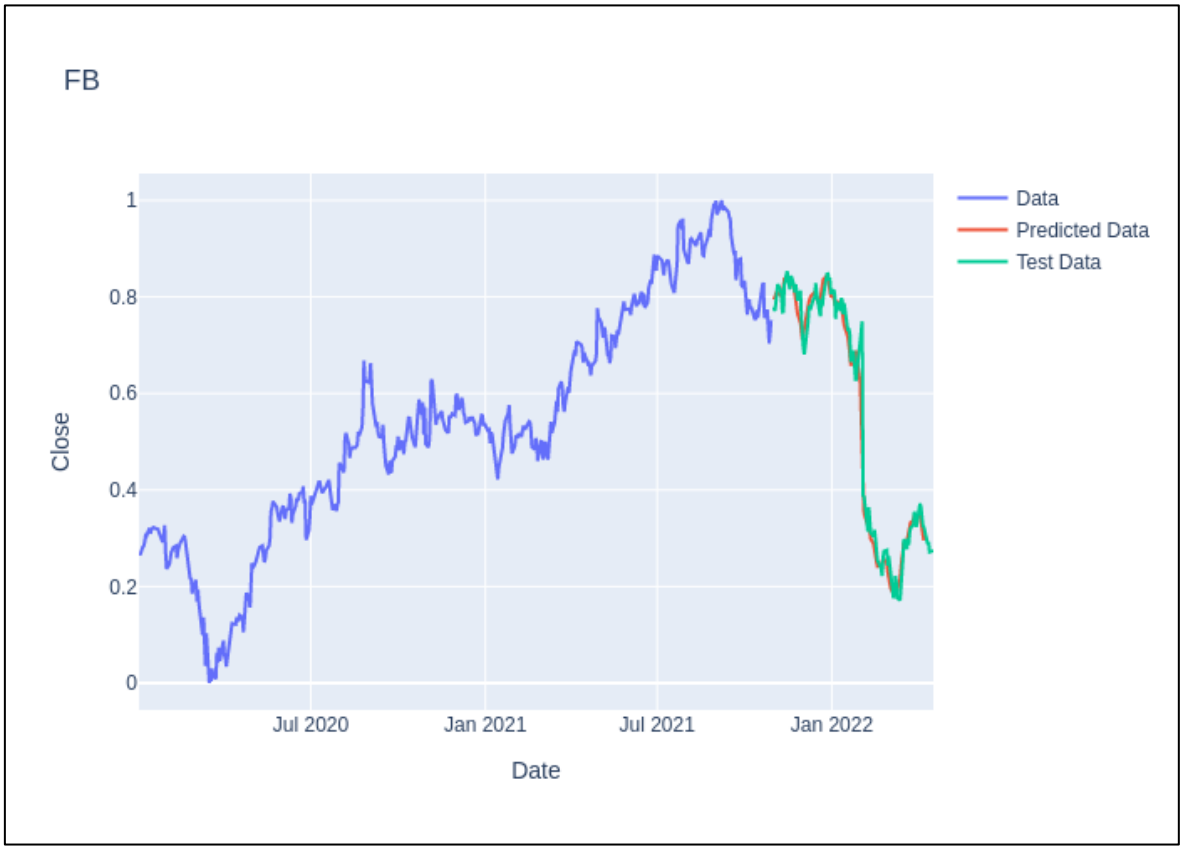

Fig 5.10: Adam optimizer performance for Facebook dataset

Stock	MSE	RMSE	R^2
GOOG	0.00174303628005724	0.0417496859875286	0.60804780981142
TSLA	0.00107213157272670	0.0327434202967055	0.88945373876557
AAPL	0.00059789786488138	0.0244519501243027	0.88233631771980
AMZN	0.00133658889456249	0.0365593885966723	0.89638693982700
FB	0.00163553706856486	0.0404417738058663	0.97316797198413

Table 5.2: Adam optimizer performance metrics

Note: MSE and RMSE values are low and R^2 is values are high, it denotes that Adam optimizer is preferable for satisfactory results

3. RMSProp

Fig 5.11: RMSProp optimizer performance for Google dataset

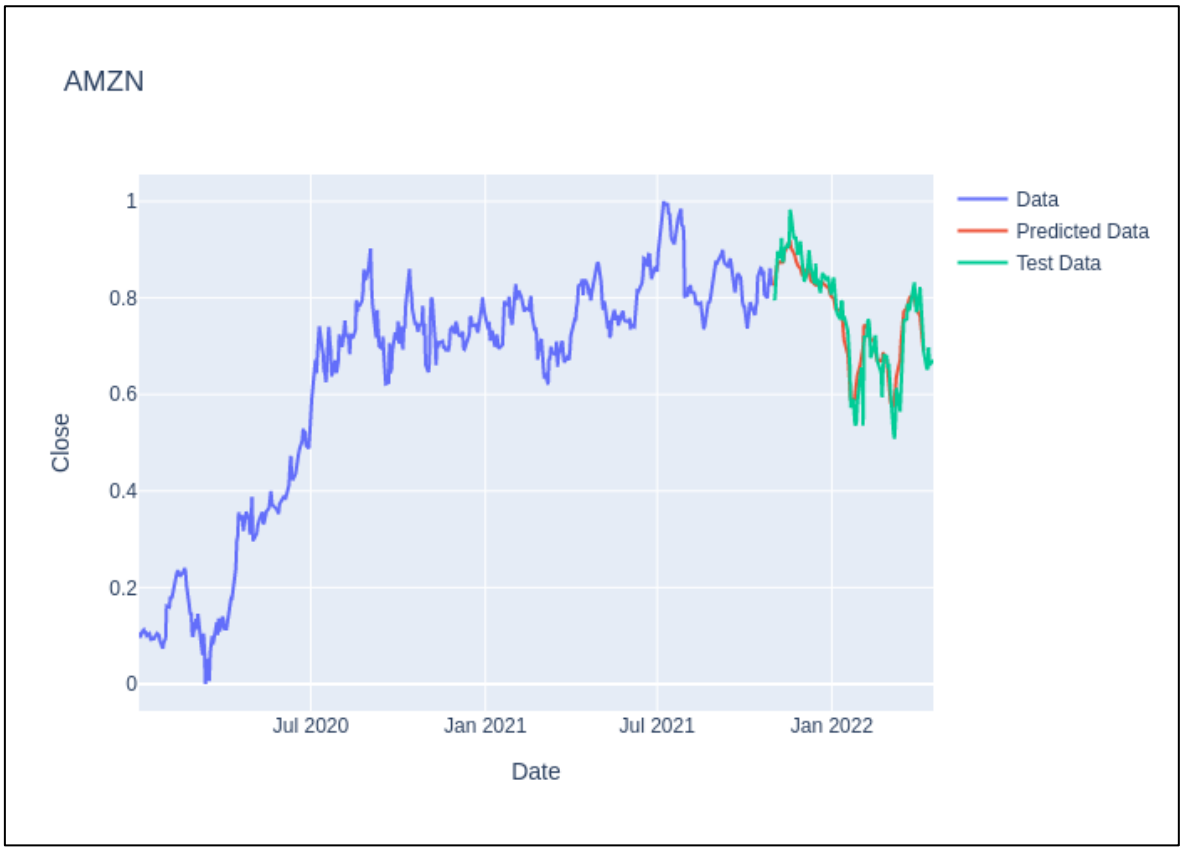

Fig 5.12: RMSProp optimizer performance for Amazon dataset

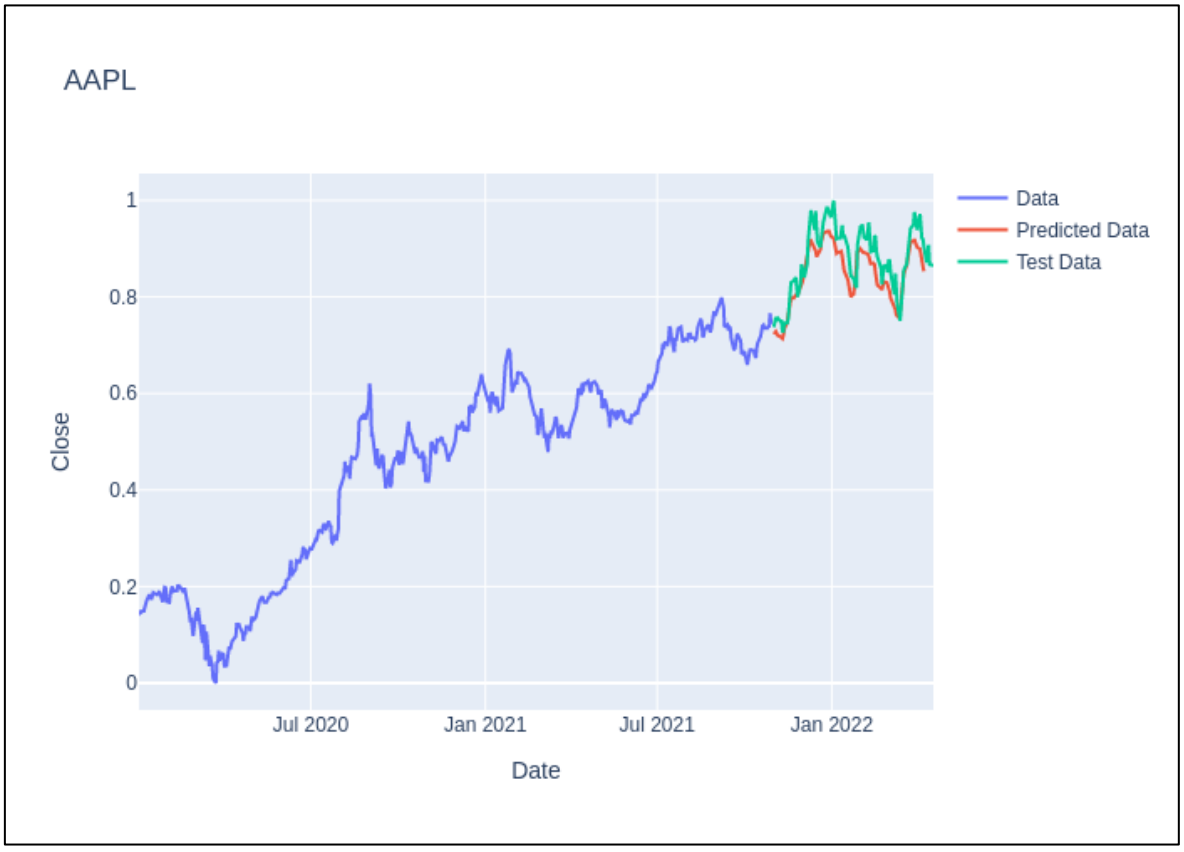

Fig 5.13: RMSProp optimizer performance for Apple dataset

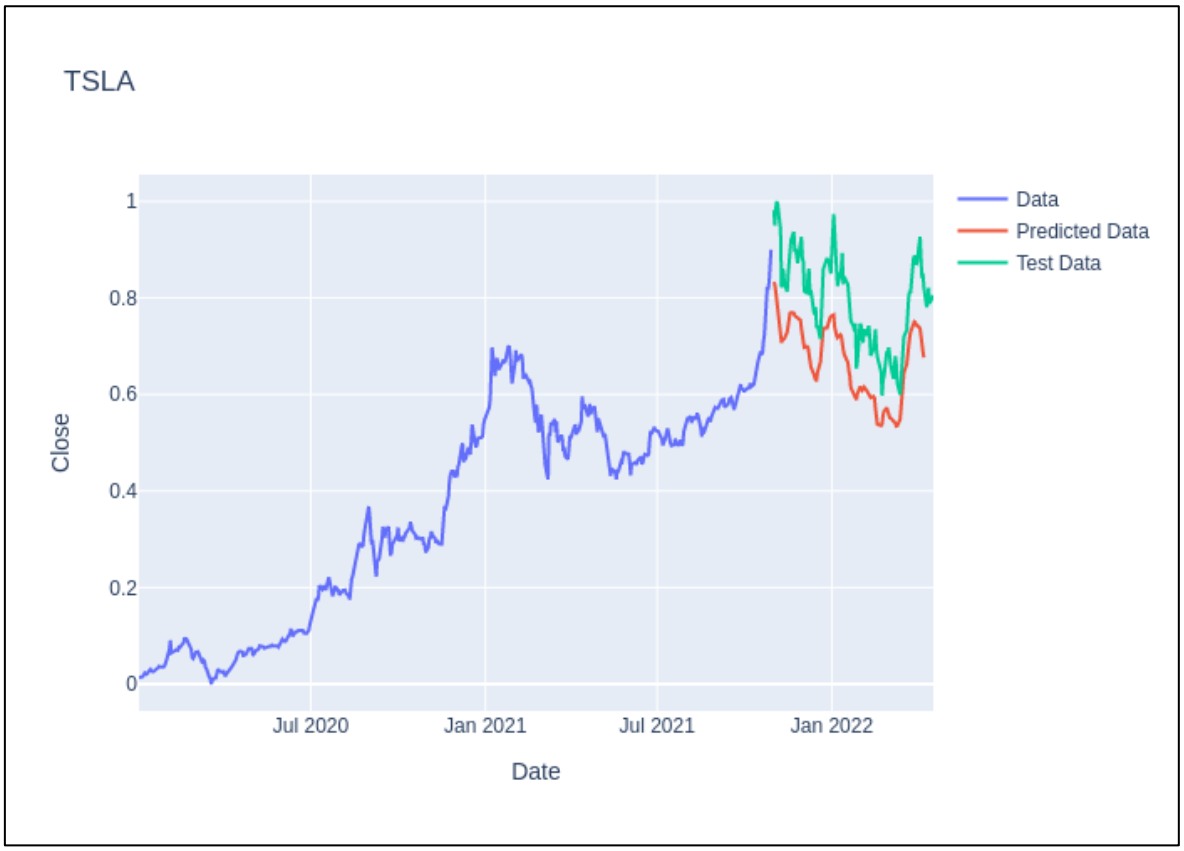

Fig 5.14: RMSProp optimizer performance for Tesla dataset

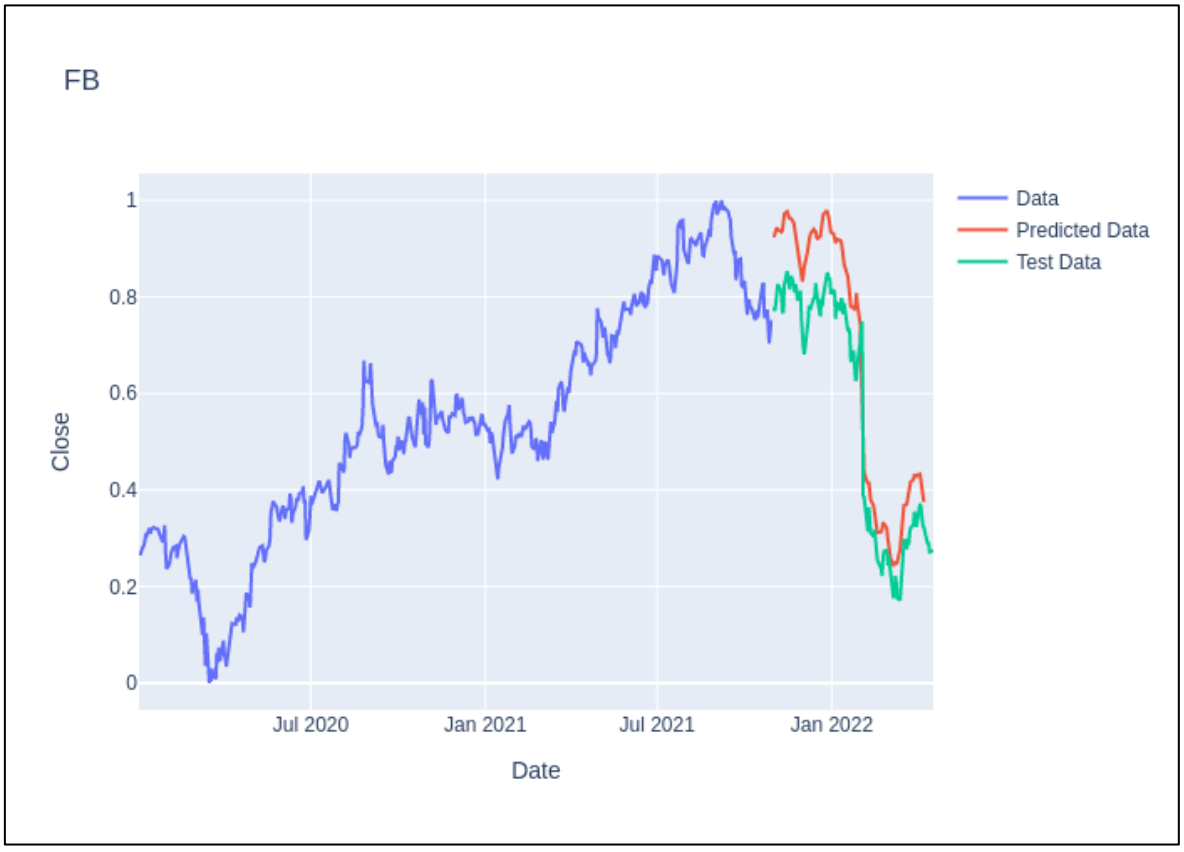

Fig 5.15: RMSProp optimizer performance for Facebook dataset

Stock	MSE	RMSE	R^2
GOOG	0.0008740735446874	0.0295647348151049	0.803449277478668
TSLA	0.0173231656734332	0.1316174975960005	-0.786171815714944
AAPL	0.0016843765279102	0.0410411565128258	0.668522073315030
AMZN	0.0014871973992564	0.0385641984132496	0.884711690898256
FB	0.0126363603316248	0.1124115667163519	0.792692455002430

Table 5.3: RMSProp optimizer performance metrics

Note: MSE and RMSE values are low and all R^2 values are not high, it denotes that **RMSProp** optimizer is not preferable for satisfactory results

4.SGD

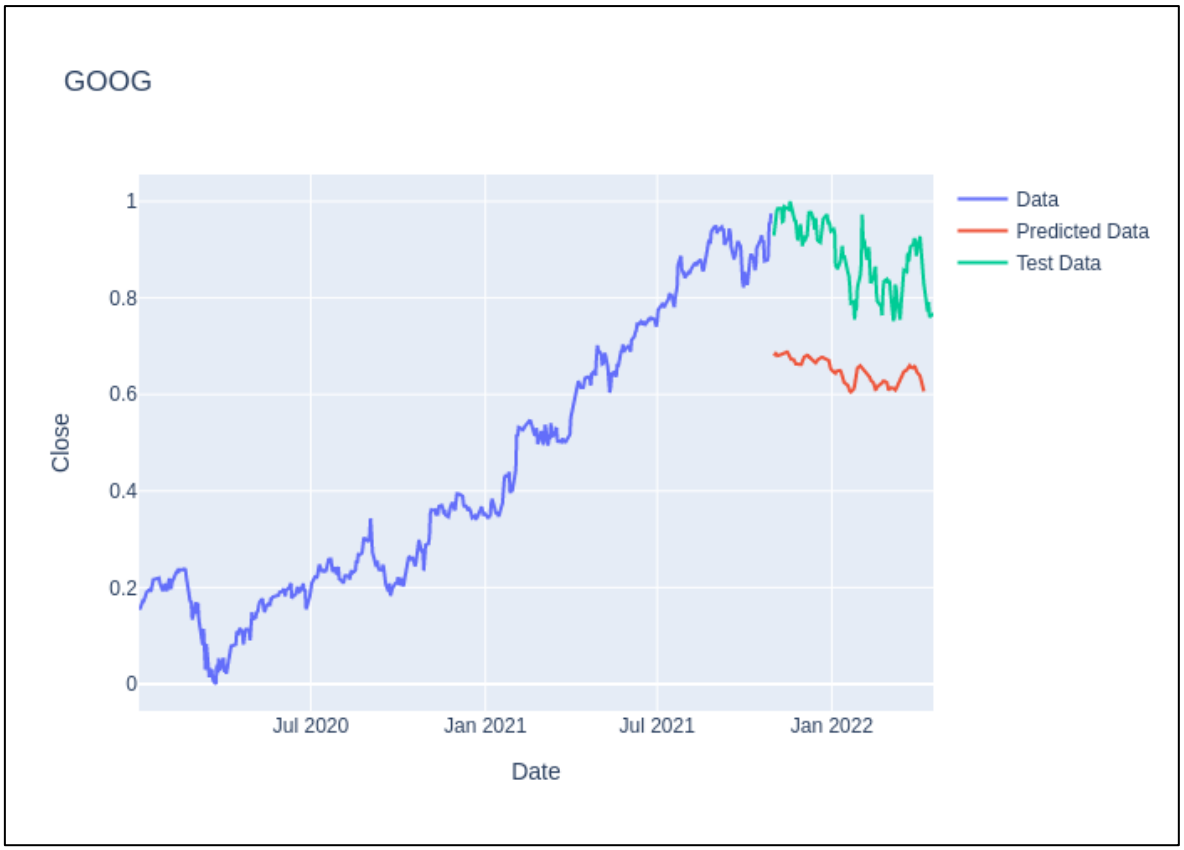

Fig 5.16: SGD optimizer performance for Google dataset

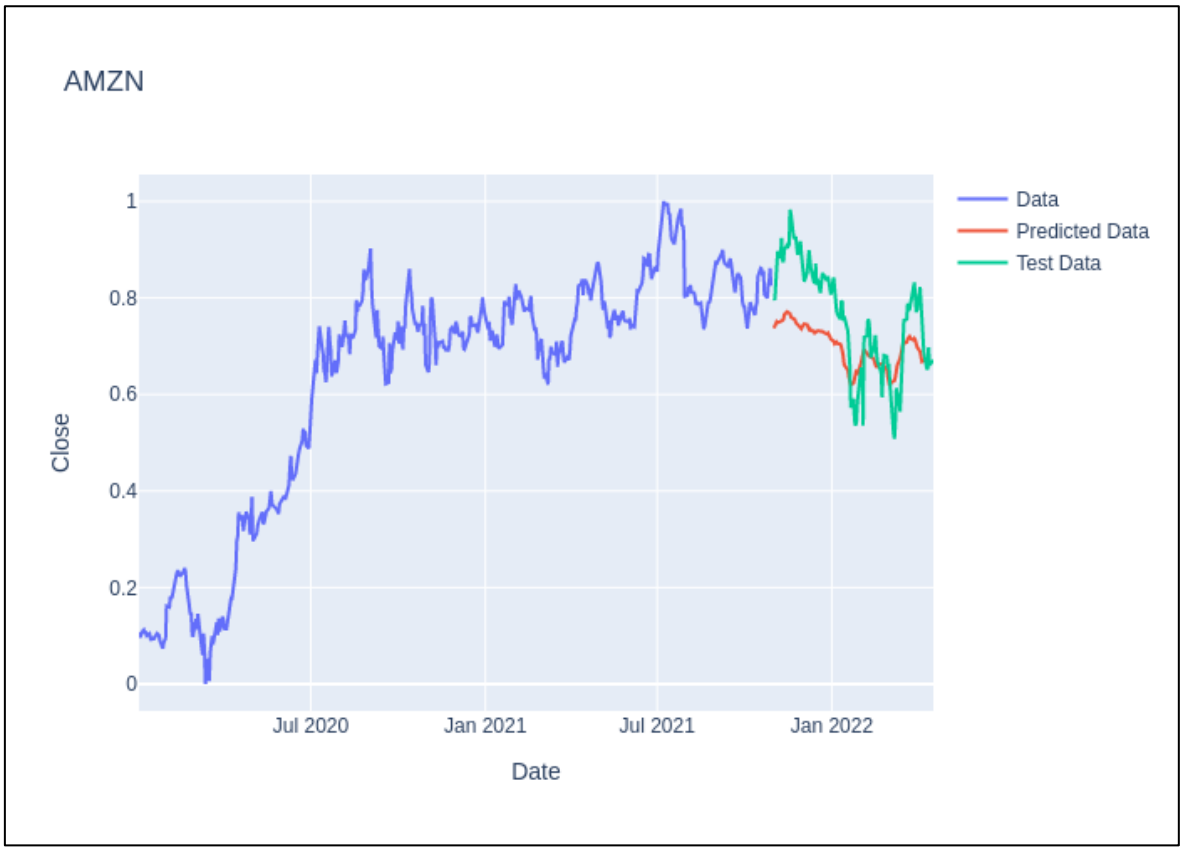

Fig 5.17: SGD optimizer performance for Amazon dataset

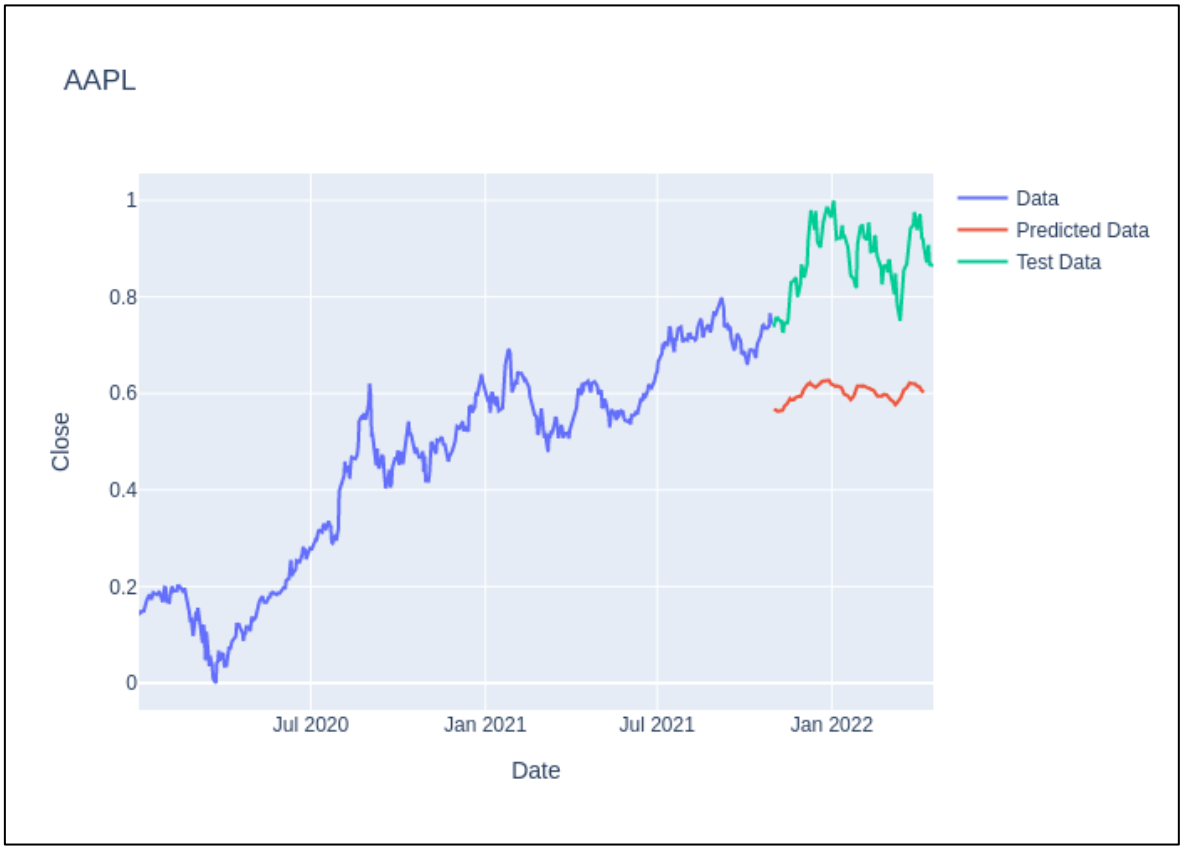

Fig 5.18: SGD optimizer performance for Apple dataset

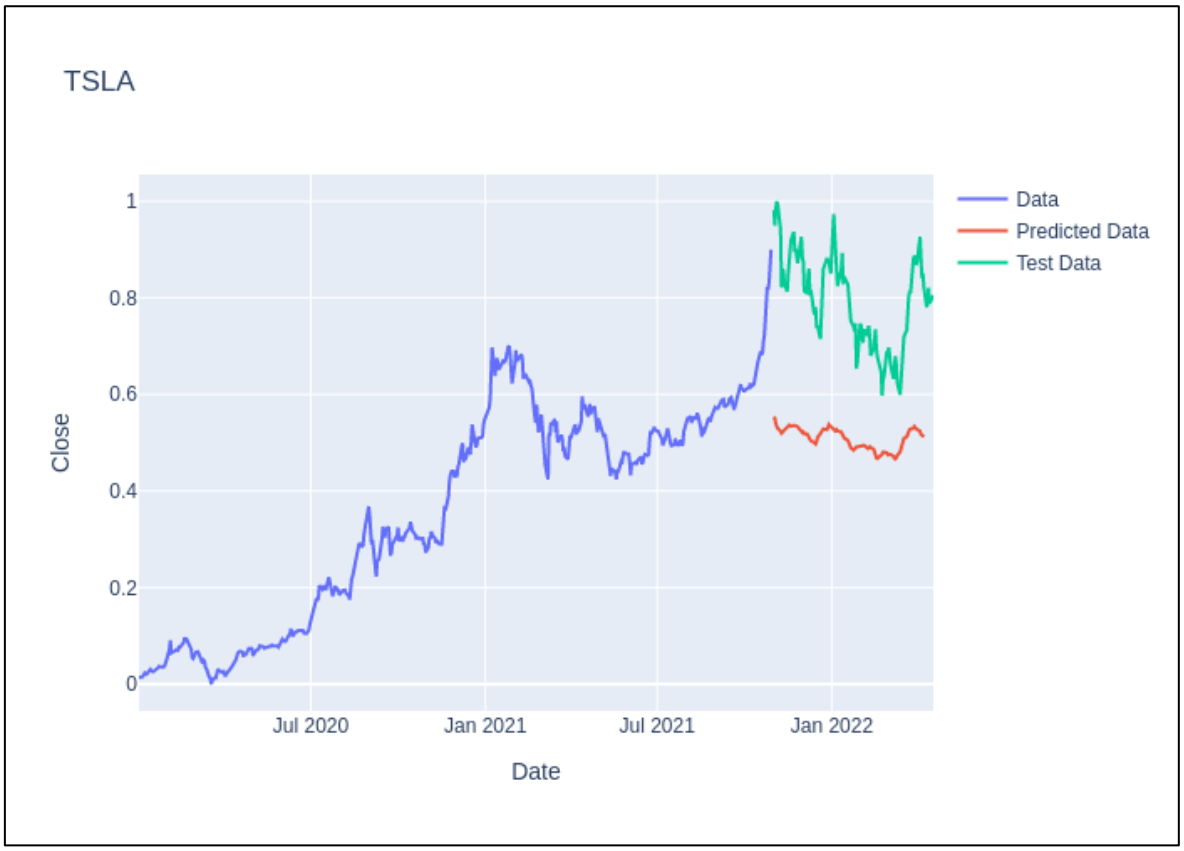

Fig 5.19: SGD optimizer performance for Tesla dataset

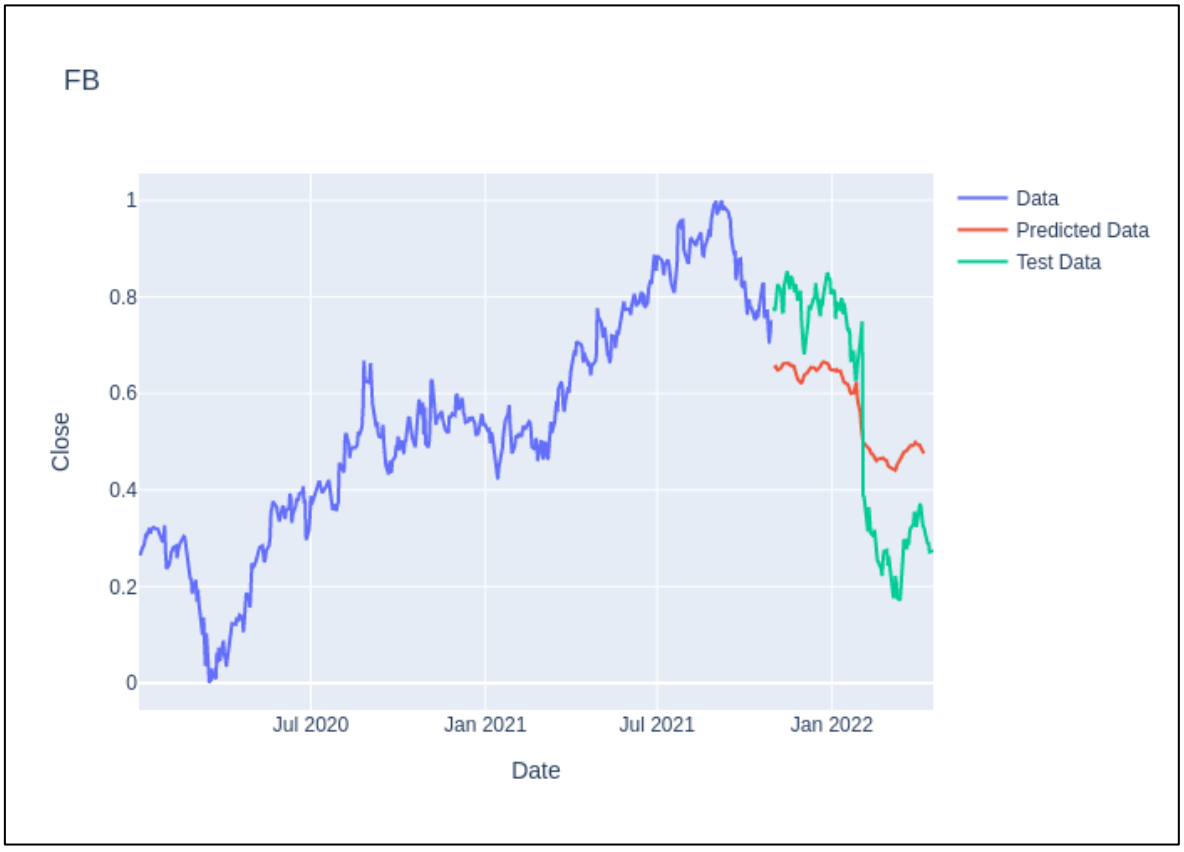

Fig 5.20: SGD optimizer performance for Facebook dataset

Stock	MSE	RMSE	R^2
GOOG	0.06160261132604412	0.2481987335302985	-12.85242447723768
TSLA	0.08883371057436526	0.2980498457881924	-8.159542378368496
AAPL	0.08054911989565998	0.2838117684234746	-14.85171416063263
AMZN	0.00996510854915196	0.0998253903030284	0.227499647846733
FB	0.02732996593033029	0.1653177725785412	0.551634490217527

Table 5.4: SGD optimizer performance metrics

Note: MSE and RMSE values are low but R^2 values are not high, it denotes that **SGD** optimizer is not preferable for satisfactory results

We have noticed that Adam optimizer is the best optimizer

5.1.2 LOSS MEASURES

1. Binary Cross Entropy

Fig 5.21: Binary Cross Entropy performance for Google dataset

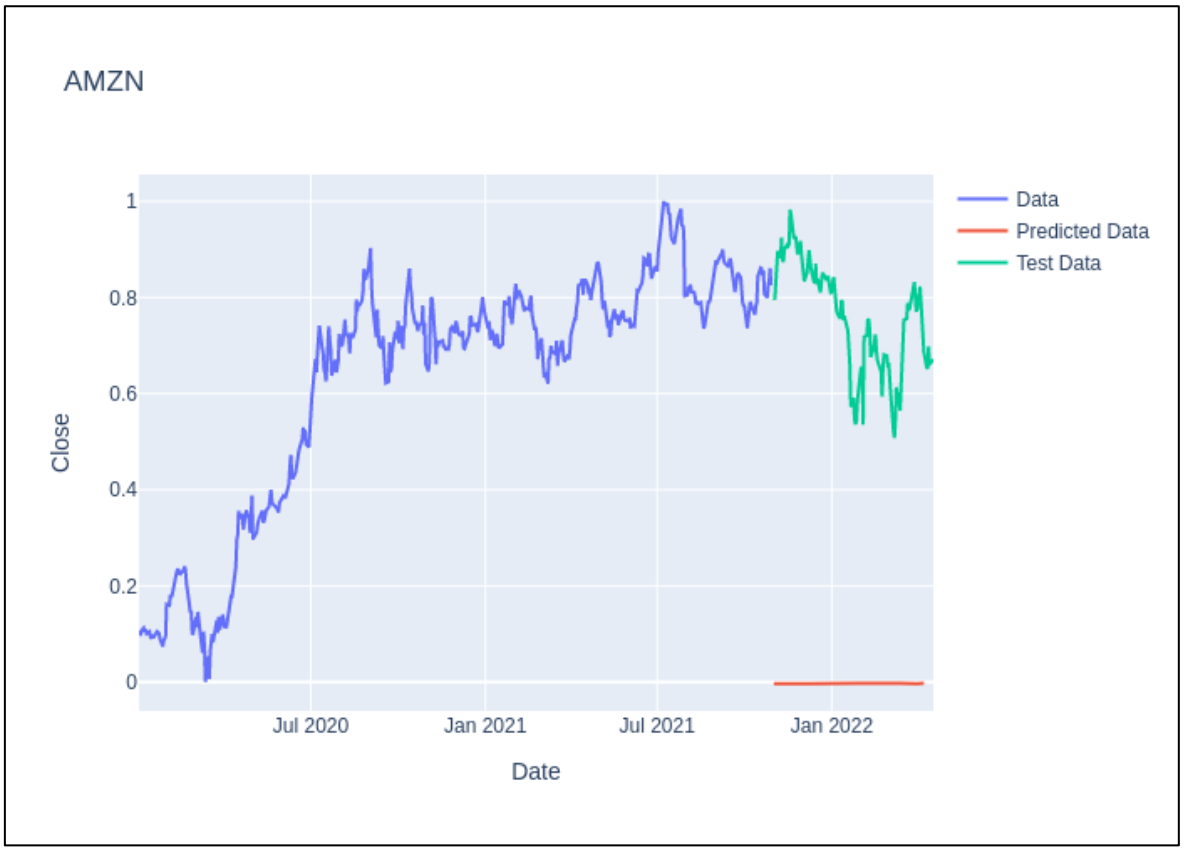

Fig 5.22: Binary Cross Entropy performance for Amazon dataset

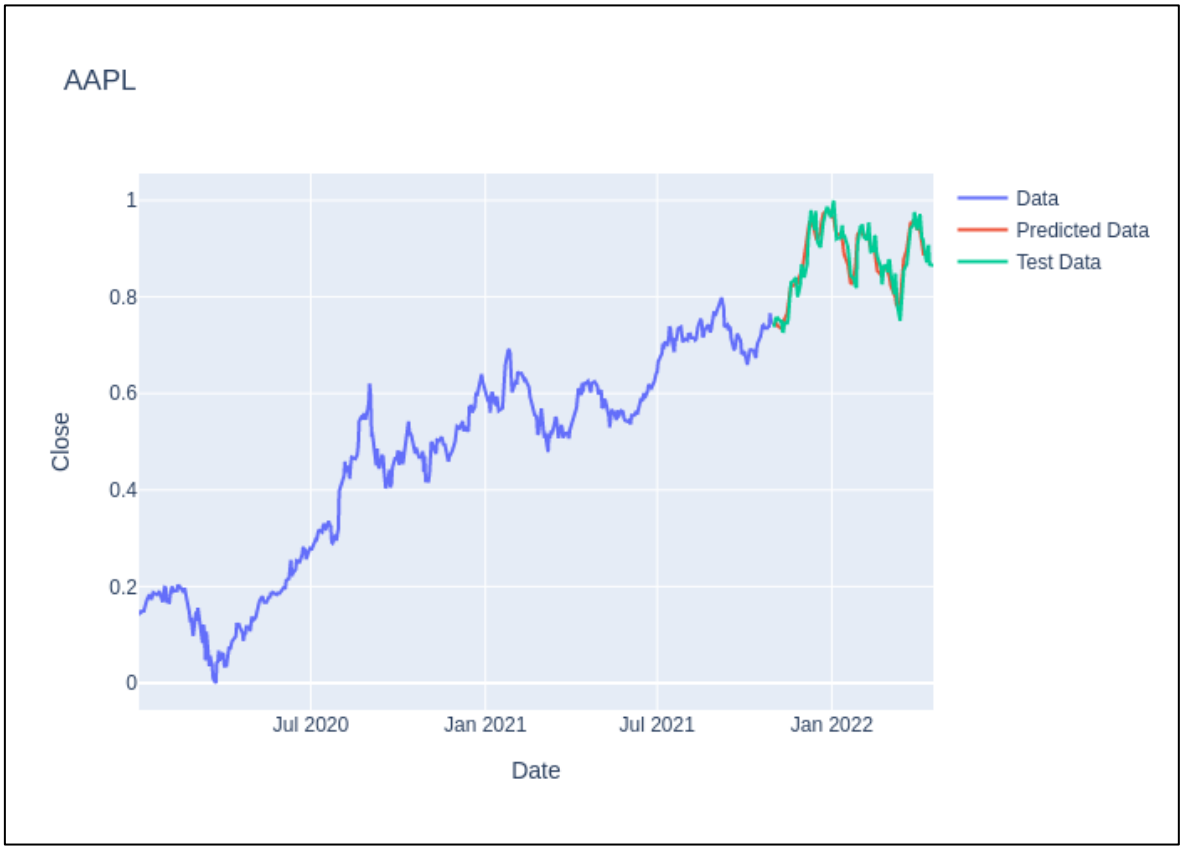

Fig 5.23: Binary Cross Entropy performance for Apple dataset

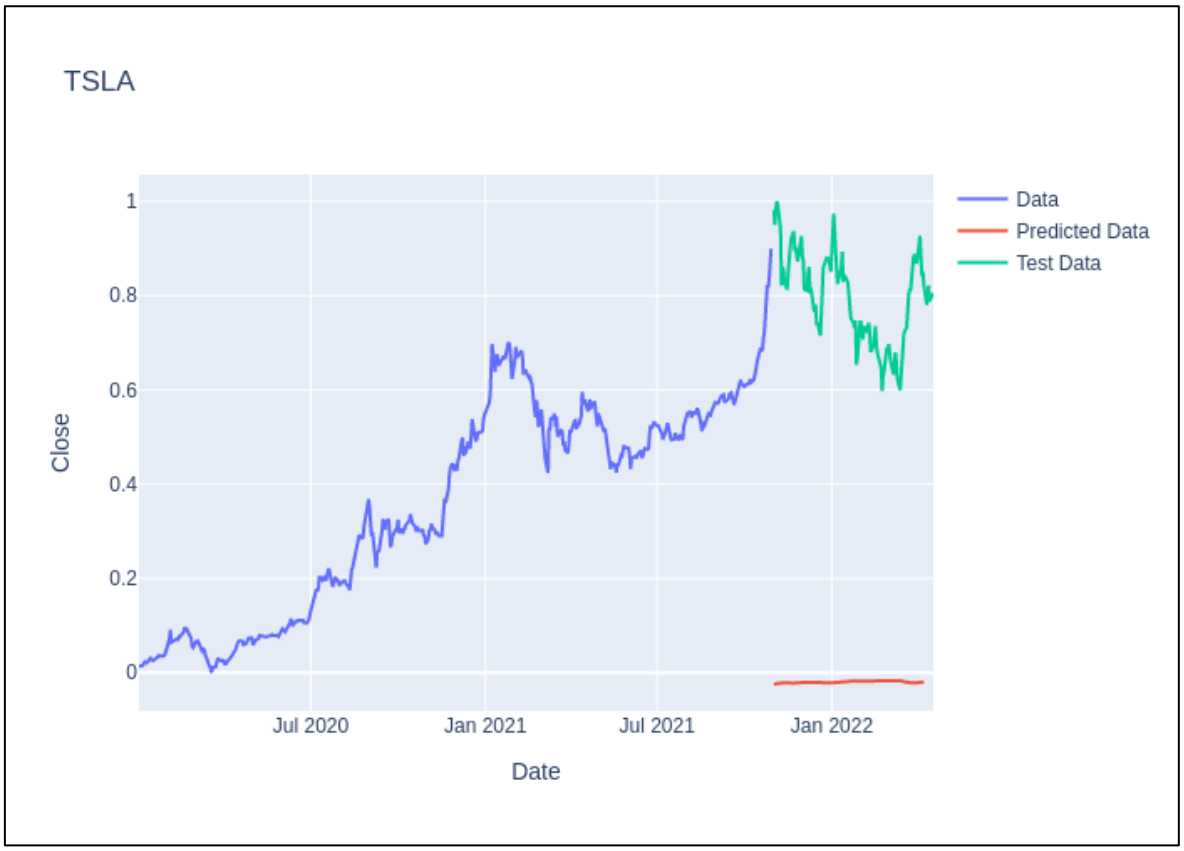

Fig 5.24: Binary Cross Entropy performance for Tesla dataset

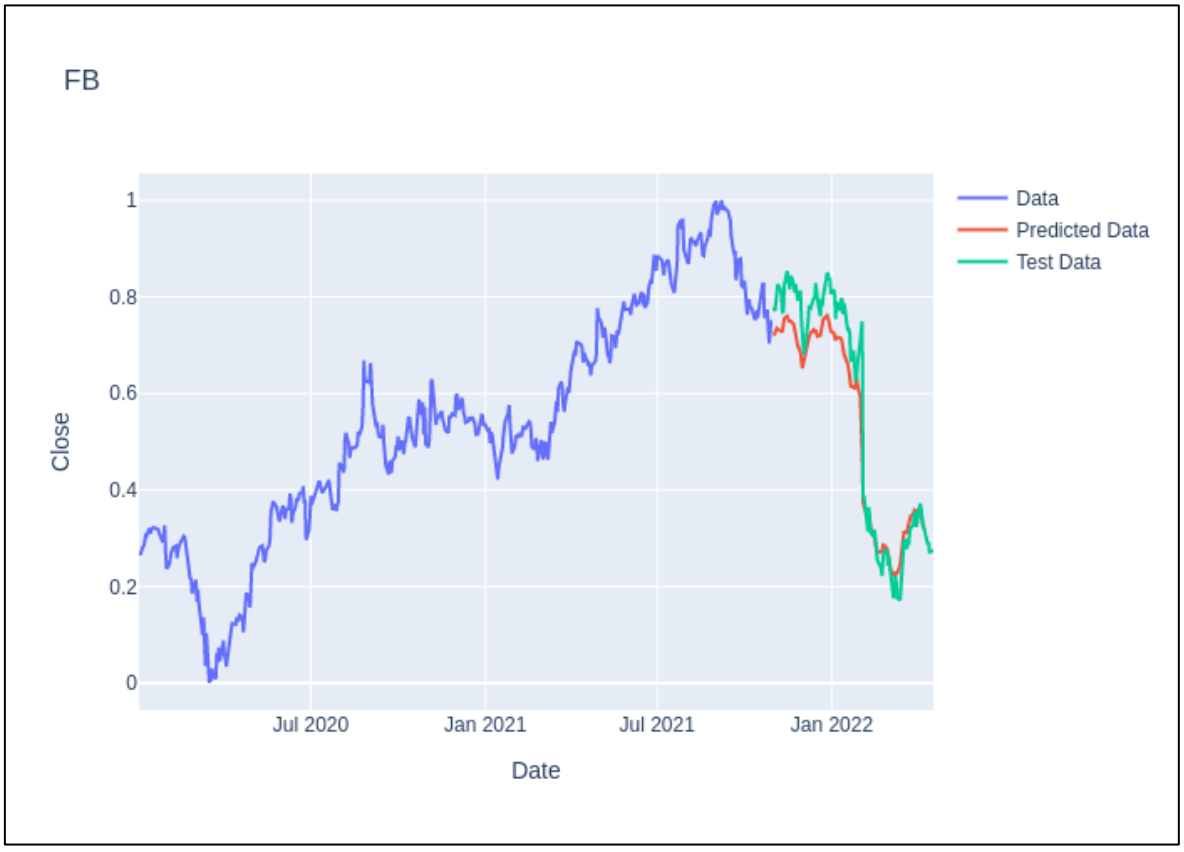

Fig 5.25: Binary Cross Entropy performance for Facebook dataset

Stock	MSE	RMSE	R^2
GOOG	0.00210050573662958	0.04583127465639138	0.527664550993404
TSLA	0.6754275623752923	0.8218440012406809	-68.6425640794958
AAPL	0.00050154160293980	0.02239512453503665	0.901298808233215
AMZN	0.5996937861008489	0.7743989837937862	-45.4885714653309
FB	0.00434053659146433	0.06588274881533354	0.928790730785303

Table 5.5: Binary Cross Entropy performance metrics

Note: MSE and RMSE values are low but R^2 values are not high, it denotes that **Binary Cross Entropy** is not preferable for satisfactory results

2. MSE

Fig 5.26: MSE performance for Google dataset

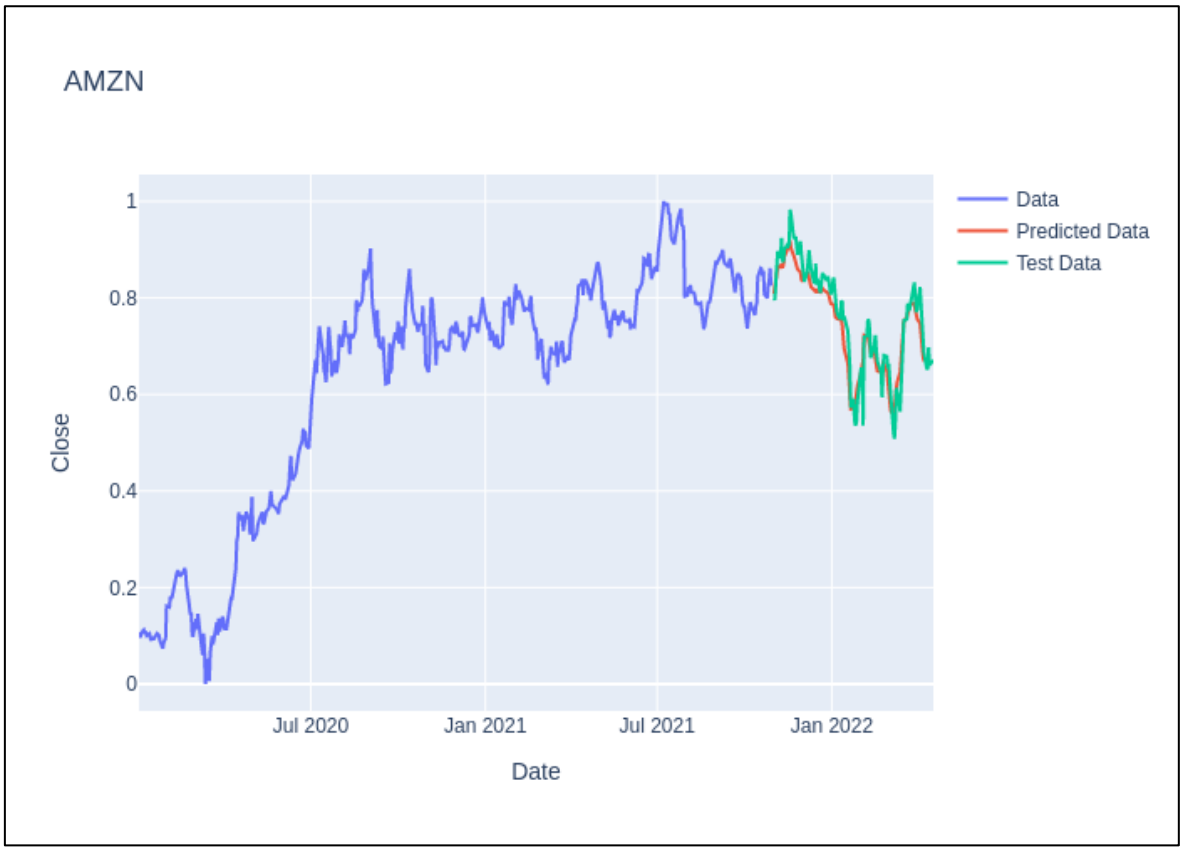

Fig 5.27: MSE performance for Amazon dataset

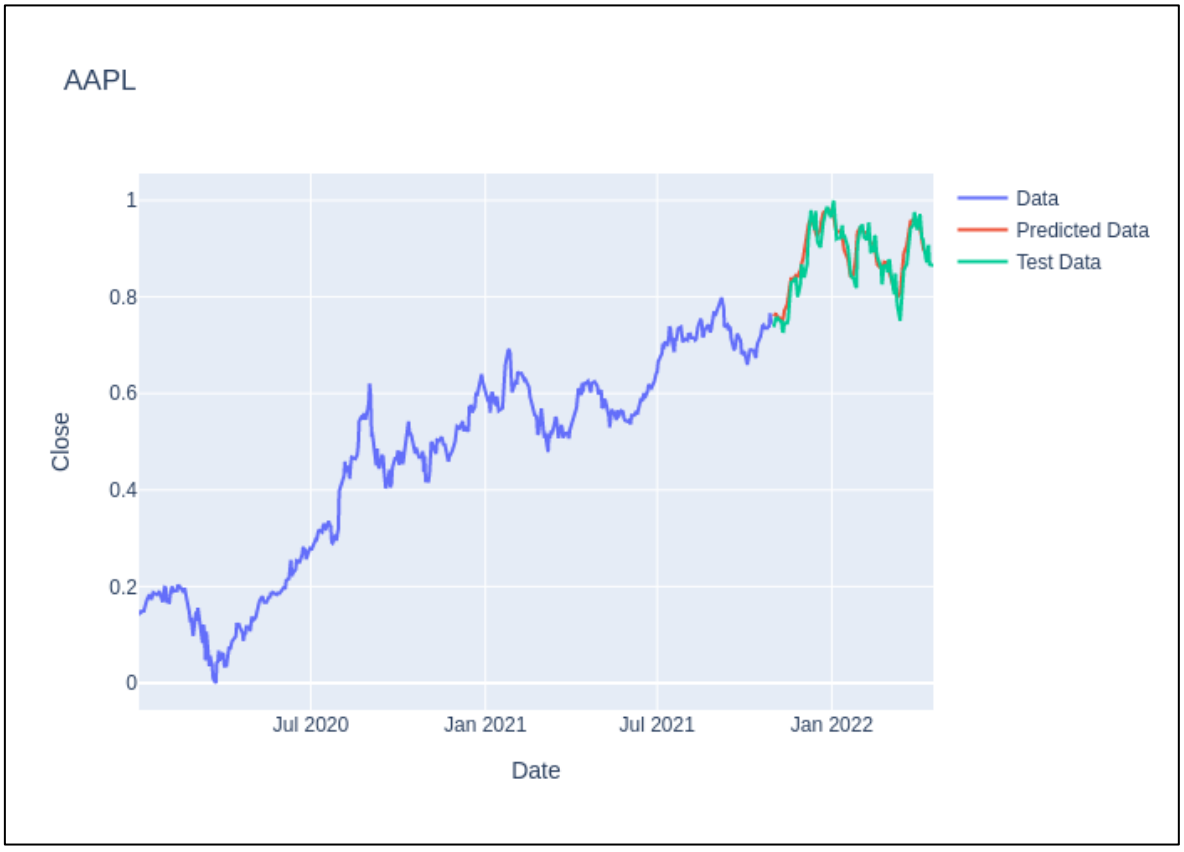

Fig 5.28: MSE performance for Apple dataset

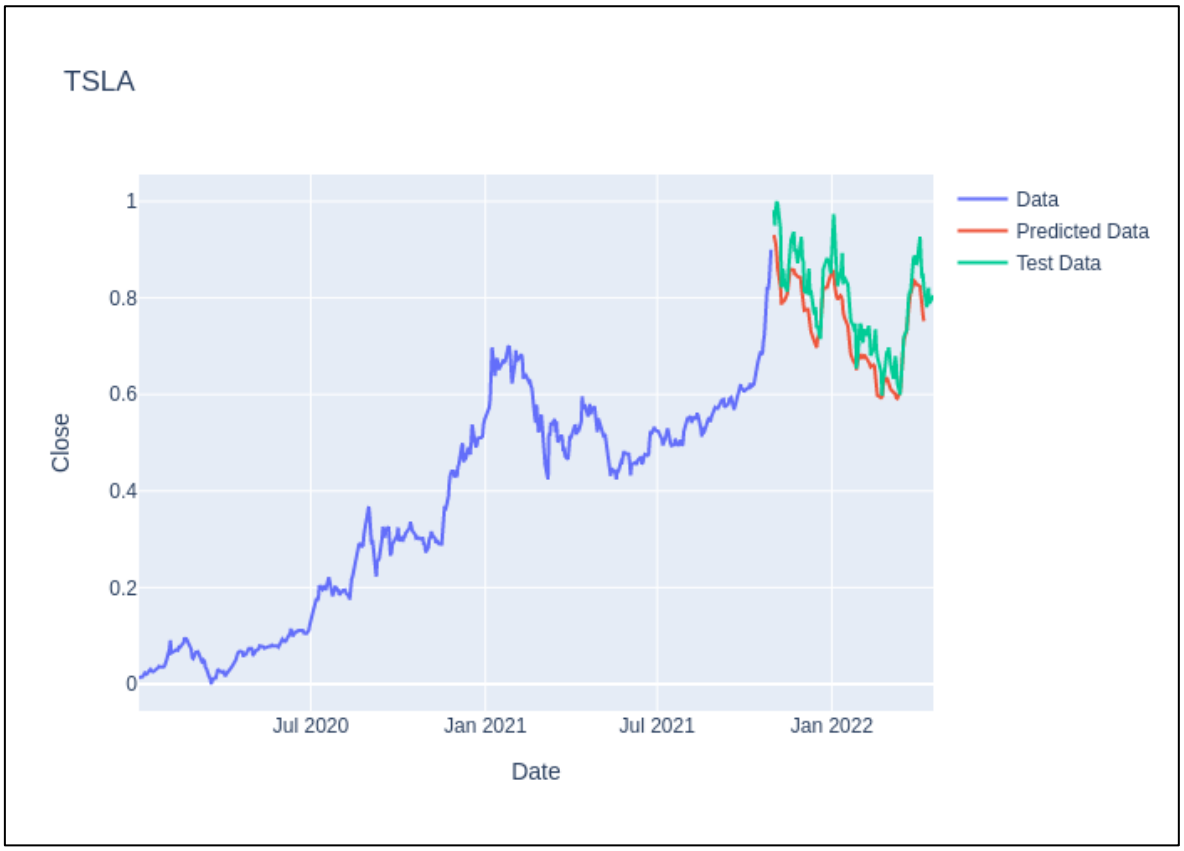

Fig 5.29: MSE performance for Tesla dataset

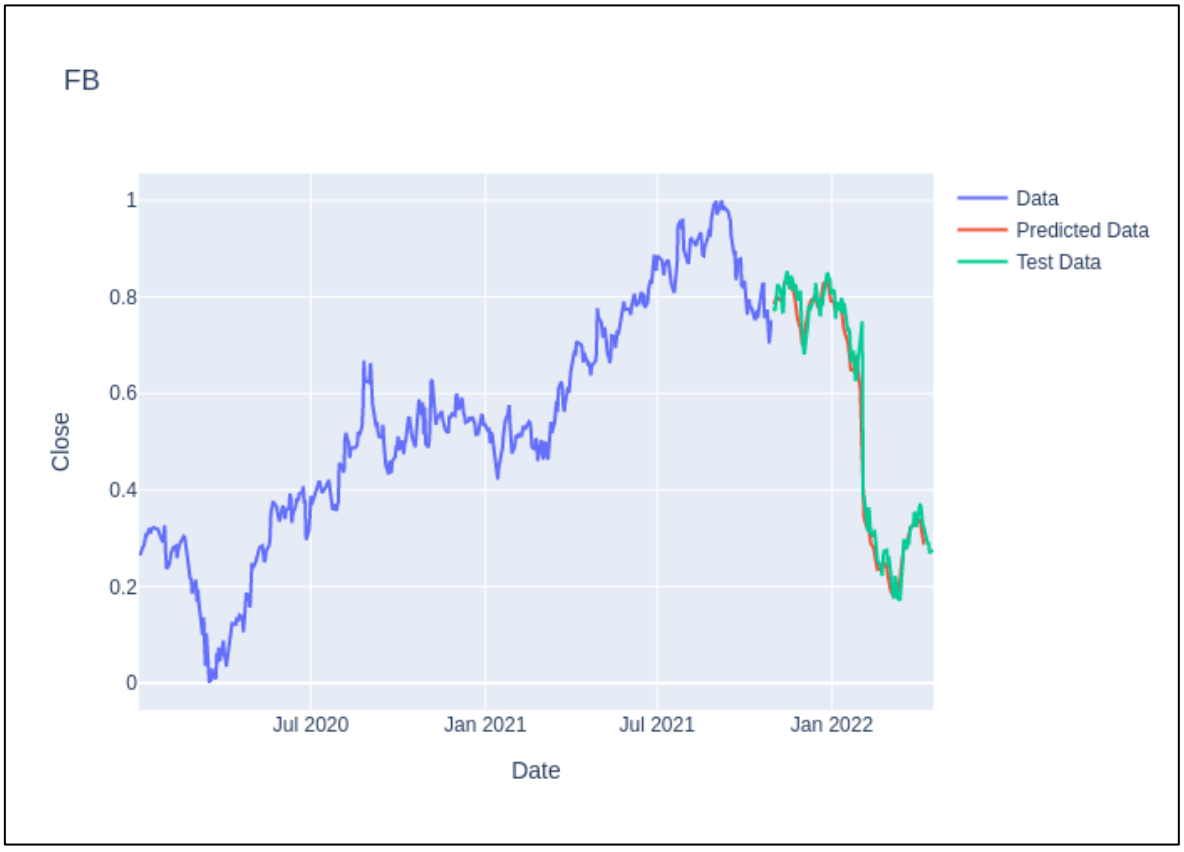

Fig 5.30: MSE performance for Facebook dataset

Stock	MSE	RMSE	R^2
GOOG	0.00089839427162912	0.02997322591295641	0.797980336699396
TSLA	0.00370023022529044	0.06082951771377486	0.618473490084650
AAPL	0.00058991481330011	0.02428816199921507	0.883907347322112
AMZN	0.00149095008374778	0.03861282278916927	0.884420780861827
FB	0.00189279344083100	0.04350624599791402	0.968947517235311

Table 5.6: MSE performance metrics

Note: MSE and RMSE values are low and R^2 values are high, it denotes that **MSE** is preferable for satisfactory results

We have noticed that MSE is the best loss measure

5.2 LSTM PERFORMANCE FOR DIFFERENT DATASETS

5.2.1 FROM 2015-01-01 TO 2022-05-01

Fig 5.31: LSTM performance for Google dataset 2015-22

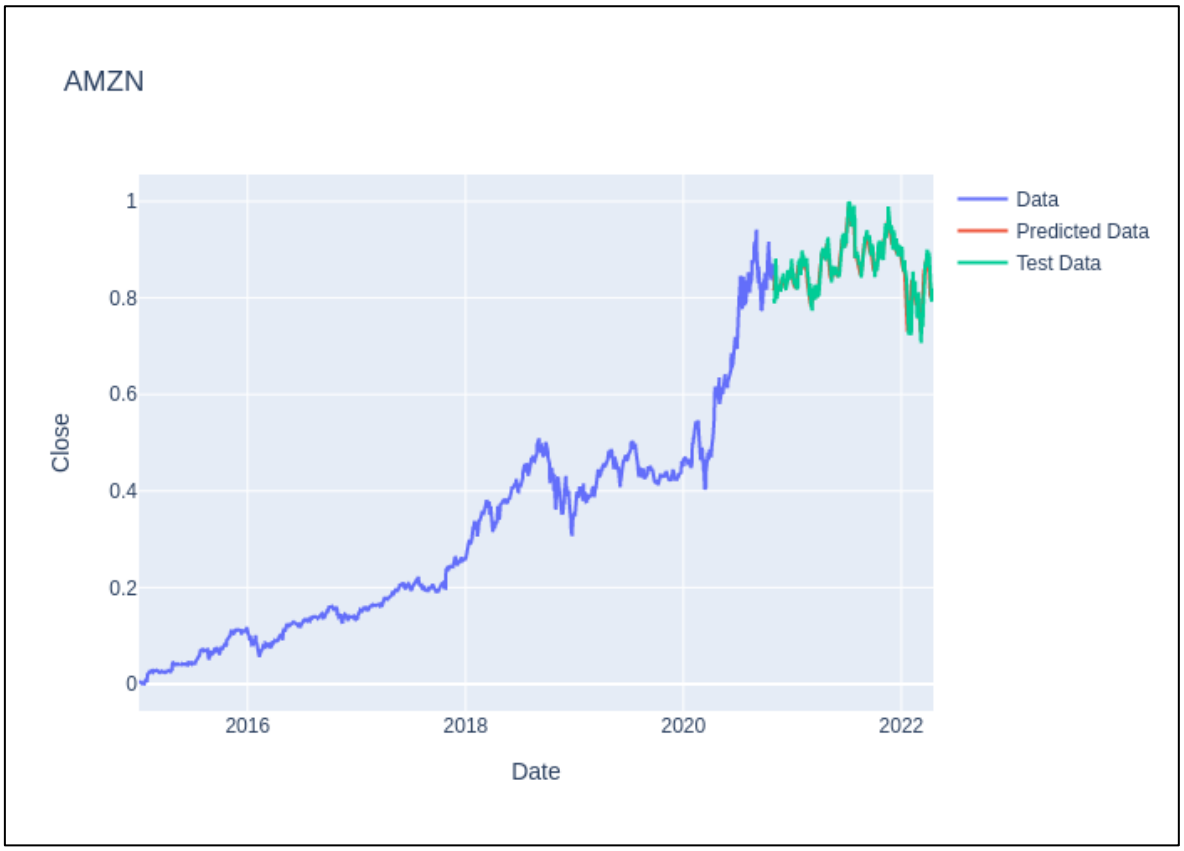

Fig 5.32: LSTM performance for Amazon dataset 2015-22

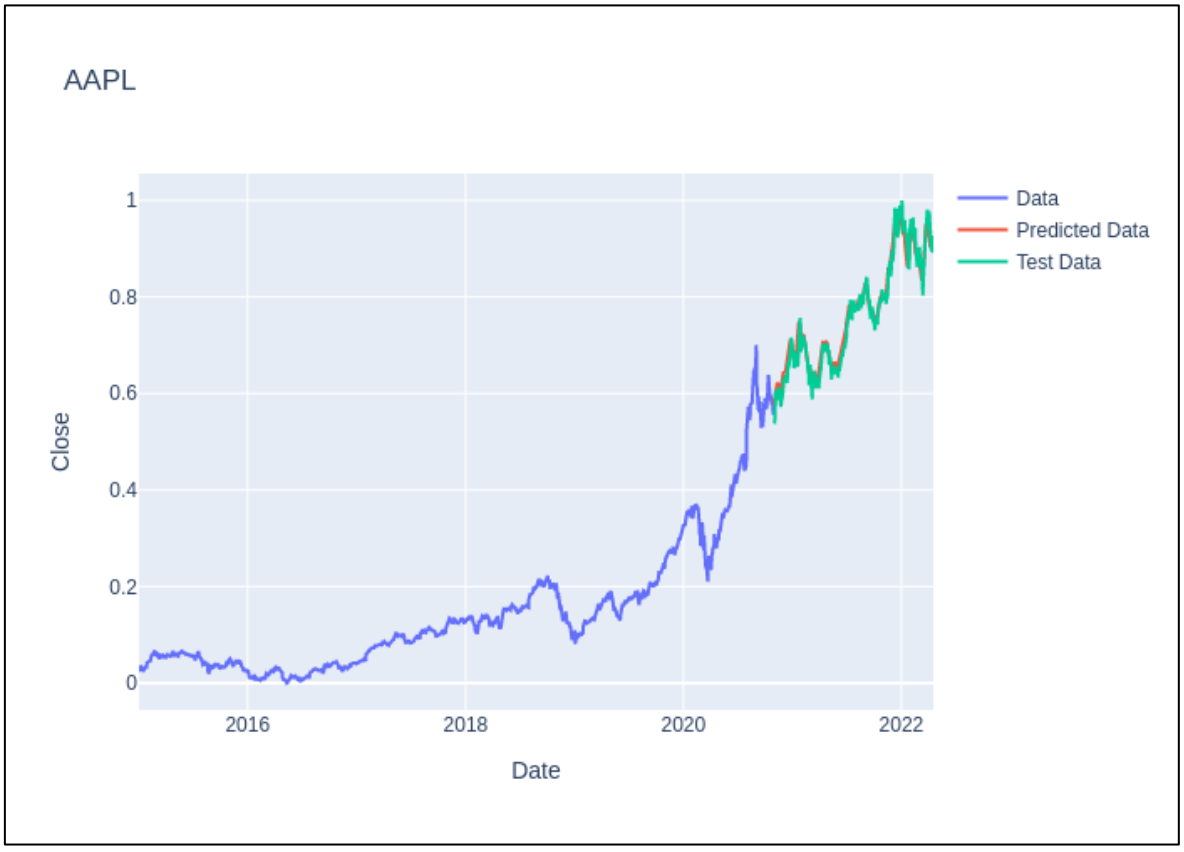

Fig 5.33: LSTM performance for Apple dataset 2015-22

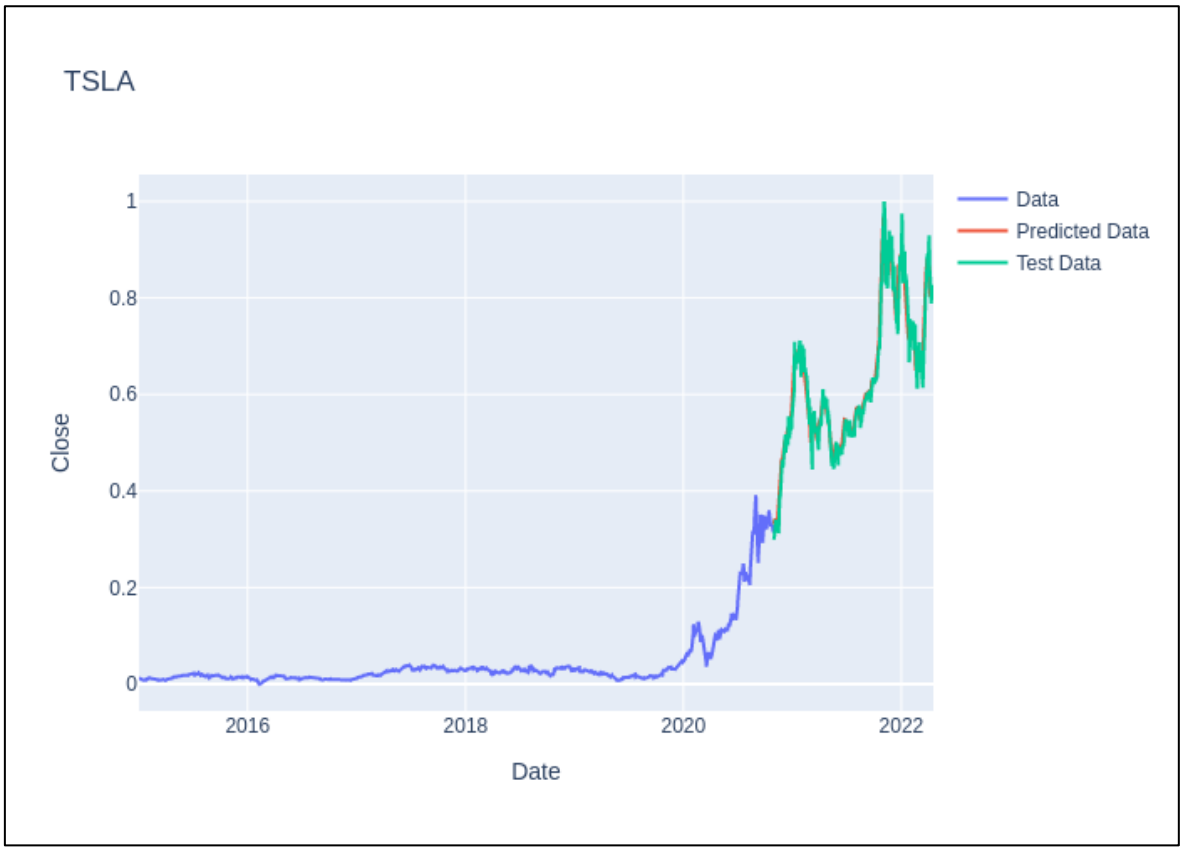

Fig 5.34: LSTM performance for Tesla dataset 2015-22

Fig 5.35: LSTM performance for Facebook dataset 2015-22

Stock	MSE	RMSE	R^2
GOOG	0.000951702849542145	0.0308496815144361	0.964078995107780
TSLA	0.000754424173241407	0.0274667830886947	0.966671113702235
AAPL	0.000374036884008534	0.0193400331956422	0.972056333786266
AMZN	0.000303442140312043	0.0174195907044925	0.896289012584822
FB	0.002584571315014066	0.0508386793201206	0.889693526027689

Table 5.7: LSTM performance metrics for Datasets 2015-22

Note: MSE and RMSE values are low and R^2 values are high, it denotes that **LSTM** is preferable for satisfactory results

5.2.2 FROM 2020-01-01 TO 2022-05-01

Fig 5.36: LSTM performance for Google dataset 2020-22

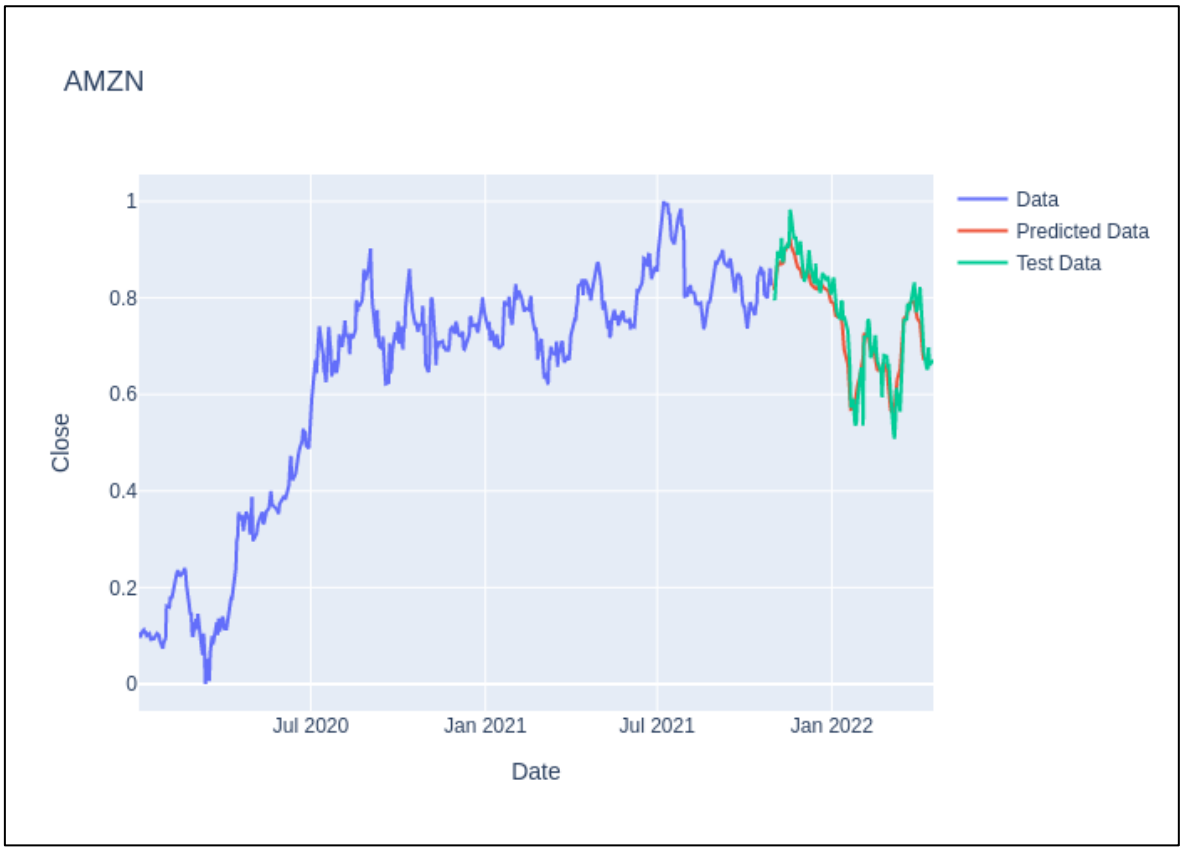

Fig 5.37: LSTM performance for Amazon dataset 2020-22

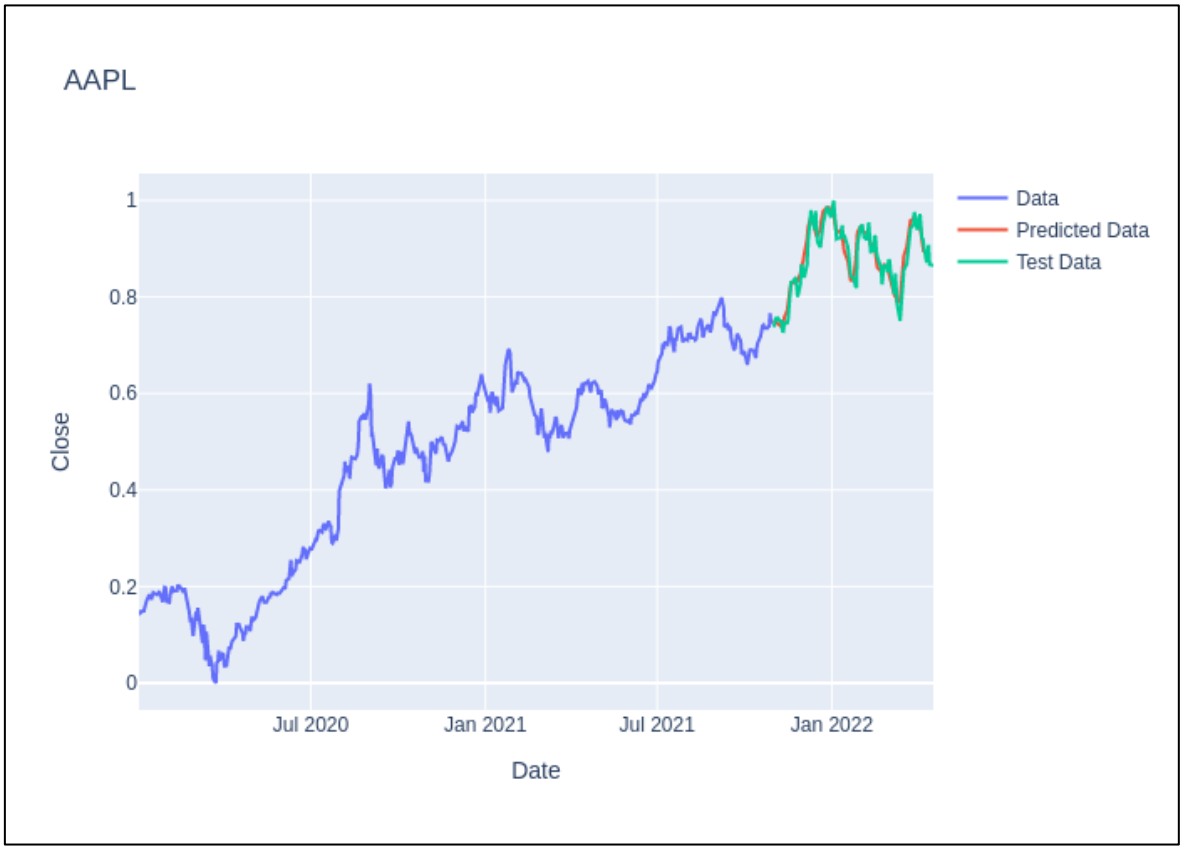

Fig 5.38: LSTM performance for Apple dataset 2020-22

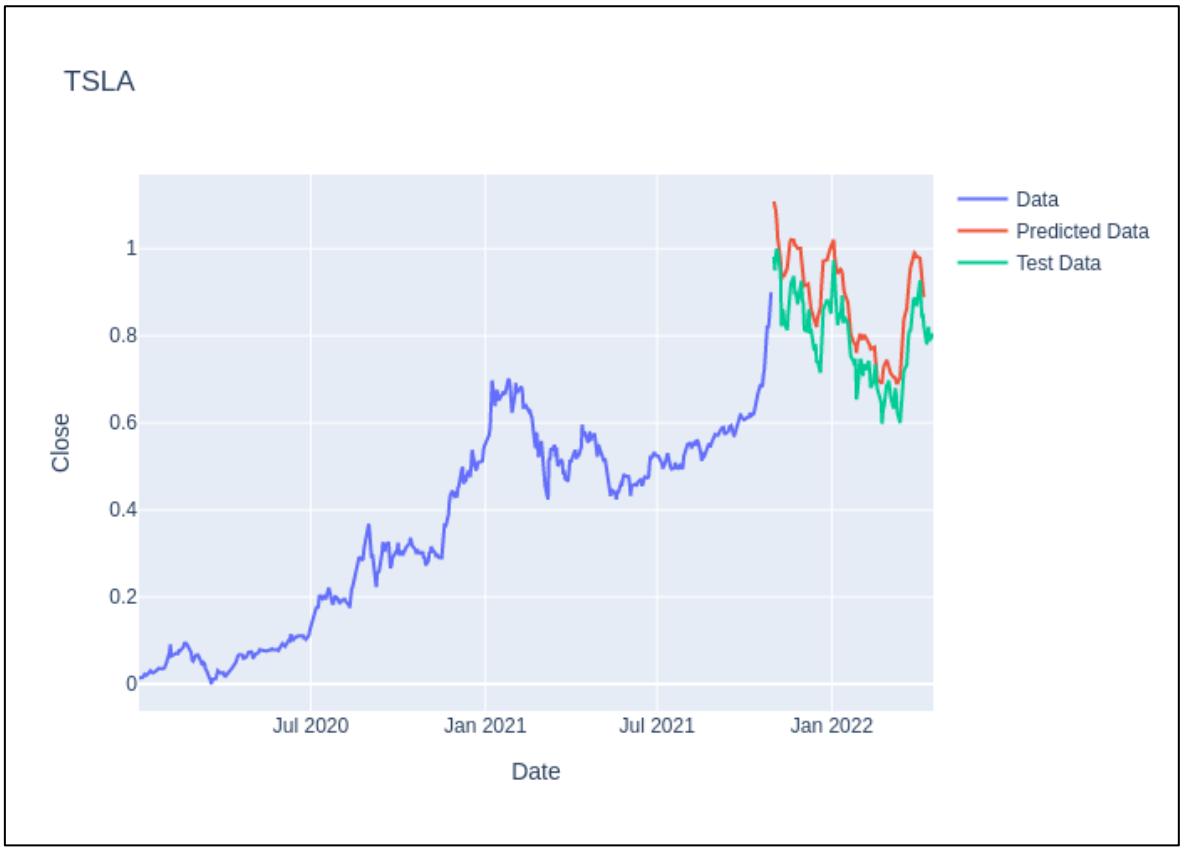

Fig 5.39: LSTM performance for Tesla dataset 2020-22

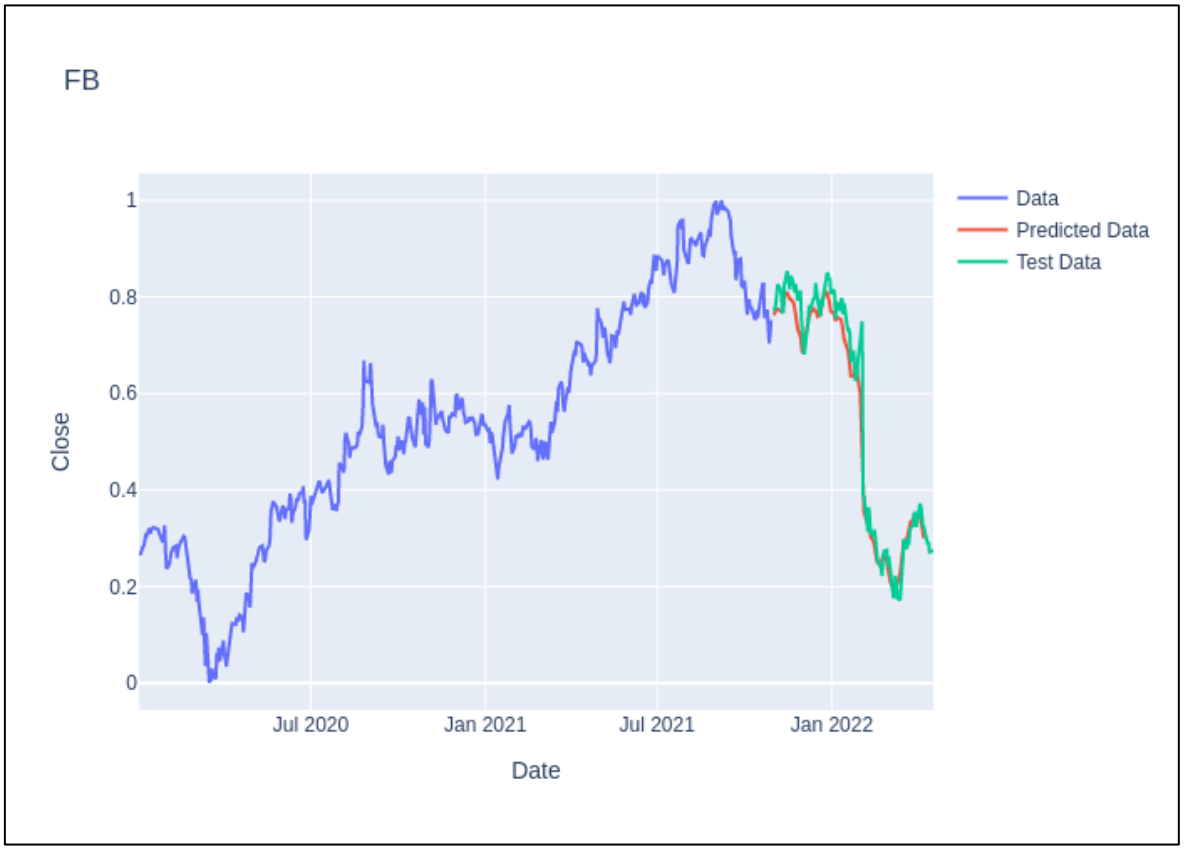

Fig 5.40: LSTM performance for Facebook dataset 2020-22

Stock	MSE	RMSE	R^2
GOOG	0.00129257681205064	0.0359524242861402	0.709341498931063
TSLA	0.00798724587006540	0.0893713929065974	0.576444206143242
AAPL	0.00048913048922225	0.0221162946539933	0.903741261078393
AMZN	0.00144607399614524	0.0380272796311443	0.887899598308244
FB	0.00219677376380134	0.0468697531869045	0.9639605262957732

Table 5.8: LSTM performance metrics for Datasets 2020-22

Note: MSE and RMSE values are low and R^2 values are high, it denotes that **LSTM** is preferable for satisfactory results

5.3 COMPARE LSTM TO OTHER MODELS

Dataset range 2015-22

Note: MSE and RMSE values should be low and R^2 values should be high

Stock	MODEL	MSE	RMSE	R^2	BEST MODEL
GOOG	LSTM	0.001743036	0.041749685	0.808047809	LSTM
	RNN	5022.300541	70.86819132	0.728510611	
	ARIMA	3773.978584	61.43271591	0.795990875	
	SMA	19929.37611	141.1714422	-0.07731787	
AMZN	LSTM	0.001336588	0.036559388	0.896386939	LSTM
	RNN	8791.871955	93.76498256	0.836409021	
	ARIMA	9812.722059	99.05918463	0.817413991	
	SMA	61294.70551	247.5776757	-0.14051488	
AAPL	LSTM	0.000597897	0.024451950	0.882336317	LSTM
	RNN	18.03839872	4.247163609	0.766545727	
	ARIMA	14.24036165	3.773640371	0.815700200	
	SMA	172.4774882	13.13306850	-1.23221624	
TSLA	LSTM	0.001072131	0.032743420	0.889453738	LSTM
	RNN	2804.609871	52.95856749	0.774718625	
	ARIMA	2398.896534	48.97853136	0.807307706	
	SMA	34719.06597	186.3305288	-1.78882242	

Table 5.9: COMPARE LSTM TO OTHER MODELS

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

From the above tables and figures we can infer that LSTM model with ADAM optimizer performs much better than other models such as ARIMA, SMA and RNN. We got satisfactory results that is above 80% accurate stock prediction.

6.2 FUTURE WORK

We add new features like finding the local minima and local maxima to buy and sell the stock respectively, add API’s or some other operations to let the algorithm do the trading for us.

CHAPTER 7

REFERENCES

7. REFERENCES

[1] [For SMA description and formula]

- https://www.investopedia.com/terms/s/sma.asp

[2] [For ARIMA terms and details]

- https://journal.jis-institute.org/index.php/ijmhrr/article/view/235
- https://www.investopedia.com/terms/a/autoregressive-integrated-moving-averagearima.asp

[3] [For RNN details and images]

- https://www.ijert.org/research/future-stock-price-prediction-using-recurrent-neural-network-lstm-and-machine-learning-IJERTCONV9IS05065.pdf
- https://builtin.com/data-science/recurrent-neural-networks-and-lstm

[4] [For SMA and ARIMA implementation]

- https://github.com/0xpranjal/Stock-Prediction-using-different-models/blob/main/Notebooks/1.%20Time%20Series%20Forecasting%20with%20Naive%2C%20Moving%20Averages%2C%20and%20ARIMA.ipynb

[5] [For LSTM details and images]

- https://www.researchgate.net/publication/348390803_Stock_Price_Prediction_Using_LSTM
- https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/
- https://en.wikipedia.org/wiki/Long_short-term_memory#:~:text=Long%20short%2Dterm%20memory%20(LSTM,networks%2C%20LSTM%20has%20feedback%20connections.
- https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

[6] [For LSTM implementation]

- https://towardsdatascience.com/time-series-forecasting-with-recurrent-neural-networks-74674e289816
- https://medium.com/visionary-hub/using-lstms-to-predict-future-stock-prices-61f4458fc860