# BEAR-BULL STOCK MARKET PREDICTION USING STACKED LSTM

**A MINI PROJECT REPORT**

***Submitted by***


**HARINI R** **(721220243014)**

**MAHINDHA** **(721220243029)**

**MANJULA DEVI S** **(721220243031)**

**PAVITHRA R** **(721220243042)**


***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**ARTIFICIAL INTELLIGENCE & DATA SCIENCE**


**KARPAGAM INSTITUTE OF TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY: CHENNAI 600 025**


**DECEMBER 2022**

# ANNA UNIVERSITY: CHENNAI 600 025

## BONAFIDE CERTIFICATE

Certified that this project report "**BEAR-BULL STOCK MARKET PREDICTION USING STACKED LSTM"** is the bonafide work of "**HARINI R (721220243014)**, **MAHINDHA C (721220243029)**, **MANJULA DEVI S (721220243031), PAVITHRA R (721220243042)"** who carried out the project work under my supervision.

**Mrs.N.MANJUBASHINI, M.E.,**

**SUPERVISOR**

Assistant professor,

Department of Artificial Intelligence &Data Science,

Karpagam Institute of Technology,

Coimbatore – 641105

**Dr.R.NALLAKUMAR, M.E.,Ph.D,**

**HEAD OF THE DEPARTMENT**

Assistant professor,

Department of Artificial Intelligence & Data Science,

Karpagam Institute of Technology,

Coimbatore – 641105

Submitted for the university project Viva-voce examination conducted at Karpagam Institute of Technology, Coimbatore, on …….

**Internal Examiner** **External Examiner**

# ACKNOWLEDGEMENT

With genuine humility, we are obediently thankful to God Almighty without him, this work would have never been a reality.

We express our profound gratitude to our respected Chairman **Dr.R.Vasanthakumar**, for giving this opportunity to pursue this course. At this pleasing moment of having successfully completed the projects work.

We wish to acknowledge sincere gratitude and heartfelt thanks to our respected Principal **Dr.P. Manimaran Ph.D**., and Vice Principal **Dr.D.Bhanu., M.E.,Ph.D.**, for having us given the adequate support and opportunity for completing the project work successfully.

We express our deep sense of gratitude and sincere thanks to our beloved Head of the Department and project coordinator **Dr.R.Nallakumar,M.E.,Ph.D,** who has been a spark for enlightening our knowledge.

Our profound gratitude goes to project guide **Mrs.N.MANJUBASHINI,M.E.,** and review members and all the faculty members of Department of Artificial Intelligence and Data Science for the invaluable knowledge they have imparted on us.

Our humble gratitude and hearties thanks go to our family members and friends to their encouragement and support throughout the course of this project.

**HARINI R (721220243014)**

**MAHINDHA C (721220243029)**

**MANJULA DEVI S (721220243031)**

**PAVITHRA R (721220243042)**

# ABSTRACT

The stock market is uncertain, volatile, and multidimensional. Stock prices have been difficult to predict since they are influenced by a variety of factors. In order to make critical investment and financial decisions, investors and analysts are interested in predicting stock prices. Predicting a stock's price entails developing price pathways that a stock might take in the future. The use of Long Short-Term Memory (LSTM) networks for stock market prediction is a popular method in the field of financial time series analysis. LSTMs are a type of recurrent neural network (RNN) that are well-suited for modeling sequential data, such as stock market data, due to their ability to retain long-term memory. In a typical LSTM-based stock market prediction model, historical stock market data, such as stock prices and trading volumes, are used as input to the network. The LSTM network is trained to learn the patterns and relationships in the data, and to make predictions about future stock prices. One key aspect of using LSTMs for stock market prediction is the preprocessing of the data. This typically involves normalizing the data, removing any irrelevant features, and creating a set of input-output pairs for the network to learn from. The input to the LSTM network is a sequence of historical stock market data, and the output is the corresponding future stock price. Once the LSTM network is trained, it can be used to make predictions about future stock prices. These predictions can then be used by traders and investors to make informed decisions about buying and selling stocks.

# TABLE OF CONTENTS

# LIST OF ABBREVIATIONS

**LSTM** - Long Short Term Memory

**CSV** - Comma Separated Value

**ANN** - Artificial Neural Network

**GBM** - Geometric Brownian Motion

**APPL** - Apple Stock Trade

**ML** - Machine Learning

# LIST OF FIGURES

# CHAPTER 1
# INTRODUCTION

## 1.1 DATA SCIENCE

Data science is an interdisciplinary field that uses scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data. It involves using techniques from statistics, machine learning, and computer science to analyze and interpret data, and make predictions or decisions. Data science can be applied to a wide range of industries, including finance, healthcare, retail, and technology. Data science can be used to predict stock market trends and make investment decisions. This is done by analyzing historical stock market data, such as prices, volumes, and news articles, and using machine learning algorithms to identify patterns and make predictions.

## 1.2 STOCK MARKET

The stock market is a marketplace where stocks (or shares) of publicly traded companies are bought and sold. When a company wants to raise money, it can do so by issuing stocks, which are then bought by investors. The price of a stock is determined by supply and demand in the market. If more people want to buy a stock than sell it, the price will go up. If more people want to sell a stock than buy it, the price will go down. The stock market is often divided into two main categories: the primary market and the secondary market. The primary market is where new stocks are issued, typically through an initial public offering (IPO). The secondary market is where existing stocks are bought and sold after they have been issued in the primary market.

The most well-known stock market is the New York Stock Exchange (NYSE), which is located in the United States. Other major stock markets include the NASDAQ, the London Stock Exchange (LSE), the Apple (AAPL) and the Tokyo Stock Exchange (TSE). Investing in the stock market can be a way to grow wealth over the long term, but it also comes with risks. The value of stocks can fluctuate significantly in response to changes in the economy, company performance, and other factors. It's important for investors to conduct thorough research and diversify their portfolios to manage risk.

## 1.3 CHALLENGES INVOLVED IN STOCK MARKET PREDICTION

Predicting the stock market is a complex task that comes with its own set of challenges. Some of the main challenges include:

- Non-linearity: The stock market is a non-linear system, which makes it difficult to predict future price movements using linear models.
- Noise: The stock market is affected by a variety of factors, such as economic indicators, company news, and political events, which can create a lot of noise in the data. This can make it difficult to identify meaningful patterns and make accurate predictions.
- Complexity: The stock market is a complex system that is influenced by a large number of interrelated factors. This makes it difficult to develop models that can accurately capture the underlying relationships and make accurate predictions.
- High dimensionality: The stock market generates a large amount of data, which can make it difficult to process and analyze. This can lead to overfitting and poor model performance.
- Overfitting: Overfitting occurs when a model is trained on historical data and performs well on that data but poorly on new unseen data.
- Data availability and quality: High-quality and relevant data is crucial for stock market predictions. However, data availability and quality can be a challenge, especially for smaller companies or emerging markets.
- Lack of consensus: There is no consensus among experts about the best methods and models for stock market prediction, which can make it difficult to determine the most effective approach.
- Ethics and regulation: With the increasing use of AI and Machine Learning models in the stock market, ethical and regulatory issues may arise, such as insider trading, market manipulation, and data privacy.

Despite these challenges, researchers and practitioners continue to develop new methods and models to improve stock market predictions, such as using advanced techniques like Deep Learning and Ensemble methods, and combining multiple sources of data. However, it's important to remember that stock market prediction is not a sure thing and past performance does not guarantee future results.

## 1.4 STOCK MARKET PREDICTION ALGORITHMS

There are many algorithms that can be used for stock market prediction, with varying levels of complexity and accuracy. Some of the most commonly used algorithms include:

- Moving Average: This is a simple technique that calculates the average price of a stock over a certain period of time. It can be used to identify trends in the stock market and make predictions about future price movements.
- Time Series Models: These are a class of models that are specifically designed to handle time-series data. Examples include ARIMA, Exponential Smoothing and GARCThey are used to make predictions about future values based on historical data.
- Machine Learning: Machine learning algorithms such as Decision Trees, Random Forest, Support Vector Machine, and Neural Networks can be used to make predictions about stock prices. They can be trained on historical data and used to make predictions about future price movements.
- Deep Learning: Deep Learning models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) Networks can be used to analyze large amounts of data, such as financial news, and make predictions about stock prices.
- Hybrid Models: Hybrid models are a combination of two or more models, they can be used to improve the prediction performance by combining the strengths of multiple models. For example, a combination of Time series models and Machine Learning models.
- Sentiment Analysis: Sentiment analysis algorithms can be used to analyze social media, news articles, and other forms of text data to determine the overall sentiment towards a particular stock or market. The sentiment can then be used as a feature in a machine learning model to make predictions about future stock prices.

It's important to note that no single algorithm is guaranteed to provide accurate predictions, and the choice of algorithm will depend on the specific problem and the available data. Also, it's important to remember that past performance does not guarantee future results, and stock market predictions should be treated as a probabilistic approach, rather than a definite answer.

## 1.5 NEED FOR THE STUDY

The study of stock market prediction is important for several reasons:

- Investment decisions: Accurate stock market predictions can help investors make more informed decisions about when to buy or sell stocks.
- Risk management: Stock market predictions can help investors and traders better manage their risk by providing a better understanding of potential market movements.

- Trading strategies: Stock market predictions can be used to develop trading strategies, such as algorithmic trading, that can be executed automatically based on predicted market movements.
- Portfolio management: Stock market predictions can be used to optimize portfolio performance by identifying undervalued or overvalued stocks.
- Hedge funds and asset management: Accurate stock market predictions can help hedge funds and asset management firms generate higher returns for their clients.
- Business forecasting: Companies and organizations use stock market predictions to make decisions about investments, financing, and other financial matters.
- Research: The study of stock market prediction can help researchers and practitioners to understand the underlying mechanisms and relationships that drive market behavior and develop new methods and models to improve predictions.
- Policy Making: Governments and regulatory bodies use stock market predictions to make decisions about monetary policy, fiscal policy and other economic policies that can affect the stock market.

Overall, the study of stock market prediction is important for anyone who wants to make informed investment decisions, manage risk, or pursue a career in finance. Additionally, the study of stock market prediction can help researchers and practitioners to better understand the complex relationships that drive market behavior and develop new methods and models to improve predictions and forecasting.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 STOCK MARKET PREDICTION USING SUPERVISED MACHINE LEARNING TECHNIQUES

- Numerous classification and regression models have been used in stock market predictions for many years. Several supervised machine learning techniques applied in stock market prediction yielded a better outcome. For more than a decade, a number of supervised machine models and techniques has been proposed or implemented for stock market predictions.
- Despite the better accurate results recorded in stock market predictions by using techniques like Support Vector Machine (SVM) , K NearestNeighbor (KNN), Support Vector Regression (SVR), Linear Regression and Artificial Neural Network (ANN)
- Many algorithms have shown a promising prediction result, depending on the type of data used for training models. Some algorithms perform better with small amount of data while some predict better with large amount data like LSTM, SVM etc…
- To overcome the challenges in the stock market analysis, several computational models based on soft-computing and machine learning paradigms have been used in the stock market analysis, prediction, and trading.
- There is need for an efficient technique(s) that will overcome the challenges faced by using the existing techniques. i.e LSTM.

## 2.2 STOCK MARKET PREDICTION USING MACHINE LEARNING TECHNIQUES

- With the rise of computational power and the availability of big data, researchers have tended to use web resources to predict stock market movements. Machine learning techniques, Natural Language Processing techniques can be used to predict the behavior of the stock markets.
- Historical market data and live stock prices were gathered using Yahoo finance API.But in our implementation those data are collected using Tingo.
- Gathered Tweets were English, and the only keyword used to search Tweets was company Cash tag. e.g. $AAPL for Apple Inc. Twitter sentiment values are generated with each tweet and the mean sentiment score is calculated.

- The predicted stock values are compared with the actual values to evaluate the accuracy. Following section describes stock market predictions done with Twitter, web news, search engine query hits and the integrated model with ensemble methodology. Twitter sentiment correlation with stock market values of $AAPL is 0.5.
- The predicted stock values are compared with the actual values to evaluate the accuracy.Likewise we predict the accuracy or stock price for next 10 days.

## 2.3 PREDICTING STOCK MARKET PRICE MOVEMENT USING MACHINE LEARNING TECHNIQUE

- A GBM is frequently used to predict stochastic price movements of financial assets using estimates of drift and volatility. There is already a wealth of historical financial data accessible for download, which may be used to estimate parameters and compare simulations to real world pricing
- The problematic portion of the GBM model is depicted as part of the stock's unpredictability and stochastic interaction termed the Weiner measure, which further consolidates arbitrary instability over time,but in LSTM there is no such issues.
- Only the characteristics that would be given to the neural network and GBM model are chosen at this layer. The selected characteristic was from the list of Date, open, high, low, close, and volume but in LSTM we have attributes like symbol, date, close, high, low, open, volume, adjClose, adjHigh, adjLow, adjVloume, divCash.
- The average cumulative error rate (MAPE) was calculated between the actual and stimulated values but we calculate RMSE value .
- To obtain higher accuracy in the predict price value new variable have been created using existing variables. ANN and Geometric Brownian Method is used for predicting the next day closing price of the selected stock.

## 2.4 SHARE PRICE PREDICTION USING MACHINE LEARNING TECHNIQUE

- Using LSTM and RNN they form the model to train the architecture and predict the data and the accuracy of themethod is very close in few tests and can be reliable to predict the stock price using this method.
- It follows the methodology of data then model then predict and finally error calculation where there are few more works gone across this field using various other different

methods using deep learning using text based method also using the numerical and both using RNN, LSTM and few other mechanism.

- Supporting vectors in machine learning can provide the classification and regression for the model and few other trending models such as deep neural network, linear regression etc.
- Take the set of data from NSE India based on the time-gap where we filter the company based on the conditions such as base value is greater than or equal to current market price whereas our dataset is collected using Tingo .
- So finally stock market price prediction on the dataset demonstrate that our model is effective and show that this model work efficiently than other methods.

## 2.5 STOCK MARKET ANALYSIS USING SUPERVISED MACHINE LEARNING

- The dataset taken is for GOOGL by WIKI and can be extracted from quandl using the token “WIKI/GOOGL”. Whereas The dataset taken is for APPLE by pandas_reader and is extracted from TINGO using the reference key “7d9ae600455f1b60063e5e2742667e250846caef ”.
- We select the attribute “Close” to be our label (The variable which we shall be predicting) and use “Adj. Open, Adj. High, Adj. Close, Adj. Low and Adj. Volume” to extract the features that will help us predict the outcome better.
- They used SciPy, Scikit-learn and Matplolib libraries in python to program their model, train them with the features and label which we extracted and then test them with the same data but we used numpy, pandas_datareader, matplotlib, MinMaxScaler, tensorflow, keras.
- For testing in supervised machine learning, we input some combination of features into the trained classifier and cross check the output of the classifier with the actual label. This helps us determine the accuracy of our classifier. Which is very crucial for our model.
- Training of a model is very straightforward. You only need to make sure that the data is consistent, coherent and is available in great abundance. A large set of training data contributes to a stronger and more accurate classifier which ultimately increases the overall accuracy.
- Testing is also a very straightforward process. Make sure your test data is at least 20% of the size of your training data.

# CHAPTER 3

# SYSTEM SPECIFICATION

## 3.1 HARDWARE REQUIREMENTS

- 8 GB RAM
- 80 GB SSD
- Intel Core i5-8259U, or AMD Ryzen 5 2700X (Processor)
- NVIDIA GT 1050 or Quadro P1000 (Graphic Card)

## 3.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM : WINDOWS 10 AND ABOVE

LABGUAGES : Python

SOFTWARE : Pycharm, Jupyter Notebook

SERVER : Local Database(If requires)

## 3.3 SOFTWARE OVERVIEW

### 3.3.1 Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes. It supports multiple programming paradigms beyond object-oriented programming, such as procedural and functional programming. It uses a simplified syntax with an emphasis on natural language, for a much easier learning curve for beginners. And, because Python is free to use and is supported by an extremely large ecosystem of libraries and packages, it's often the first-choice language for new developers.

### 3.3.2 Pycharm

It allows viewing of the source code in a click. Software development is much faster using PyCharm. The feature of error spotlighting in the code further enhances the development process. The community of Python Developers is extremely large so that we can resolve our queries/doubts easily. PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. It makes Python development accessible to those who are new to the world of software programming. PyCharm Community Edition is excellent for developers who wish to get more experience with Python.

### 3.3.3 Jupyter Notebook

Jupyter Notebook allows users to compile all aspects of a data project in one place making it easier to show the entire process of a project to your intended audience. Through the web-based application, users can create data visualizations and other components of a project to share with others via the platform. The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience. Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows you to render a publicly available notebook in the browser directly. Jupyter is another best IDE for Python Programming that offers an easy-to-use, interactive data science environment across many programming languages besides Python.

### 3.3.4 Local Database

Local databases reside on your local drive or on a local area network. They often have proprietary APIs for accessing the data. When they are shared by several users, they use filebased locking mechanisms. Because of this, they are sometimes called file-based databases. The Oracle. Oracle is the most widely used commercial relational database management system, built-in assembly languages such as C, C++, and Java. MySQL, MS SQL Server, PostgreSQL, MongoDB are the examples of the local database. Personal database system is the local database system which is only for one user to store and manage the data and information on their own personal system. There are number of applications are used in local computer to design and managed personal database system.

### 3.3.5 Python Libraries:

There are several popular Python libraries that can be used for Stock market prediction, including:

**Tensorflow:** Tensorflow is an end-to-end open-source deep learning framework developed by Google and released in 2015. It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android. TensorFlow is a symbolic math library used for neural networks and is best suited for dataflow programming across a range of tasks. It offers multiple abstraction levels for building and training models.

**Keras:** Keras is an effective high-level neural network Application Programming Interface (API) written in Python. This open-source neural network library is designed to provide fast experimentation with deep neural networks, and it can run on top of CNTK, TensorFlow, and Theano.Keras focuses on being modular, user-friendly, and extensible. It doesn't handle low-level computations; instead, it hands them off to another library called the Backend.

**Pandas:** Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

**Scikit-learn**: Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

**Matplotlib:** Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications.

**Sklearn.metrics:** The sklearn.metrics module implements several loss, score, and utility functions to measure classification performance. Some metrics might require probability estimates of the positive class, confidence values, or binary decisions values.

# CHAPTER 4

# DESIGN METHODOLOGY

## 4.1 PROBLEM DEFINITION

The stock market is uncertain, volatile, and multidimensional. Stock prices have been difficult to predict since they are influenced by a variety of factors. In order to make critical investment and financial decisions, investors and analysts are interested in predicting stock prices.

Techniques used in machine learning will give more accurate, precise and simple way to solve such issues related to stock and market prices. There are ample amounts of data about stocks but the most difficult and intriguing thing is to predict the price of these stocks based on old data.

## 4.2 PROPOSED METHODOLOGY

It is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. After you have your requirements for your system, the next step is translating them into technical specifications so you can construct your system. This is where system design comes in.

### 4.2.1 SYSTEM ARCHITECTURE

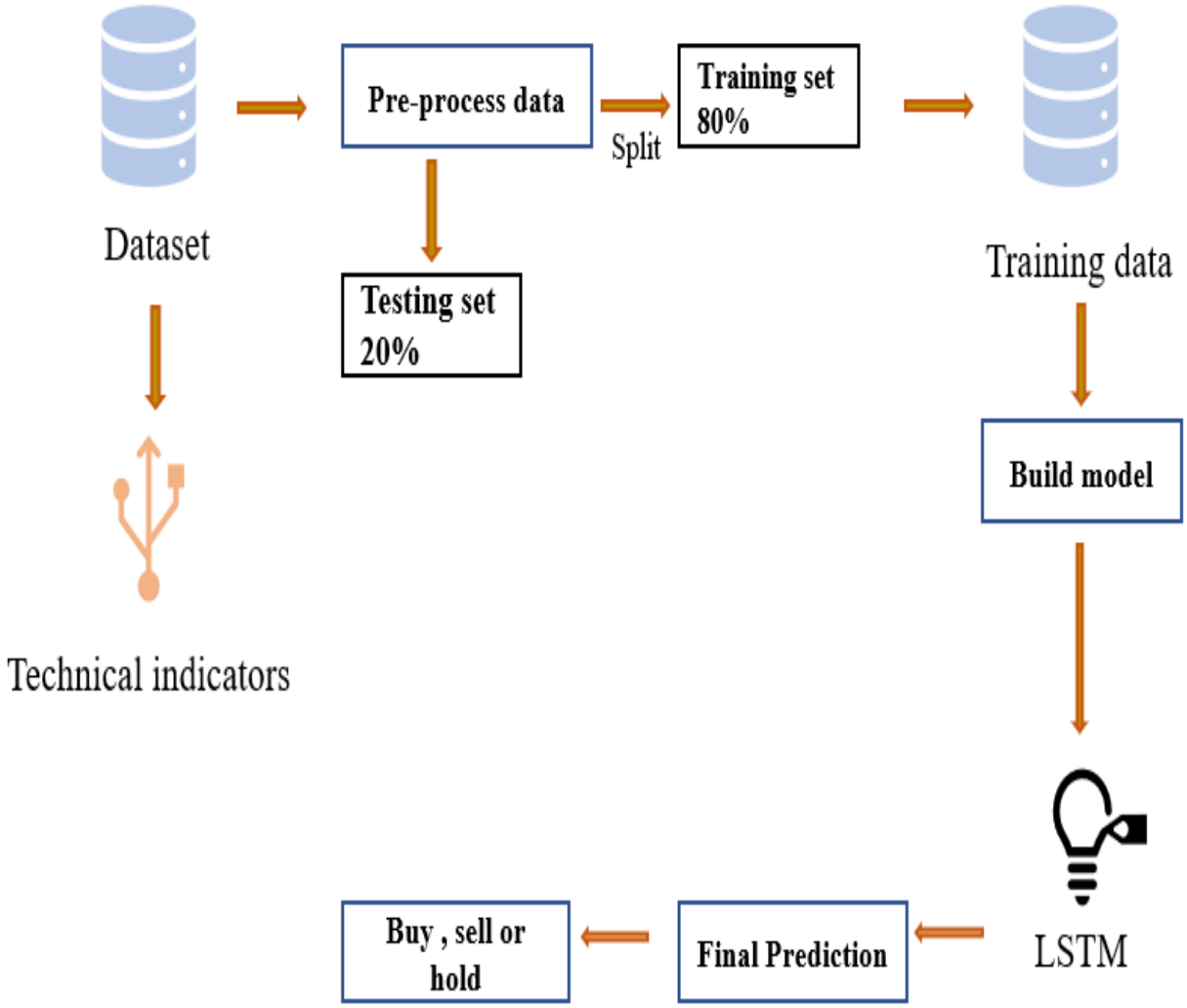


**Figure 4.1 System Architecture**

### 4.2.2 Dataflow diagram for Data Collection, Data Preparation and Cleaning

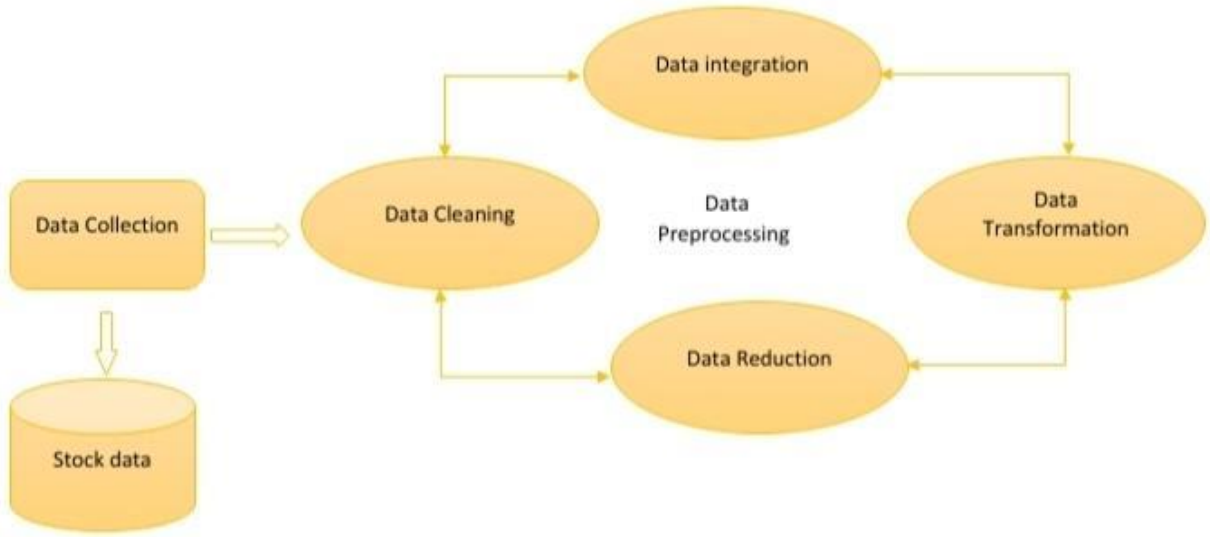


**Figure 4.2 Data Flow Diagram for Data Preparation**

Data preparation is the later stage of the ML lifecycle. Firstly, the data is collected from tingo data source, and later garbage data is cleaned and transformed into real-time machine learning projects to uncover insights or make predictions. Data preparation is also known as data "pre-processing," "data wrangling," "data cleaning," "data pre-processing," and "feature engineering". It is the later stage of the machine learning lifecycle, which comes after data collection.

Here, we collecting historical stock market data, such as stock prices, volumes, and other relevant financial indicators. Cleaning and pre-processing the data to handle missing values, outliers, and any other inconsistencies. Feature engineering, which involves creating new features or transforming existing ones to improve the model's performance. Normalizing the data to ensure that all features have the same scale and distribution. Splitting the data into training, validation, and test sets. Creating a time series dataset by converting the data into a format that can be used by LSTM, such as a sequence of historical data with a fixed window size. Finally, the data is ready to be used for training the LSTM model.

### 4.2.3 Dataflow diagram for splitting the dataset

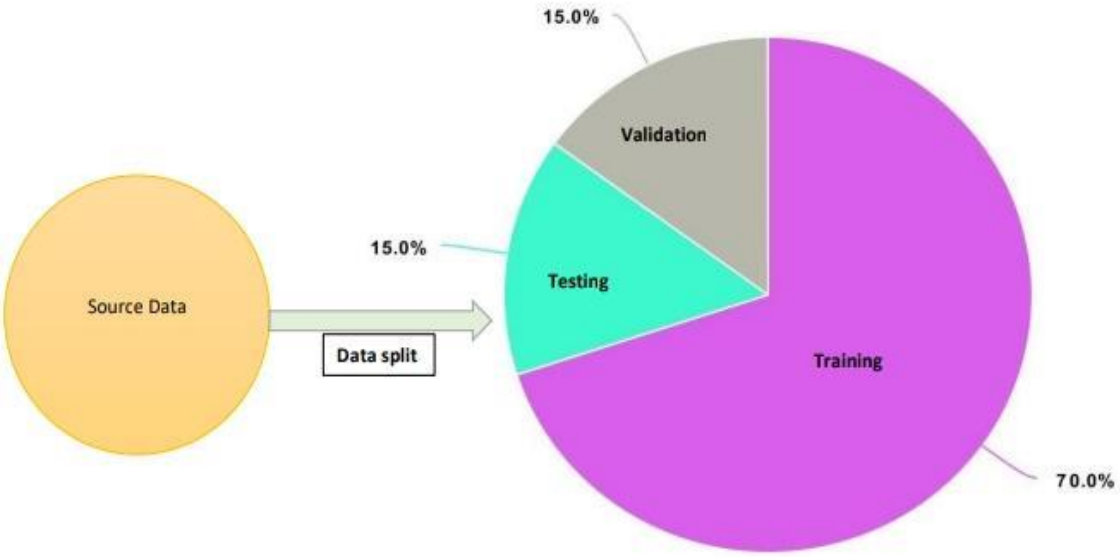


**Figure 4.3 Data Flow Diagram for Dataset splitting**

Data splitting is when data is divided into two or more subsets. Typically, with a two-part split, one part is used to evaluate or test the data and the other to train the model. In machine learning, data splitting is typically done to avoid overfitting. That is an instance where a machine learning model fits its training data too well and fails to reliably fit additional data.

Here, we training the set is used to train by LSTM model. Typically, this set contains the largest portion of the data, around 80%. The validation set is used to evaluate the model's performance during training and to fine-tune the model's hyperparameters. This set typically contains around 10% of the data. The test set is used to evaluate the model's performance on unseen data after training. This set typically contains around 10% of the data. It is important to note that the data in the test set should not be used in any way during the model training or hyperparameter tuning process, as this would lead to overfitting and artificially high performance scores. Another way to split dataset is using Time series splitting technique, where data is split into training and validation based on time, this is useful when data is time dependent.

### 4.2.4 Dataflow diagram for loading dataset into the model

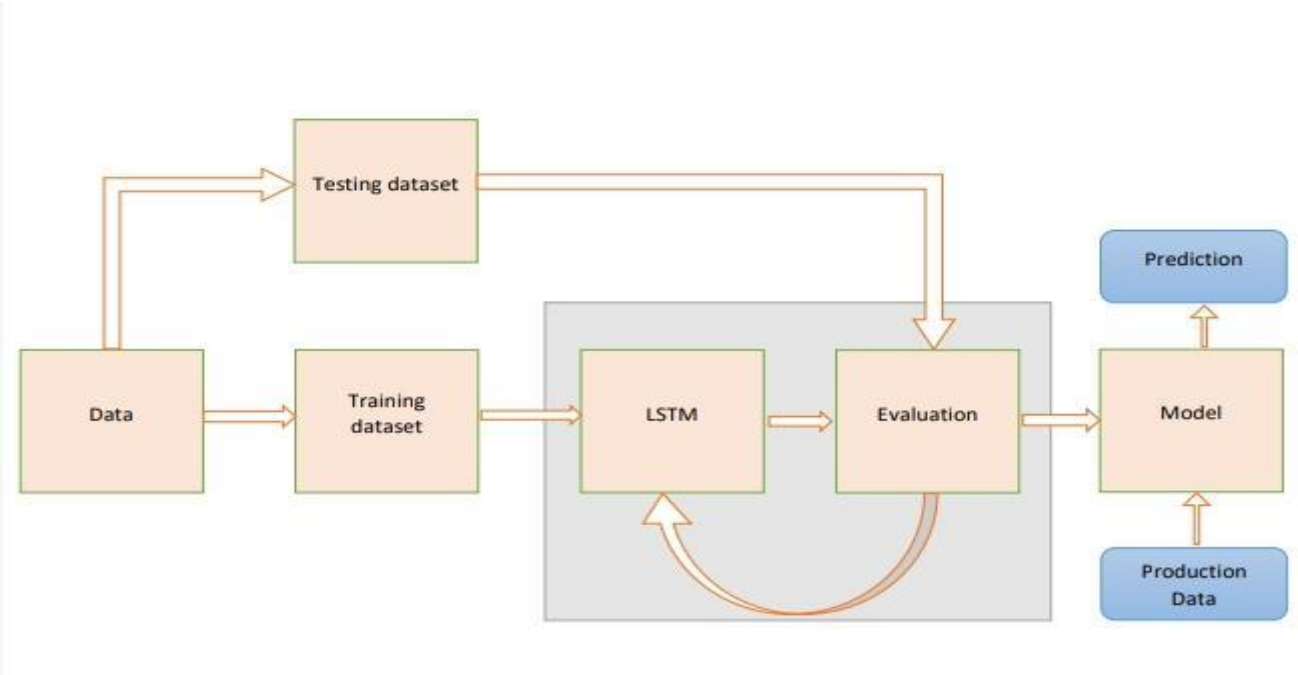


**Figure 4.4 Data Flow Diagram for Loading dataset into the model**

Machine Learning algorithms enable the machines to make predictions and solve problems on the basis of past observations or experiences. These experiences or observations an algorithm can take from the training data, which is fed to it. Further, one of the great things about ML algorithms is that they can learn and improve over time on their own, as they are trained with the relevant training data.

Once the model is trained enough with the relevant training data, it is tested with the test data. We can understand the whole process of training and testing in three steps, which are as follows:

1. **Feed:** Firstly, we need to train the model by feeding it with training input data.
2. **Define:** Now, training data is tagged with the corresponding outputs (in Supervised Learning), and the model transforms the training data into text vectors or a number of data features.
3. **Test:** In the last step, we test the model by feeding it with the test data/unseen dataset. This step ensures that the model is trained efficiently and can generalize well.

### 4.2.5 Dataflow diagram for predicting the accuracy of the model and visualize the outcome

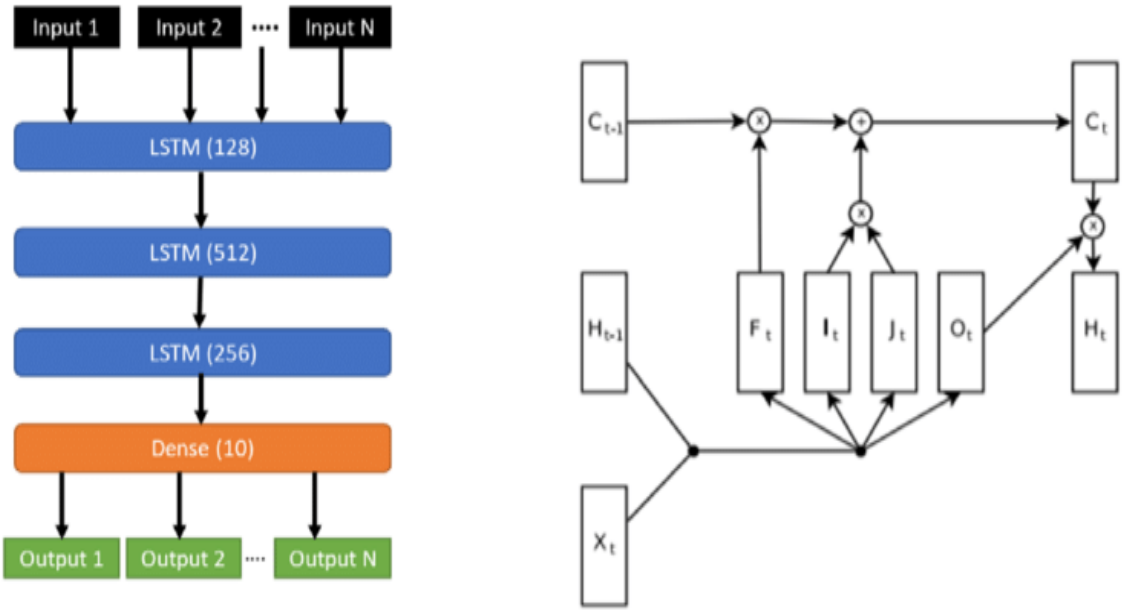


**Figure 4.5 Data Flow Diagram for Predicting and visualizing the model**

Now that we have completed training, let us see if the network performed well. We can test the model on testing data and see if the prediction and the actual values overlap. Rather than computing loss between predicted and actual values, we can plot it. From the graph, we can see that prediction and the actual value When predicting the future, there is a good possibility that model output is uncertain to a great extent. The model's output is fed back into it as input. This causes the model's noise and uncertainty to be repeated and amplified. But still, we have created a model that gives us a trend of the graphs and also the range of values that might be in the future somewhat overlap.

A typical data flow for predicting and visualizing the model involves several steps:

**Data collection:** The stock data is collected, along with labels associated with required information.

**Data preprocessing:** The stock are preprocessed to ensure that they are consistent in size and format, and to remove noise or null value in the data .

**Feature extraction:** The preprocessed data are then processed to extract features that are relevant for stock prediction, such as the open and close price of the data during the trade.

**Model training:** The Machine Learning model, such as Stacked LSTM(Long-Short Term Memory), is trained on the extracted features and labels to learn the relationship between the features and the predict the data with dense value.

**Model evaluation:** In model evaluation, the trained model get tested with the separated test data to make the trained model to get efficient and to check if it required any further improvement in the model.

**Deployment:** The trained model is get deployed in the stock data, where it is used to predict the stock range in new data and match them with the identities in the dataset.

# CHAPTER 5

# IMPLEMENTATION

## 5.1 DATA COLLECTION, PREPARATION AND CLEANING

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes. The data collection component of research is common to all fields of study including physical and social sciences, humanities, business, etc. While methods vary by discipline, the emphasis on ensuring accurate and honest collection remains the same.

### 5.1.1 Tingo Dataset

Here the collection of data is done from https://www.tingo.com using Hilres.TingoApi.

### 5.1.2 Cleaning

In our dataset “AAPL.csv” there is no noise(process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset) in the data and there is no need for doing the cleaning the dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled. If data is incorrect, outcomes and algorithms are unreliable, even though they may look correct. So, we check the data information by “df.info()” it shows all the columns have “non-nullvalues” in the data and the data are in “object”, “float64”and “int64” datatype.

## 5.2 SPLITTING THE DATASET INTO TRAIN, VALIDATION AND TEST DATASET

The MinMaxScalar procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. So, here we split the dataset into 80-20% of form. In this the total dataset can be divided into 80% data is for training dataset

and the remaining 20%dataset if for testing dataset, this all done by using “MinMaxScalar” technique from sklearn library.

### 5.2.1 Need for splitting the dataset into Train and Test dataset

Splitting the dataset into train and test sets is one of the important parts of data pre-processing, as by doing so, we can improve the performance of our model and hence give better predictability. We can understand it as if we train our model with a training set and then test it with a completely different test dataset, and then our model will not be able to understand the correlations between the features. In this, we split the data by “feature_range” and fit the data in “df1”

## 5.3 LOADING THE TRAINED AND TEST DATASET INTO THE ML MODEL

### 5.3.1 Reshaping the data

As LSTM only works with multidimensional data here we are reshaping the dataset into 3 dimensional data by using reshape() function. The input layer expects a 3D array of data when fitting the model and when making predictions, even if specific dimensions of the array contain a single value. So, we reshape the data then stored the data in “X_train” and “X_test”

### 5.3.2 Importing Tensorflow / Keras

TensorFlow's flexible architecture allows for the easy implementation of a wide range of machine learning models, including those with complex architectures and large amounts of data, making it well-suited for prediction tasks. Additionally, TensorFlow provides a variety of tools for visualizing and debugging models, as well as for optimizing their performance, making it a popular choice among researchers and practitioners in the field of machine learning. Using Tensorflow and keras we fitted the dataset into the LSTM model and make prediction of train and test dataset. Here we using the tensorflow by importing Sequential, Dense and LSTM from “tensorflow.keras.layers”. Then prepare the model and the fit the model in the trained data.

## 5.4 TEST THE MODEL WITH THE TRAINED DATASET FOR PREDICTING THE STOCK VALUE

Testing the model for stock market prediction involves evaluating the performance of a predictive algorithm by comparing its predictions to actual market data. This can be done through various methods such as backtesting, where the algorithm is tested on historical market data, or forward testing, where the algorithm is tested on current market data. The goal of the testing is to determine the accuracy and reliability of the algorithm in predicting future stock market movements. Additionally, testing can also be used to identify and address any bugs or issues in the algorithm. nother approach is to use a technique called cross-validation, where the data is split into several subsets, and the model is trained and tested on different subsets. This helps to reduce the chance of overfitting, which occurs when a model is too closely fit to the training data.

Another technique is called backtesting which is used to see how well the model would have performed in the past when applied to historical market data. The predicted values are compared against the actual historical data and the performance of the model is evaluated based on how well it predicts the past trends.

It's also important to evaluate the model's performance using metrics such as accuracy, precision, recall, and F1 score, which can give an overall picture of how well the model is performing.In summary, a combination of the above techniques should be used to test the performance of a machine learning model for stock market prediction.

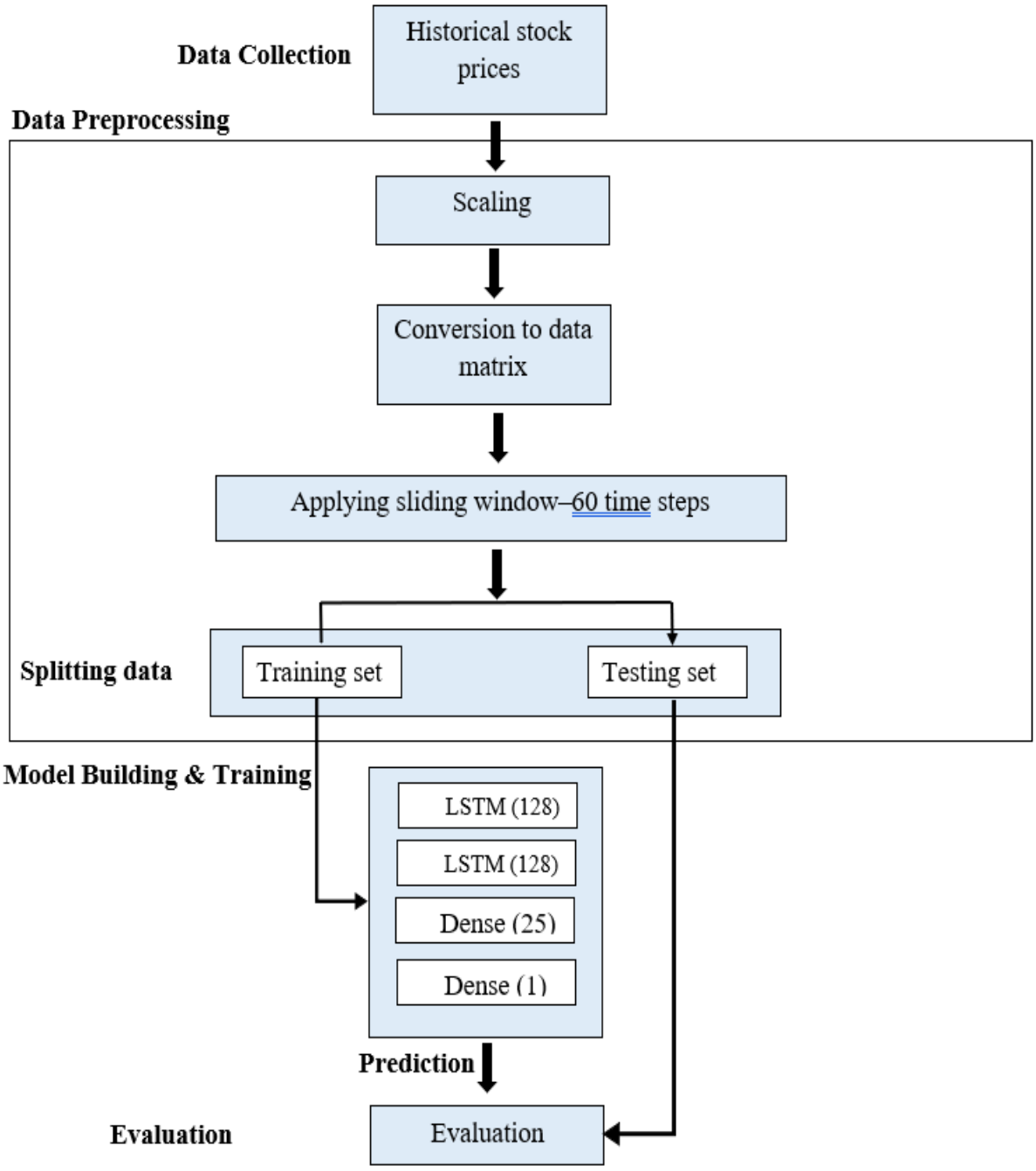


**Figure 5.1 Testing the model for predicting the Stock Value**

# CHAPTER 6

# CONCLUSION AND FUTURE ENHANCEMENT

## 6.1 CONCLUSION

stock market prediction is a complex task that involves analyzing a wide range of factors and data. While past performance does not guarantee future results, understanding historical trends and patterns can be helpful in making informed investment decisions. However, it is important to remember that the stock market is inherently unpredictable and subject to a variety of risks, so investors should always conduct their own research and consider their own risk tolerance before making any investment decisions. It is also important to note that there is no single method or algorithm that can accurately predict the stock market with 100% accuracy, so investors should be aware of the limitations of any predictions they may encounter. Software testing for stock market prediction involves evaluating the performance of a predictive algorithm by comparing its predictions to actual market data. This can be done through various methods such as back testing, where the algorithm is tested on historical market data, or forward testing, where the algorithm is tested on current market data. The goal of the testing is to determine the accuracy and reliability of the algorithm in predicting future stock market movements. Additionally, testing can also be used to identify and address any bugs or issues in the algorithm.

## 6.2 FUTURE ENHANCEMENT

Stock market prediction can be enhanced in various ways to improve its performance, such as:

- Incorporating more data, such as economic indicators, news articles, and social media sentiment, to provide a more comprehensive view of the market.
- Using more advanced architectures, such as attention mechanisms, to better capture the dependencies between different inputs.
- Incorporating more fine-tuning technique such as using different window size, using different input features, using different normalization techniques, etc.
- Using more advanced training techniques such as transfer learning, meta-learning, and adversarial training to improve the generalization ability of the model.

# CHAPTER 7

# REFERENCES


1. Abidin, S. N. Z., & Jaffar, M. M. (2014).
    a. Forecasting share prices of small size companies in Bursa Malaysia using geometric Brownian motion. Applied Mathematics & Information Sciences, 8(1), 107.
2. A Deep Learning Framework for Financial Time Series Using Stack Autoencoders and Long-Short Term Memory" by Y. Zhang et al. (2017)
3. A Hybrid Deep Learning Model for Stock Price Prediction" by X. Liu et al. (2019)
4. A Survey of Machine Learning for Big Financial Data" by
    a. J. Peietal. (2019)"IEEE Transactions on Industrial Informatics"
5. Brewer, K. D., Feng, Y., & Kwan, C. C. (2012).
    a. Geometric Brownian motion, option pricing, and simulation: Some spreadsheet-based exercises in financial modeling. Spreadsheets in Education, 5(3), 4598.
6. Deep Learning for Stock Market Prediction" by S. Gao et al.: This paper presents a deep learning-based approach for stock market prediction, which uses a combination of long short-term memory (LSTM) and gated recurrent units (GRU) neural networks.
7. IEEE Transactions on Neural Networks and Learning Systems
8. Ladde, G. S., & Wu, L. (2009).
9. Development of modified geometric Brownian motion models by using stock price data and basic statistics. Nonlinear Analysis: Theory, Methods & Applications, 71(12), e1203-e1208.
10. Reddy, K., & Clinton, V. (2016).
    a. Simulating stock prices using geometric Brownian motion: Evidence from Australian companies. Australasian Accounting, Business and Finance Journal, 10(3), 23-47.
11. Stock Market Forecasting Using a Hybrid ARIMA and Support Vector Machine Model" by C. K. Phua et al.: This paper presents a hybrid model that combines ARIMA and support vector machines (SVMs) for stock market prediction.
12. Stock Market Forecasting Using Machine Learning Algorithms" by S. Raschka and V. Mirjalili: This paper discusses the use of various machine learning algorithms, including decision trees, random forests, and support vector machines, for stock market prediction.

13. Stock Market Prediction Using Technical Analysis and Neural Networks" by R.J. Hyndman and A.B. Koehler (2006)

## Web References

1. https://ieeexplore.ieee.org/document/9642681
2. https://ieeexplore.ieee.org/document/9791640
3. https://link.medium.com/D6oXSzYQAub

https://www.researchgate.net/publication/264671556_Efficient_Machine_Learning_Techniques_for_Stock_Market_Prediction#:~:text=We%20have%20classified%20different%20techniques,TDNN%2C%20ICA%2DBPN

# APPENDIX 1 - SAMPLE CODE

```
#Data collection using tingo with a reference key

import requests

headers = {

    'Content-Type': 'application/json'

    }

requestResponse =
requests.get("https://api.tiingo.com/api/test?token=7d9ae600455f1b60063e5e2742667e250846caef",   headers=headers)

print(requestResponse.json())

# Importing pandas_datareader

import pandas_datareader as pdr

key="7d9ae600455f1b60063e5e2742667e250846caef"

#Accessing the data

df = pdr.get_data_tiingo('AAPL', api_key=key)

#Loading the dataset

df.to_csv('AAPL.csv')

#Importing pandas libraries

import pandas as pd

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

df

#Viewing the data format
```

```
df.head()

df.tail()

# reset the index using reset_index()

df1=df.reset_index()['close']

df1

#Importing matplotlib for visualization

import matplotlib.pyplot as plt

plt.plot(df1)

#Importing numpy libraries

import numpy as np

df1

# Importing MinMaxScaler using sklearn

from sklearn.preprocessing import MinMaxScaler

scaler=MinMaxScaler(feature_range=(0,1))

#As lstm supports only for higher dimensional , reshaping is done using reshape()

df1=scaler.fit_transform(np.array(df1).reshape(-1,1))

print(df1)

# Defining the training and testing data size

training_size=int(len(df1)*0.65)

test_size=len(df1)-training_size

train_data,test_data=df1[0:training_size,:],df1[training_size:len(df1),:1]

#Printing the train_data

training_size,test_size
```

```
train_data

import numpy

# convert an array of values into a dataset matrix

def create_dataset(dataset, time_step=1):

        dataX, dataY = [], []

        for i in range(len(dataset)-time_step-1):

                a = dataset[i:(i+time_step), 0]   ###i=0, 0,1,2,3-----99   100

                dataX.append(a)

                dataY.append(dataset[i + time_step, 0])

        return numpy.array(dataX), numpy.array(dataY)


# reshape into X=t,t+1,t+2,t+3 and Y=t+4

time_step = 100

X_train, y_train = create_dataset(train_data, time_step)

X_test, ytest = create_dataset(test_data, time_step)

print(X_train.shape), print(y_train.shape)

# reshape input to be [samples, time steps, features] which is required for LSTM

X_train =X_train.reshape(X_train.shape[0],X_train.shape[1] , 1)

X_test = X_test.reshape(X_test.shape[0],X_test.shape[1] , 1)

#Importing Sequential,Dense and LSTM model and layers from tensorflow.keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import LSTM
```

```
#assinging the moldel as Sequential()

model=Sequential()

model.add(LSTM(50,return_sequences=True,input_shape=(100,1)))

model.add(LSTM(50,return_sequences=True))

model.add(LSTM(50))

model.add(Dense(1))

model.compile(loss='mean_squared_error',optimizer='adam')

#  Getting summary of the model

model.summary()

#Fitting the train and validation_data to the model

model.fit(X_train,y_train,validation_data=(X_test,ytest),epochs=100,batch_size=64,verbose=1)

import tensorflow as tf

tf._version_

### Lets Do the prediction and check performance metrics

train_predict=model.predict(X_train)

test_predict=model.predict(X_test)

##Transformback to original form

train_predict=scaler.inverse_transform(train_predict)

test_predict=scaler.inverse_transform(test_predict)

### Calculate RMSE performance metrics

import math

from sklearn.metrics import mean_squared_error

math.sqrt(mean_squared_error(y_train,train_predict))
```

```
### Test Data RMSE

math.sqrt(mean_squared_error(ytest,test_predict))

### Plotting

# shift train predictions for plotting

look_back=100

trainPredictPlot = numpy.empty_like(df1)

trainPredictPlot[:, :] = np.nan

trainPredictPlot[look_back:len(train_predict)+look_back, :] = train_predict

# shift test predictions for plotting

testPredictPlot = numpy.empty_like(df1)

testPredictPlot[:, :] = numpy.nan

testPredictPlot[len(train_predict)+(look_back*2)+1:len(df1)-1, :] = test_predict

# plot baseline and predictions

plt.plot(scaler.inverse_transform(df1))

plt.plot(trainPredictPlot)

plt.plot(testPredictPlot)

plt.show()

len(test_data)

x_input=test_data[341:].reshape(1,-1)

x_input.shape

temp_input=list(x_input)

temp_input=temp_input[0].tolist()

temp_input
```

```
# demonstrate prediction for next 10 days
from numpy import array
lst_output=[]
n_steps=100
i=0
while(i<30):
    if(len(temp_input)>100):
        #print(temp_input)
        x_input=np.array(temp_input[1:])
        print("{} day input {}".format(i,x_input))
        x_input=x_input.reshape(1,-1)
        x_input = x_input.reshape((1, n_steps, 1))
        #print(x_input)
        yhat = model.predict(x_input, verbose=0)
        print("{} day output {}".format(i,yhat))
        temp_input.extend(yhat[0].tolist())
        temp_input=temp_input[1:]
        #print(temp_input)
        lst_output.extend(yhat.tolist())
        i=i+1
    else:
        x_input = x_input.reshape((1, n_steps,1))
        yhat = model.predict(x_input, verbose=0)
```

```
        print(yhat[0])
        temp_input.extend(yhat[0].tolist())
        print(len(temp_input))
        lst_output.extend(yhat.tolist())
        i=i+1
print(lst_output)
len(lst_output)
day_new=np.arange(1,102)
day_pred=np.arange(102,132)
import matplotlib.pyplot as plt
len(df1)
# Prediction for next 10 days
plt.plot(day_new,scaler.inverse_transform(df1[1158:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
df3=df1.tolist()
df3.extend(lst_output)
plt.plot(df3[1200:])
df3=scaler.inverse_transform(df3).tolist()
plt.plot(df3)
```

# APPENDIX 2 – SCREENSHOTS

## INPUT

| | symbol | date | close | high | low | open | volume | adjClose | adjHigh | adjLow | adjOpen | adjVolume | divCash | splitFactor |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | AAPL | 2018-01-23 00:00:00+00:00 | 177.04 | 179.44 | 176.82 | 177.300 | 31702531 | 42.031743 | 42.601536 | 41.979512 | 42.093471 | 126810124 | 0.0 | 1.0 |
| 1 | AAPL | 2018-01-24 00:00:00+00:00 | 174.22 | 177.30 | 173.20 | 177.250 | 50562257 | 41.362236 | 42.093471 | 41.120074 | 42.081600 | 202249028 | 0.0 | 1.0 |
| 2 | AAPL | 2018-01-25 00:00:00+00:00 | 171.11 | 174.95 | 170.53 | 174.510 | 39661804 | 40.623879 | 41.535548 | 40.486179 | 41.431086 | 158647216 | 0.0 | 1.0 |
| 3 | AAPL | 2018-01-26 00:00:00+00:00 | 171.51 | 172.00 | 170.06 | 172.000 | 37121805 | 40.718845 | 40.835178 | 40.374595 | 40.835178 | 148487220 | 0.0 | 1.0 |
| 4 | AAPL | 2018-01-29 00:00:00+00:00 | 167.96 | 170.16 | 167.07 | 170.160 | 48434424 | 39.876026 | 40.398336 | 39.664727 | 40.398336 | 193737696 | 0.0 | 1.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1253 | AAPL | 2023-01-13 00:00:00+00:00 | 134.76 | 134.92 | 131.66 | 132.030 | 57809719 | 134.760000 | 134.920000 | 131.660000 | 132.030000 | 57809719 | 0.0 | 1.0 |
| 1254 | AAPL | 2023-01-17 00:00:00+00:00 | 135.94 | 137.29 | 134.13 | 134.830 | 63646627 | 135.940000 | 137.290000 | 134.130000 | 134.830000 | 63646627 | 0.0 | 1.0 |
| 1255 | AAPL | 2023-01-18 00:00:00+00:00 | 135.21 | 138.61 | 135.03 | 136.815 | 69672800 | 135.210000 | 138.610000 | 135.030000 | 136.815000 | 69672800 | 0.0 | 1.0 |
| 1256 | AAPL | 2023-01-19 00:00:00+00:00 | 135.27 | 136.25 | 133.77 | 134.080 | 58280413 | 135.270000 | 136.250000 | 133.770000 | 134.080000 | 58280413 | 0.0 | 1.0 |
| 1257 | AAPL | 2023-01-20 00:00:00+00:00 | 137.87 | 138.02 | 134.22 | 135.280 | 80223626 | 137.870000 | 138.020000 | 134.220000 | 135.280000 | 80223626 | 0.0 | 1.0 |

1258 rows × 14 columns

**Figure A.2.1 Input dataset**

By importing the **pandas** library which is used to manipulate the data sets, i.e. to edit, change, and replace particular elements of a DataFrame class object. Saving the csv file and then reloading it using **read_csv(). ( )** function to view the entire dataset which displays index size, symbol, date, close, high, low, open, volume, adjClose, adjHigh, adjLow, adjVolume, divCash, splitfactor.

**OUTPUT**

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1258 entries, 0 to 1257
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   symbol       1258 non-null   object
 1   date         1258 non-null   object
 2   close        1258 non-null   float64
 3   high         1258 non-null   float64
 4   low          1258 non-null   float64
 5   open         1258 non-null   float64
 6   volume       1258 non-null   int64
 7   adjClose     1258 non-null   float64
 8   adjHigh      1258 non-null   float64
 9   adjLow       1258 non-null   float64
 10  adjOpen      1258 non-null   float64
 11  adjVolume    1258 non-null   int64
 12  divCash      1258 non-null   float64
 13  splitFactor  1258 non-null   float64
dtypes: float64(10), int64(2), object(2)
memory usage: 137.7+ KB
```

**Figure A.2.2 Dataset Information**

It gives the complete information about the datatype in the “AAPL.csv”file. The datatype can be viewed by using **“describe( )”** function. It shows that the file contains fourteen columns (symbol, date, close, high, low, open, volume, adjClose, aadjLow, adjOpen, adjVolume, divCash and splitFactor) and the it have object, float64 and int64 datatype. The index range have 1258 entries that have 0 to 1257.

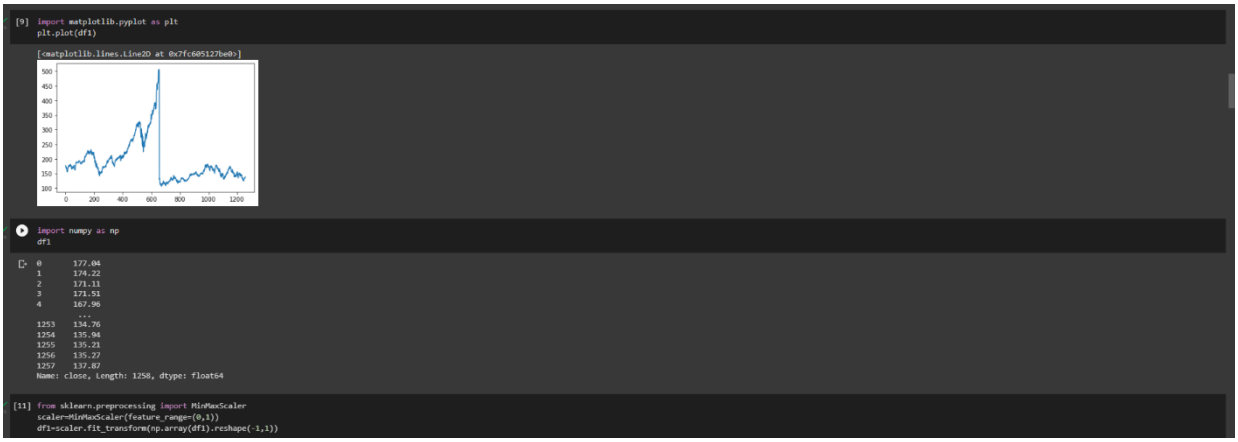


**Figure A.2.3 Plotting the Close value in the dataset**

By using matplotlib here we plot the **"close**" value in the entire dataset which is stored in "df1". For plotting the df, we use **plt.plot(df1)** which can plot the closing value in the AAPL.csv file of the stock value.

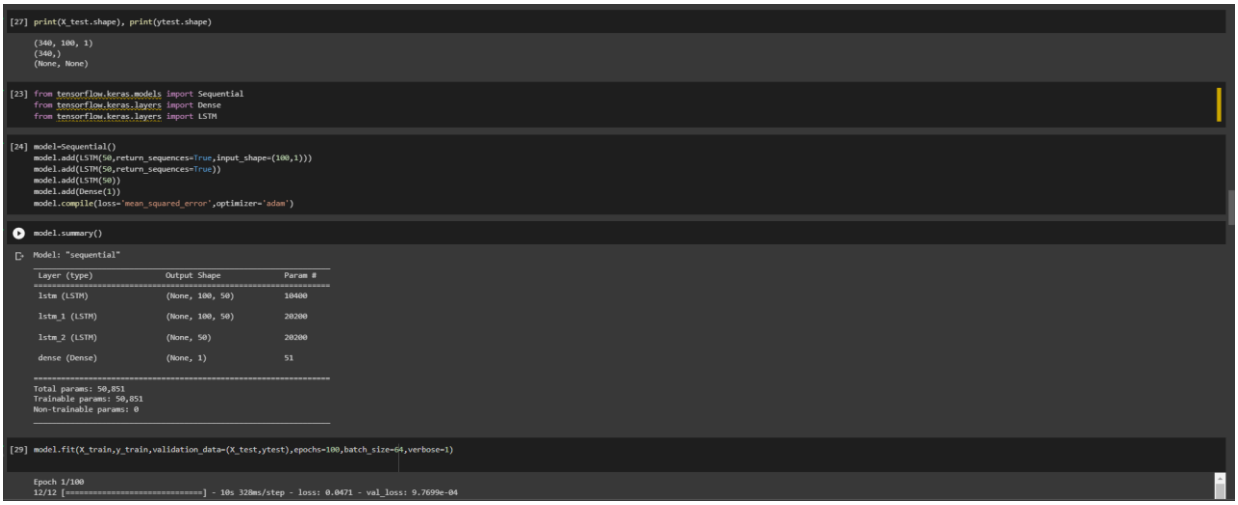


**Figure A.2.4 LSTM model**

In this it show the information of the created LSTM model. This can be viewed by **"summary( )"** function. Here we build the LSTM model with the help of **"sequential( )"** function. The created model contain four layers ( lstm (LSTM), lstm_1(LSTM), lstm_2(LSTM) and dense(Dense)) , total params of 50,851 and trainable_params of 50,851.

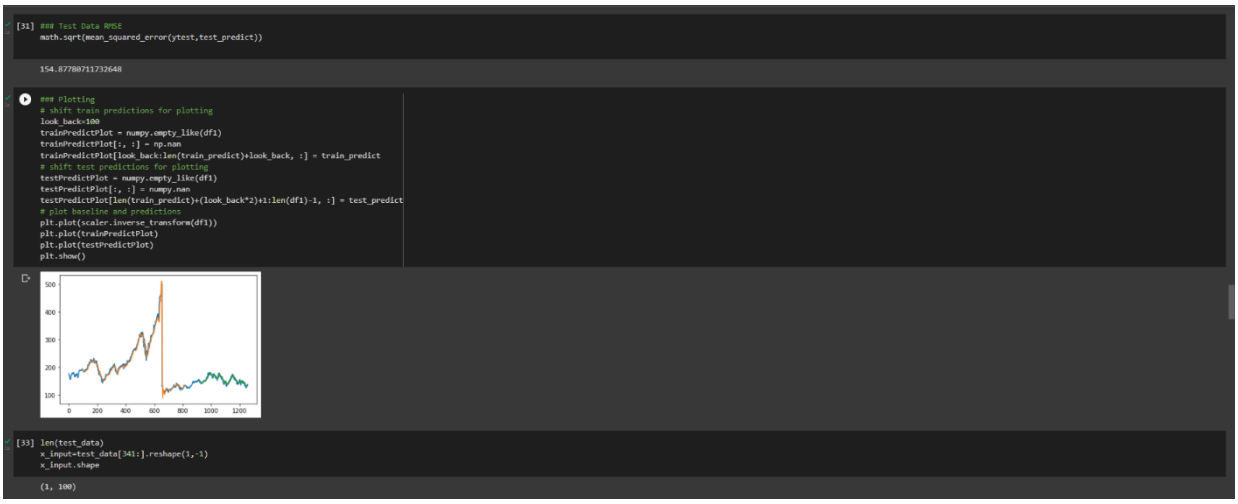

**Figure A.2.5 Plotting the output-1**

Predicting x_train and x_test, transforming the predicted set into its original form using **scaler.inverse_transform**. For plotting, we use graph to predict the output. In the output graph the individual colors denotes individual features. Green colour indicates the predicted output for the test data, orange colour indicates the predicted output for the training data and blue colour indicates the complete dataset.

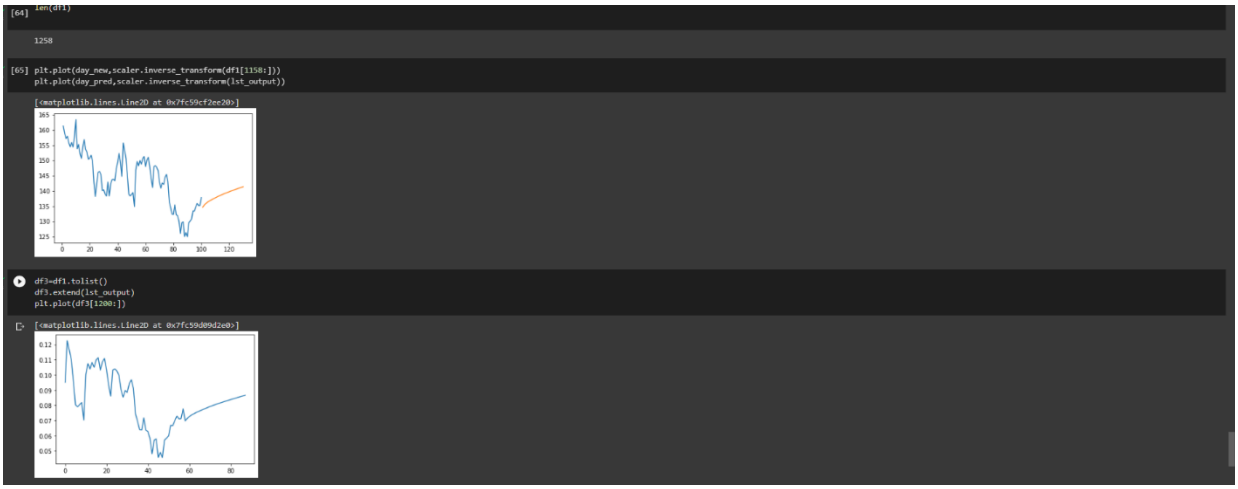


**Figure A.2.6 Plotting the output-2**

Here we predicting the future days by taking the input as the previous 100 days and then reshaping it and converting it into list using **list().** Demonstrating the prediction of next 30 days by passing the data into the model and making the prediction and creating a while loop which executes until it completes 30 loops and gives the predicted output. Plotting the predicted output using matplotlib and in the output graph we get orange colour line which is the prediction for the 30 days. To visualize the whole prediction we created new variable df3 for combining the dataset by **"df3=scaler.inverse_transform(df3).tolist() plt.plot(df3)"** then, the output graph which displays the whole prediction of the stock value.