A PROJECT REPORT ON

STOCK MARKET PREDICTION USING BIDIRECTIONAL RNN AND ARIMA

Major project submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY
(2018-2022)

BY

S.V.N Sai Pratheek	**18241A12B4**
M.Manohar	**18241A1296**
D.Bala Yashwanth	**18241A1276**
G.Baradwaj	**18241A1278**

Under the Esteemed guidance of

DR.N.Rajasekhar

Professor**, Information Technology**

DEPARTMENT OF INFORMATION TECHNOLOGY
GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY
(AUTONOMOUS)

CERTIFICATE

This is to certify that is a bonafide record of Project work entitled “STOCK MARKET PREDICTION” done by S.V.N Sai Pratheek(18241A12B4),M.Manohar (18241A1296), D.BalaYashwanth(18241A1276), G.Baradwaj (1824A1278), students of B.Tech(IT) in the Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2018-2022 in the partial fulfillment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY from GRIET, Hyderabad.

Dr.N.Rajasekhar
(Internal Project Guide)

Dr. N. V. Ganapathi Raju
(Head of the Department)

(Project External)

ACKNOWLEDGEMENT

We take immense pleasure in expressing gratitude to our Internal guide **Dr.N.Rajasekhar, Professor,** Department of Information Technology, GRIET. We express our sincere thanks for his encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to Dr**. N. V. Ganapathi Raju** HOD IT Department our Project Co-coordinators **G.Vijendar Reddy** and **A.Srilakshmi**, for their constant support during the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Email: venkataprateek@gmail.com
Contact No: 8464836831
Address: Kukatpally, Hyderabad.

Email: mmanohar0111@gmail.com
Contact No:8978552525
Address: Kukatpally, Hyderabad

Email: balayashwanth73@gmail.com
Contact No:9100744189
Address: Chandanagar, Hyderabad.

Email: baradwaj.gajawada@gmail.com
Contact No: 9603124130
Address: Bobbili veddi, Nizamabad

DECLARATION

This is to certify that the project entitled "**STOCK MARKET PREDICTION "** is a bonafide work done by us in partial fulfillment of the requirements for the award of the degree BACHELOR OF **TECHNOLOGY** IN INFORMATION TECHNOLOGY from Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad.

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

S.V.N Sai Pratheek	**18241A12B4**
M.Manohar	**18241A1296**
D.Bala Yashwanth	**18241A1276**
G.Baradwaj	**18241A1278**

TABLE OF CONTENTS

11. LIST OF FIGURES

ABSTRACT

In the present day, The stock market is a crucial profession or process for improving a person's income; it is one of the means for boosting a person's financial growth. So, what exactly is a stock exchange? It is an aggregation of buyers and sellers of stocks that represent corporate ownership.. The stock market may consist of securities listed on the public stock exchange and the stocks traded privately that are shared by private companies like Amazon, Reliance, and so on and so more. Our main aim is to predict the value of the share or stock of a specific company in the coming future. We take the help of Deep Learning to forecast the values of the stocks under deep learning; we use Arima Model and, where Arima stands for Autoregressive Integrated Moving Average algorithm and Bidirectional RNN model here, we take the live data of a particular company and apply the deep learning model to it in order to forecast the values of the stocks. With the help of the forecasted results, the end-user can sell or buy their stocks which get them profits. Finally, we will compare the above models and extract the final results which may help the end user.

Index Terms: Stock market, Deep learning, Forecast, Arima,Bidirectional RNN, Deep learning, Forecast, Model, Autoregressive Integrated Moving Average.

INTRODUCTION

1.1 Introduction to Project

In the modern age, stock investment has been a key source of income. Generally, stocks function in such a way that a person invests in a company's share and later sells it to a buyer in such a way that he makes a profit. This is a process where deep analysis of historical data plays a significant role in obtaining profits or projecting the future status of shares of a specific firm. In such circumstances, deep learning or machine learning enables us to make the best decisions.

As the final output will depend on the properties of the data and the model's behavior on the provided data, we may conclude that the dataset plays a vital part in deep learning as the data may influence the final outcomes of the model. In this project, we use Tiingo to acquire live data that consists of numerous variables. The data will be separated into two categories: test data and train data so that the constructed model can be trained and tested for prediction purposes.

The application's algorithm, The Bidirectional Rnn, is capable of remembering data and ultimate outcomes over time along with we use the time series model, the Arima. Finally, the graphs will be plotted and compared between the models to depict the forecast for the following n days and the model's behavior for the supplied data..

1.2 Existing System:

Stocks are currently anticipated based on random assumptions. Aside from that, there are a few methods for estimating a firm's market, such as Support Vector Machine and Random Forest. Customers sometimes utilize trial and error approaches while selling the firm's stocks or shares, which might result in a significant loss.

1.3 Proposed System

Our primary goal is to forecast a specific company's stock and share prices using live data, which we can do with the help of an API called Tiingo. We utilize live data because it allows us to closely examine the algorithms' behavior on dynamic data and build an effective model of it. We use the Bidirectional Rnn and ARIMA algorithm to extract better insights for understanding stocks.

2. REQUIREMENT ENGINEERING

2.1 Hardware Requirements

- Processor – Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz (64-bit OS).
- Memory – 8GB RAM (Higher specs are recommended for high performance)

2.2 Software Requirements

- Windows 10
- Anaconda navigator
- Python ide (Jupyter)
- tiingo

3. LITERATURE SURVEY

In their study, Aparna Nayak, M. M. Manohara Pai, and Radhika M. Pai indicate that the past few years have seen the bulk of individuals investing within the stock exchange to get income. A high risk of losing all invested funds also exists for the investor. so as to know future market trends, one needs an efficient predictive model. Despite the very fact that there are many market-trend forecasting methods, none of them are accurate. In an effort to create an efficient predictive model of the stock exchange, subsequent day's trend is predicted. This is often done by considering the continual ups and downs, the quantity trades, and other patterns. Supported open-source stock exchange data available 24 hours per day and also including the sentiment of the corporate, a model has been developed and tested. With the dataset in hand, Decision Boosted Tree outperforms Support Vector Machine and Logistic Regression.

Jingyi Shen and M. Omair Shafiq prepared three parts of their work: data extraction and pre-processing of the Chinese stock exchange dataset, feature engineering, and a model of stock price trend prediction supported the long STM (LSTM). during this project, the researchers collected, cleaned, and structured 2 years' worth of Chinese stock exchange data. a replacement algorithm component was developed to require advantage of the necessity for feature extension employed by real-world investors, and it proved to be effective. FE and RFE approaches with principal component analysis (PCA) were combined to construct a feature engineering procedure that incorporated recursive feature elimination, it's effective and efficient at an equivalent time. It outperforms all leading models in most related papers by combining the feature engineering procedure with an LSTM prediction model. The results of this work also are comprehensively evaluated. The researchers conclude that by comparing the foremost commonly used machine learning models with the proposed LSTM model under the feature engineering a part of the proposed system, they're likely to get variety of heuristics for future research in both technical and financial domains. As against the previous works, their approach is exclusive customization because instead of simply proposing an LSTM model, they proposed a deep learning prediction system that's fine-tuned and customized also as utilizing extensive feature engineering combined with LSTM to form predictions.

Researchers Subba Rao, Srinivas, and Krishna Mohan present a comprehensive review of and comparative analysis of stock exchange prediction parameters. The authors address stock exchange performance and trends also as technical indicators. stock exchange forecasting systems enable investors to maximize their returns. a completely unique approach is employed during this study to assist predict the results of stocks, which suggests we'll combine two or more methods to make a completely unique approach.

Troy J. Strader, John J. Rozycki, Thomas H. Root, Yu-Hsiang (John) Huang,This study analyzes this literature to identify directions for future analysis on stock market prediction exploitation machine learning. several conclusions ar usually drawn regarding current analysis throughout this space, given the ML-related systems, problematic contexts, and outcomes delineated in every hand-picked paper, and therefore the compartmentalization classes conferred higher than. to begin with, mil ways have a very robust regard to the prediction issues they are related to . associate degree analogy may well be the distinction of activity technology (Goodhue and Thompson, 1995), throughout that system performance is set by the appropriate match between activities associate degreed technologies Artificial neural networks ar best used to predict indicant values Vector machines ar ideal for classification tasks like decisive stock market outlook A genetic formula uses an organic process approach to drawback determination to identify higher quality system inputs, or to predict that actions incorporate into a portfolio to maximise their come. showed that the ways ar usually applied effectively, the distinctive applications of the ways had limitations. form of those limitations ar usually relieved by machine techniques. emulsified learning. there is a hurdle with complicated systems as a result of a has some lubricated p, the systems aren't from now on helpful.

Jiake Li, withinside the discipline of clever prognostication, studies at the inventory alternate ought to be a heat material withinside the long run . inventory alternate modifications play associate crucial operate among the country's financial trends. Their studies examines the short-time amount fashion prognostication of shares supported capitalist sentiment extraction and compares the result of quite one records reassets at the accuracy of the version. a entire of parts of studies paintings is run among the article to resolve the above-noted drawback. because of its certain and beneficiant returns, inventory alternate prognostication area unit typically a long-time amount scenario of the capital marketplace. the event of prognostication ways has to boot

progressed the prognostication results. in a trial to use capitalist sentiment records to form bigger correct predictions at the inventory alternate, this paper establishes a inventory marketplace index prediction version supported through facts and deep learning. in a trial to alternate straightforward emotional capabilities among the emotional extraction method, CNN goes to be used on a datum version to extract deep emotional records. To boot enhance the prognosticative overall performance of the version, additional records reassets, like essential capabilities, area unit accessorial on the data offer level. Supported the results, the set of rules of the theme is feasible and economical, and it's ready to higher expect the modifications among the marketplace inventory marketplace index.

B. Uma Devi, D.Sundar, and Dr. P. AlliIn, In this document, the facts is accumulated from NSE.com. Five-12 months ancient records from 2007 to 2011 became taken into thought for analysis. The BoxJenkins technique is utilized to pick out the version. The AICBIC check standards observe to the higher than facts to pick out the foremost effective version. The foremost effective version equations arise for all indexes. The accuracy of MAPE, PMAD, and one or two of mistakes is run to create clear the distinction among their various ancient records and consequently prognosticative records. This study infers chance funding choices or laws that guide the minimum % of mistakes performed via the general performance signs higher than. Future forecasts for every index over the following few years also are highlighted on this paper. it's hoped that a bigger progressive technique area unit typically applied to herald hidden facts roughly the inventory market.

Fahim Faisal, In this study, The Box Jenkins technique to time assortment prediction became used to selecting the foremost wonderful prediction model. economics statement fashions typically contains a machine of relationships among variables of interest, and thus the relationships square measure inferred from the to be had information. However, in apply, show that a straightforward rule of thumb, that is AN extrapolation from a widowed dataset, might even be used to create pretty properly predictions. because of the shortage of studies on inflation statement in Asian country, statement is AN important a section of the decision-making manner of economic regulators and policymakers, and in addition studies on this region is needed.

In this paper, Kailash Chandra Pradhan and K. Sham Bhat investigate the valuation, causality, and forecasts of S & P CNX groovy futures. consistent with the results of the unit root check, the bang-up index and bang-up futures don't appear to be stationary at those levels. The trend is interrupted by the primary distinction, however. there's a long-run correlation between spot and futures prices supported the results of the cointegration check. Therefore, you may be able to use the Vector Error Correction Model (VECM) to analysis short dynamics and worth fluctuations among the 2 markets. The results of the vector correction model of JohanSen (VECM) discovered that the exchange lands up among the commodities exchange, and spot prices attended verify new data previous futures costs. info and integration info between spots and futures costs may be accustomed manufacture extra correct worth forecasts. As a result, the commodity exchange indicates that the commodities exchange and thus the commodity exchange square measure used as a result of the foremost marketplace for value discovery. The leading role of the commodities exchange is weakened that consider company-specific announcements (Mukherjee and MISHRA, 2006). The commodities exchange is on the immature stage started in Gregorian calendar month 2000. therefore far, many dealers and investors square measure confused for this new market. Derivatives ar difficult. Payment and risks that ar much more powerful than the consumer and merchandiser facing the exchange. New dealers and investors ar still starting futures markets. Therefore, the leader of spot mark futures markets. The results together show that the vector autoregressive model (VECM) works higher on a post-sample basis than the univariate autoregressive integrated moving average model (ARIMA) and additionally the vector autoregressive model (VAR). .. This result clearly shows that it's important to ponder the long-run relationship between futures and spot prices once predicting future spot costs.

K.K.Sureshkumar and N.M. Elango explored and applied neural classification functions using the Weka tool in this article. Based on correlation coefficients used to compare different predictive functions, isotonic regression functions are more accurate than Gaussian processes, minimum mean squares, linear regressions, multilayer perceptrons, tempo regressions, etc. Results were positive. SMO and linear regression. With this analysis, you can reduce the likelihood of making errors in forecasting future stock prices. As a result, investors have a greater chance of accurately predicting stock prices. Therefore, the stock market profit is increased since the rate of error is reduced. This paper provides information about the performance levels of different aspects of these features in a volatile stock market, such as the

Indian stock market. An investor and regulator will be guided by a high degree of accuracy in predicting market direction. Their hypothesis on this tool is that it can be used to study market forecasts and their performance indicators.

4. TECHNOLOGY

4.1 ABOUT PYTHON

Python is a programming language, because of this that each human being and machine can apprehend it. Dutch programmer Guido van Rossum advanced Python to clear up a number of the issues he encountered whilst the use of different laptop languages.
Python can end up as a high-degree laptop-orientated language for widespread interpreted programming. Python is a programming language advanced with the aid of using Guido van Rossum and at the start launched in 1991. It has a layout philosophy that prioritizes code clarity and syntax, permitting programmers to create specific ideas with fewer traces of code and extra space. Its shape makes it clean for application domestically and widespread.

Python has a dynamic kind machine and automated reminiscence management. It has a crucial and tremendous preferred library, and helps numerous programming paradigms, along with object-orientated, mandatory, practical, and procedural.
Python interpreters may be utilized in numerous running systems. Python reference, C Python, and nearly all implementations are open supply software programs with a community-pushed improvement strategy. The Python Software Foundation is a non-income employer operated with the aid of using C Python.

4.2 YOU CAN USE PYTHON FOR PRETTY MUCH ANYTHING

Knowing Python is beneficial because it is a general-purpose language that may be used for various purposes. The following are a few of the most popular Python applications:

- Data science
- Scientific and mathematical computing
- Web development
- Computer graphics
- Basic game development
- Mapping and geography (GIS software)

4.3 PYTHON IS WIDELY USED IN DATA SCIENCE

Python's ecosystem is expanding each time, and it's becoming increasingly capable of statistical analysis.

It's the best balance of size and sophistication (in terms of OD data processing).

Python places a premium on efficiency and readability.

Python is a programming language used by programmers who want to do data analysis or use statistical approaches (and by devs that turn to data science).

Python clinical programs abound for some standards like facts visualization, system learning, herbal language processing, complex facts analysis, and more. Python is a tremendous device for clinical computing due to all of those features, and it is a tremendous opportunity for business merchandise like MatLab. The following are the maximum extensively used facts technological know-how libraries and tools

4.3.1 TIINGO

Tiingo is a financial data platform that makes high quality financial tools available to all. Tiingo has a REST and Real-Time Data API, which this library helps you to access. Presently, the API includes support for the following endpoints:

• Stock Market Ticker Closing Prices + Metadata. Data includes full distribution details and is validated using a proprietary EOD Price Engine.

• Curated news from top financial news sources + blogs. Stories are tagged with topic tags and relevant stock tickers by Tiingo’s algorithms.

4.3.2 BIDIRECTIONAL RNN

Long short-time period memory (LSTM) is an synthetic recurrent neural community (RNN) structure used withinside the discipline of deep learning. Unlike popular feedforward neural networks, LSTM has comments connections. It can system now no longer handiest unmarried

information points (along with images), however additionally whole sequences of information (along with speech or video). For example, LSTM is relevant to responsibilities along with unsegmented, linked handwriting popularity, speech popularity and anomaly detection in community visitors or IDSs (intrusion detection systems).

A not unusual place LSTM unit consists of a mobileular, an enter gate, an output gate and a forget gate. The mobileular recollects values over arbitrary time durations and the 3 gates adjust the waft of statistics into and out of the mobileular.

LSTM networks are well-proper to classifying, processing and making predictions primarily based totally on time collection data, on account that there may be lags of unknown period among essential activities in a time collection. LSTMs have been advanced to address the vanishing gradient trouble that may be encountered while schooling conventional RNNs. Relative insensitivity to hole duration is a bonus of LSTM over RNNs, hidden Markov fashions and different series getting to know strategies in several applications.

Stacked LSTM structure may be described as an LSTM version constructed from more than one LSTM layer. An LSTM layer above affords a chain output instead of an unmarried fee output to the LSTM layer below. Specifically, one output step with the enter time step, instead of one output time step for all enter time steps.

4.3.3 ARIMA

The time series forecasting technique ARIMA stands for Autoregressive Integrated Moving Average.Times Series Forecasting is a strategy that forecasts future values based on past values and corrects their lags (lagged forecast mistakes) from which it was formed.The ARIMA model is primarily based on three parameters: p(seasonality), d(trend), and q(quantity) (noise in the data).The order of the Autoregressive (AR) term is p. The number of lags refers to the number of lags to use as predictors.The order of the Moving Average (MA) word is q. It has to do with the number of lagged forecast errors that should be incorporated into the ARIMA Model.

5. DESIGN REQUIREMENT ENGINEERING

5.1 Unified Model Language

The unified model language is a general-purpose language used in software engineering to understand the design of a system; in other words, UML provides a standard way to visualize the system's design.

Several diagrams like use case diagram, class diagram, statechart diagram, sequence diagram, component diagram, and deployment diagrams in UML help us visualize or understand the system's design in each stage.

5.2 Use case Diagram

Use case diagrams are a diagrammatic representation of the user's interactions with the system. A use case diagram depicts various types of use cases, different types of system users and will often be joined by other types of diagrams. Circles and ellipses represent these use cases, and the actors in the use cases are used as stick-like figures.

Use case diagrams give us a drill into every possibility. In other words, using use case diagrams will help provide a higher-level view of the system that is going to be developed. Use case diagrams are the blueprint of the system which is going to be developed. It is also a type of behavioral diagram defined by and also created by use–case analysis

Use case diagrams are built in the early stage of the development, and developers often apply use case diagrams for the following purposes:

- Specify the context of the system.
- Get the requirements of the system.
- System architecture validation.
- Implement and generate test cases.
- Developed by analysts together with the respective domain experts.

In our project, we use two types of use case diagrams, namely:

- Use case diagram from the user point of view.
- Use case diagram from the system point of view.

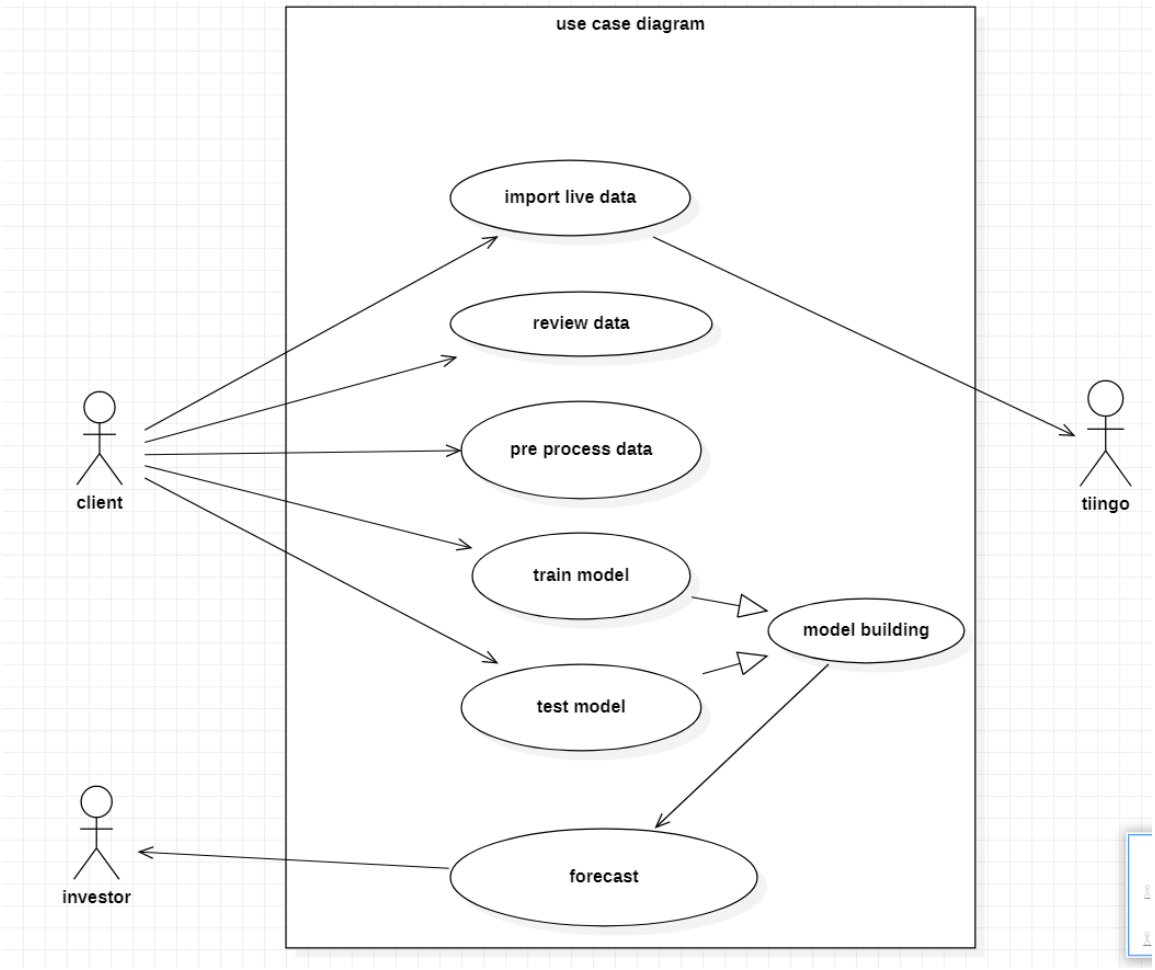

Fig 5.2: Use Case Diagram

5.3 Sequence Diagram

A series diagram, additionally termed an occasion diagram, facilitates constitute the float of messages withinside the gadget or application; including this additionally facilitates diverse dynamic aspects. It presentations verbal exchange among lifelines as a series of events, which took components withinside the run time. In UML, the lifelines are represented with the aid of using a rectangle(vertical bar), and the message bar is denoted with the aid of using a vertical dotted line that extends to the footer of the web page and is used for new release and branching.

The primary purpose of the sequence diagram is:

- To model interaction which is high level among the active objects of the applications.
- To model interaction among objects realizing the use case inside a collaboration.
- It either models certain instances of interaction or generic interactions.

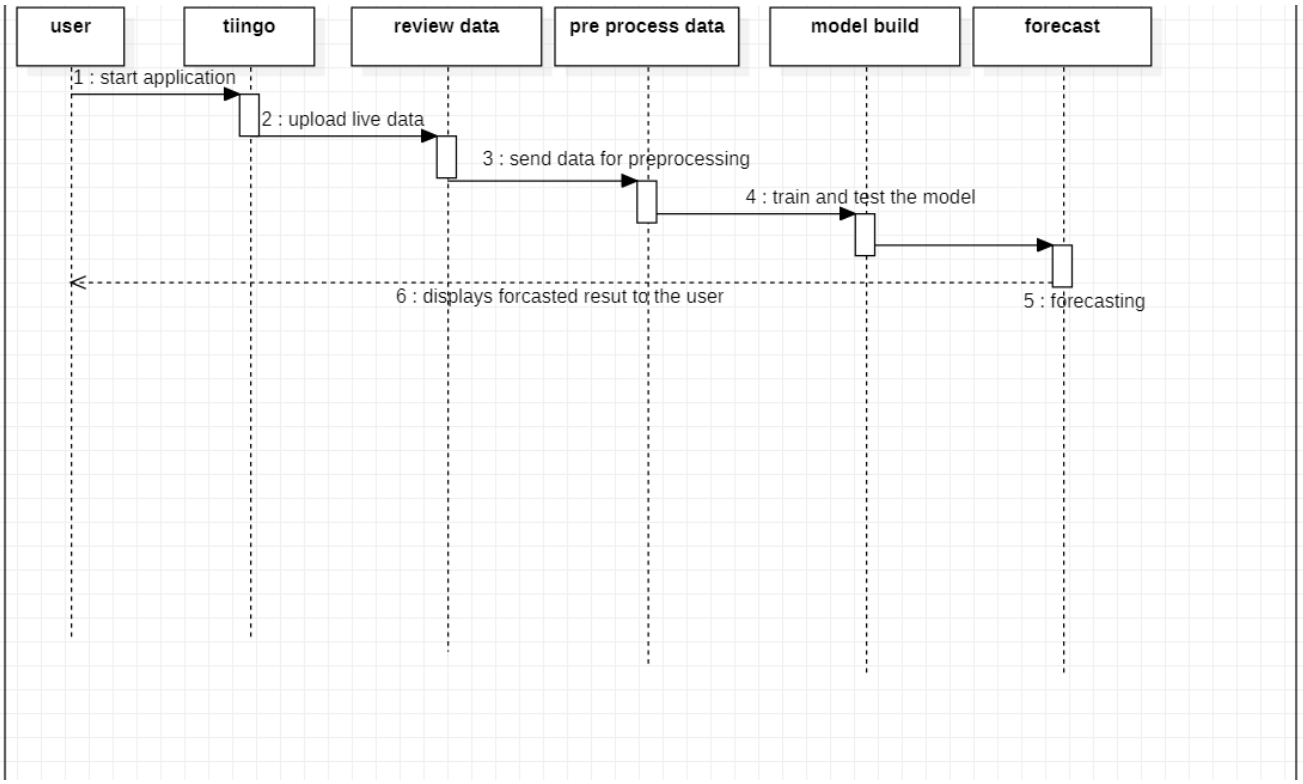

Fig 5.3: Sequence Diagram

5.4 Component diagram

Component diagrams are different in their nature and behavior when compared to other UML diagrams. These diagrams are used to model the physical aspects of the application or system, and it is also used to explain the physical artifacts of system or application; this physical aspects or artifacts are nothing but the elements such as executable files, libraries, documents, files so on, these physical aspects or artifacts reside in a node.

In other words, component diagrams help us visualize the relations and organizations among the components of the system or applications; not only relations but also these diagrams are also used to make executable systems or applications.

The primary purpose of the component diagram is:

- To Visualize the component of the system or application.
- To develop executables using forward and reverse engineering.
- To explain the relationships and organizations of the system or application.

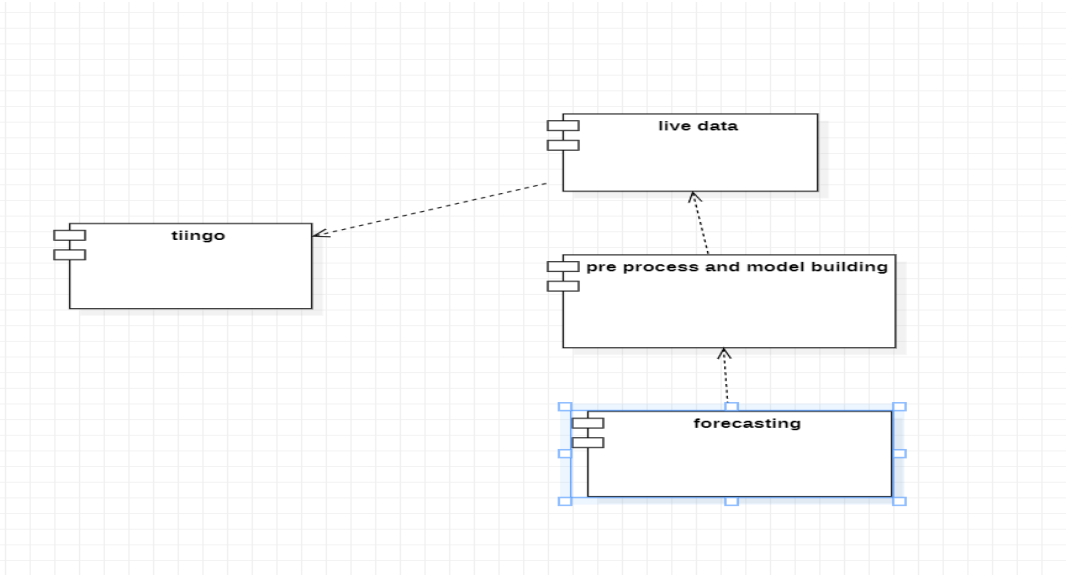

Fig 5.4: Component Diagram

5.5 Deployment Diagram

Deployment diagrams help to visualize the topology of the physical components of the system or application. In other words, the deployment diagrams help to visualize the static deployment view of the system or application. Since the physical components are deployed, deployment diagrams are related to the component diagram. A deployment diagram consists of nodes and relationships of the nodes.

The primary purpose of the deployment diagram is:

- To visualize the topology of the system or application.
- To explain the hardware components which are used to deploy the software components.
- To explain runtime processing nodes.

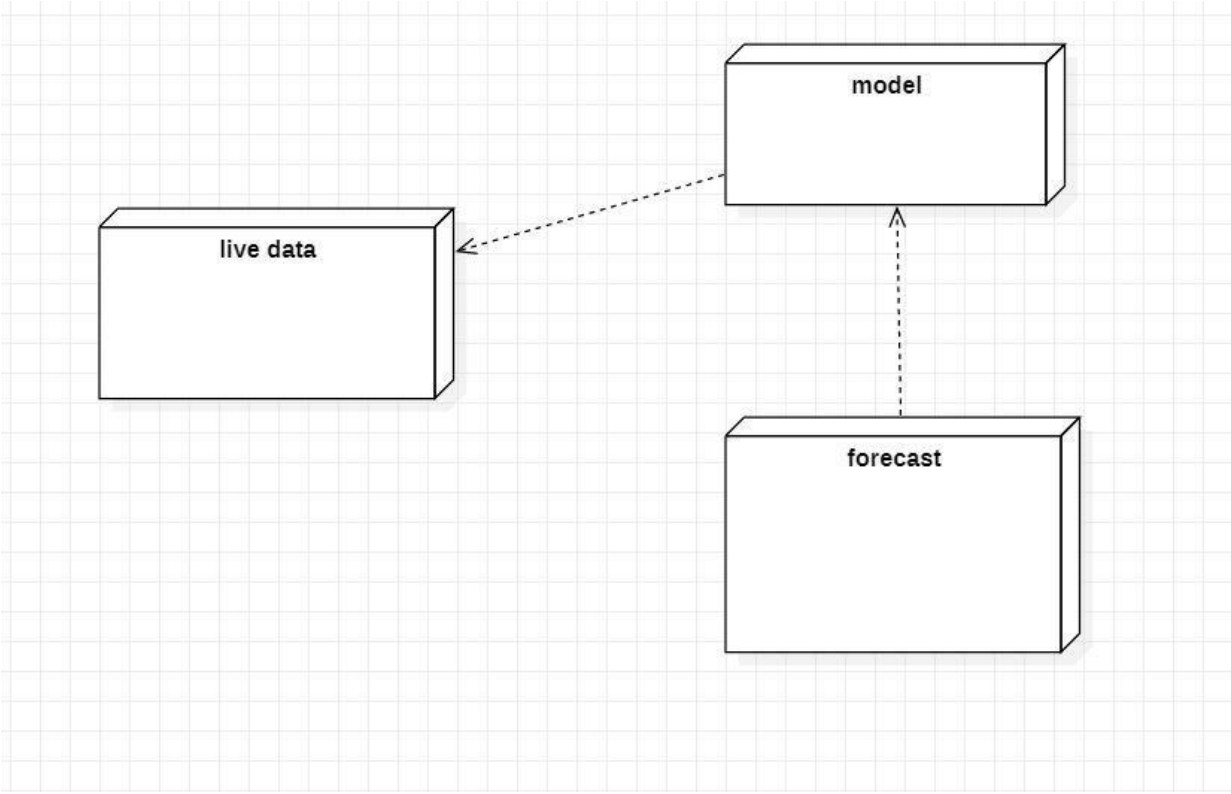

Fig 5.5: Deployment Diagram

5.6 State Chart Diagram

Statechart diagram the name itself depicts its purpose in the field of UML. Statechart diagram explains the various states and components in a system or application. In-state chart diagram, the states are specific to an object or component of the system or application.
Statechart diagram also explains the state machine, which can be explained as a machine that defines or explains various states of an object, and internal and external events handle these states.
The primary purpose of the Statechart Diagram is:

- Used to model the dynamic side of the system or application.
- Used to model the lifetime of a reactive system or application.
- To describe various states of a component or object in its lifetime.
- To depict a state machine that is used to model the states of a component or object.

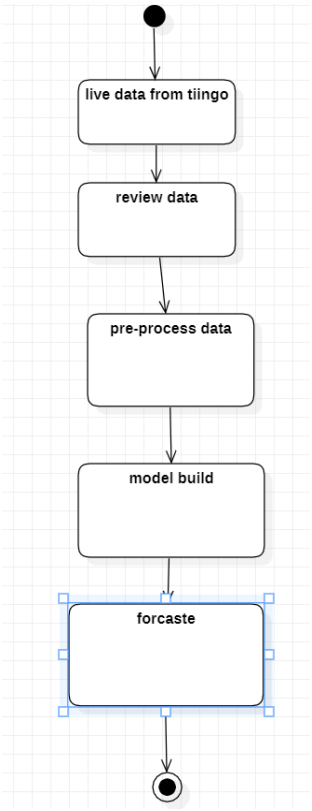

Fig 5.6: State Chart Diagram

5.7 Architecture

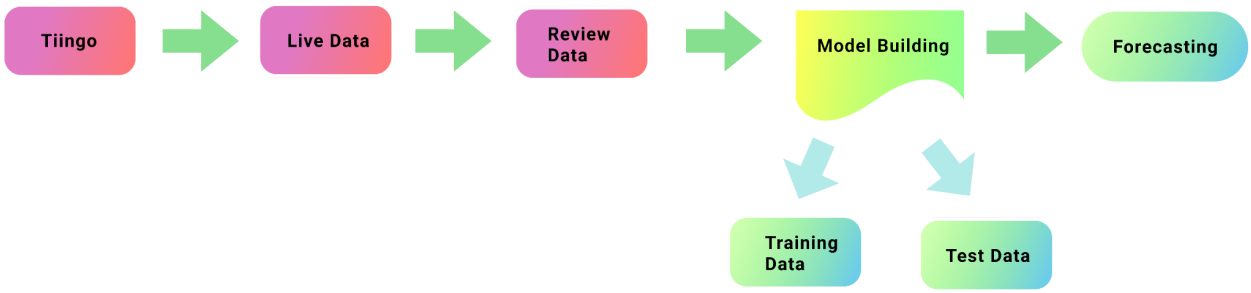

Fig 5.7: Architecture Diagram

6. IMPLEMENTATION

6.1 Modules

In this project, we use mainly four modules

1.Importing Data

2.Preprocessing

3.Model Build using bidirectional rnn

4.Prediction or Forecasting using bidirectional rnn

5.Model Build using ARIMA

6.Prediction or Forcasting using ARIMA

6.1.1 Importing Data

We utilize tiingo, which is an API that plays a crucial role in sharing live data. Tiingo has a unique authentication key that prevents other users or intruders from accessing the website and manipulating the data. In a single day, we can virtually import 50 times the live data from a single account and finally we convert the data into csv and send it for prepossessing it.

6.1.2 Preprocessing Data

In the preprocessing module, we use minmaxscaler, which is an inbuilt technique in the sklearn library. We directly import the minmaxscaler from sklearn and start preprocessing the data. It Transforms features by scaling each feature to a given range and for arima we will be removing the unwanted attributes keep the necessary attributes.

6.1.3 Model Build using bidirectional rnn

We'll manually partition the data in this module, label it as a test, and train it accordingly. Later, we'll restructure the data into [samples, time steps, and features] as needed for the LSTM. Now we'll import the TensorFlow technique and fit the train and test data into the model for 100 epochs of training.

6.1.4 Prediction or Forecasting using bidirectional rnn

We will convert the data from an array to metrics and fit them into the model in this module. Before that, we will examine the performance metrics of the data that has been preprocessed with root mean squared error. Next, we'll use the matplotlib package to plot the train predictions and then import stats from the scipy library to make predictions for the following n days.

6.1.5 Model Build Using Arima

We will index the data with reference to the attribute date after preprocessing the data by maintaining the essential attributes. We will fill in the null values later with the function bfill because it produces the most significant results. Now we will model fits the data to arima and choose the parameters p,d,q, where p,d,q refers to seasonality, trend, and noise in the data. Once the parameters have been determined, we will import the Arima model from the statsmodel.API module and fit the data for model building.

6.1.6 Predicting or Forecasting using arima

We use the plot diagnostics method to diagnose the forecast after the model has been built and stored in a variable called results. We utilize the predicted mean and record the anticipated mean values in a CSV file to examine the forecast better because the forecasted graph does not follow seasonality.

6.2 Sample Code

6.2.1 Importing Data

```
### Data Collection
import pandas_datareader as pdr
key=""
```

```
df = pdr.get_data_tiingo('AAPL', api_key="5b1dc91669e1c851407c34549b5a39e8e45215f7")
```

```
df.to_csv('AAPL.csv')
```

```
import pandas as pd
```

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

Fig 6.2.1.1: Importing Data

6.2.2 **Pre**processing Data in bidirectional rnn

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
df1=scaler.fit_transform(np.array(df1).reshape(-1,1))
```

```
print(df1)
```

```
[[0.01184721]
 [0.0115717 ]
 [0.00921728]
 ...
 [0.13570445]
 [0.13668128]
 [0.13800877]]
```

```
##splitting dataset into train and test split
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]
```

Fig 6.2.2.1: Preprocessing Data

```
# 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)
```

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

```
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')
```

Fig 6.2.2.2: Preprocessing Data

6.2.3 Model Build Using Bidirectional Rnn

```
# 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)
```

```
### Create the Stacked LSTM model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
```

```
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')
```

Fig 6.2.3.1:Reshaping the data

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

```
Epoch 38/100
12/12 [==============================] - 2s 203ms/step - loss: 2.9875e-04 - val_loss: 0.0131
Epoch 39/100
12/12 [==============================] - 3s 214ms/step - loss: 2.9370e-04 - val_loss: 0.0131
Epoch 40/100
12/12 [==============================] - 2s 205ms/step - loss: 3.0464e-04 - val_loss: 0.0125
Epoch 41/100
12/12 [==============================] - 2s 204ms/step - loss: 2.8146e-04 - val_loss: 0.0121
Epoch 42/100
12/12 [==============================] - 2s 205ms/step - loss: 3.1396e-04 - val_loss: 0.0118
Epoch 43/100
12/12 [==============================] - 2s 205ms/step - loss: 3.5645e-04 - val_loss: 0.0126
Epoch 44/100
12/12 [==============================] - 3s 213ms/step - loss: 3.0910e-04 - val_loss: 0.0110
Epoch 45/100
12/12 [==============================] - 3s 211ms/step - loss: 3.4708e-04 - val_loss: 0.0127
Epoch 46/100
12/12 [==============================] - 2s 204ms/step - loss: 3.0395e-04 - val_loss: 0.0118
Epoch 47/100
12/12 [==============================] - 2s 205ms/step - loss: 2.7932e-04 - val_loss: 0.0117
```

Fig 6.2.3.2:Model Training

6.2.4 Prediction or Forecasting Using Bidirectional Rnn

```
import tensorflow as tf
```

```
tf.__version__
```

```
'2.6.0'
```

```
### 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))
```

```
197.1566957502849
```

```
### Test Data RMSE
math.sqrt(mean_squared_error(ytest,test_predict))
```

```
176.46980680895754
```

Fig 6.2.4.1:Performance Metrics

```
### 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()
```

Fig 6.2.4.2:Train Predictions to Plotting

```
# demonstrate prediction for next 10 days
import matplotlib.pyplot as plt
from scipy import stats
import numpy as np
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)
```

Fig 6.2.4.3:Predictions For Following n days

6.2.5 Preprocessing for Arima

Data preprocessing

This step includes removing columns we do not need, check missing values, etc.

In [58]:
```
# filtering out unnecessary columns
cols = ['open', 'high', 'low', 'volume']
df.drop(cols, axis=1, inplace=True)
df = df.sort_values('date')
```

In [59]:
```
# checking for any missing values
df.isnull().sum()
```

Out[59]:
```
symbol         0
date           0
close          0
adjClose       0
adjHigh        0
adjLow         0
adjOpen        0
adjVolume      0
divCash        0
splitFactor    0
dtype: int64
```

In [60]:
```
df = df.groupby('date')['close'].sum().reset_index()
```

In [61]:
```
# here is the final df, with only date and closing price
df.head()
```

Fig 6.2.5.1:Preprocessing the data

Indexing with Time Series data

In [62]:
```
df = df.set_index('date')
df.index
```

Out[62]:
```
Index(['2017-03-21 00:00:00+00:00', '2017-03-22 00:00:00+00:00',
       '2017-03-23 00:00:00+00:00', '2017-03-24 00:00:00+00:00',
       '2017-03-27 00:00:00+00:00', '2017-03-28 00:00:00+00:00',
       '2017-03-29 00:00:00+00:00', '2017-03-30 00:00:00+00:00',
       '2017-03-31 00:00:00+00:00', '2017-04-03 00:00:00+00:00',
       ...
       '2022-03-07 00:00:00+00:00', '2022-03-08 00:00:00+00:00',
       '2022-03-09 00:00:00+00:00', '2022-03-10 00:00:00+00:00',
       '2022-03-11 00:00:00+00:00', '2022-03-14 00:00:00+00:00',
       '2022-03-15 00:00:00+00:00', '2022-03-16 00:00:00+00:00',
       '2022-03-17 00:00:00+00:00', '2022-03-18 00:00:00+00:00'],
      dtype='object', name='date', length=1259)
```

I have tried different ways to fill NA values, bfill seems to get the best results.

In [63]:
```
df.index = pd.to_datetime(df.index)
daily = df.close.resample('D')
daily = daily.fillna(method='bfill')
daily
```

Out[63]:
```
date
2017-03-21 00:00:00+00:00     843.20
2017-03-22 00:00:00+00:00     848.06
2017-03-23 00:00:00+00:00     847.38
2017-03-24 00:00:00+00:00     845.61
2017-03-25 00:00:00+00:00     846.82
                               ...
2022-03-14 00:00:00+00:00    2837.06
2022-03-15 00:00:00+00:00    2947.33
2022-03-16 00:00:00+00:00    3062.08
2022-03-17 00:00:00+00:00    3144.78
2022-03-18 00:00:00+00:00    3225.01
```

Fig 6.2.5.2:Preprocessing the data

6.2.6 Model Build (Arima)

In [65]:
```
# Define the p, d and q parameters to take any value between 0 and 2
p = d = q = range(0, 2)

# Generate all different combinations of p, q and q triplets
pdq = list(itertools.product(p, d, q))

# Generate all different combinations of seasonal p, q and q triplets
seasonal_pdq = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]

print('Examples of parameter combinations for Seasonal ARIMA...')
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[1]))
print('SARIMAX: {} x {}'.format(pdq[1], seasonal_pdq[2]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[3]))
print('SARIMAX: {} x {}'.format(pdq[2], seasonal_pdq[4]))
```

```
Examples of parameter combinations for Seasonal ARIMA...
SARIMAX: (0, 0, 1) x (0, 0, 1, 12)
SARIMAX: (0, 0, 1) x (0, 1, 0, 12)
SARIMAX: (0, 1, 0) x (0, 1, 1, 12)
SARIMAX: (0, 1, 0) x (1, 0, 0, 12)
```

In [66]:
```
warnings.filterwarnings("ignore") # specify to ignore warning messages

for param in pdq:
    for param_seasonal in seasonal_pdq:
        try:
            mod = sm.tsa.statespace.SARIMAX(daily,
                                            order=param,
                                            seasonal_order=param_seasonal,
                                            enforce_stationarity=False,
                                            enforce_invertibility=False)

            results = mod.fit()

            print('ARIMA{}x{}12 - AIC:{}'.format(param, param_seasonal, results.aic))
        except:
            continue
```

```
ARIMA(0, 0, 0)x(0, 0, 0, 12)12 - AIC:33463.04278339553
ARIMA(0, 0, 0)x(0, 0, 1, 12)12 - AIC:30913.600784366805
ARIMA(0, 0, 0)x(0, 1, 0, 12)12 - AIC:22525.208621905895
ARIMA(0, 0, 0)x(0, 1, 1, 12)12 - AIC:22378.059910226864
ARIMA(0, 0, 0)x(1, 0, 0, 12)12 - AIC:22531.153920648278
```

Fig 6.2.6.1:Model Build

In [67]:
```
mod = sm.tsa.statespace.SARIMAX(daily,
                                order=(1, 1, 1),
                                seasonal_order=(1, 1, 1, 12),
                                enforce_stationarity=False,
                                enforce_invertibility=False)

results = mod.fit()

print(results.summary().tables[1])
```

```
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9083      0.055     16.401      0.000       0.800       1.017
ma.L1         -0.9294      0.048    -19.375      0.000      -1.023      -0.835
ar.S.L12      -0.0256      0.023     -1.120      0.263      -0.070       0.019
ma.S.L12      -1.0000     26.382     -0.038      0.970     -52.708      50.708
sigma2      1480.7324   3.91e+04      0.038      0.970   -7.51e+04    7.81e+04
==============================================================================
```

In [68]:
```
results.plot_diagnostics(figsize=(16, 12))
plt.show()
```

Fig 6.2.6.2:Model Diagnosis

6.2.7 Prediction or Forecasting (Arima)

```
In [95]: pred_uc = results.get_forecast(steps=1459)
         pred_ci = pred_uc.conf_int()

         ax = daily.plot(label='observed', figsize=(18, 9))
         pred_uc.predicted_mean.plot(ax=ax, label='forecast')

         ax.fill_between(pred_ci.index,
                         pred_ci.iloc[:, 0],
                         pred_ci.iloc[:, 1], color='k', alpha=0.1)

         ax.set_xlabel('Date')
         ax.set_ylabel('Closing Price')

         plt.legend()
         plt.show()
```

Fig 6.2.7.1:Forecasting using arima

```
In [24]: pred_uc.predicted_mean

Out[24]: 2022-04-07 00:00:00+00:00    3173.324774
         2022-04-08 00:00:00+00:00    3177.082693
         2022-04-09 00:00:00+00:00    3177.021039
         2022-04-10 00:00:00+00:00    3178.964843
         2022-04-11 00:00:00+00:00    3180.598983
                                          ...
         2026-03-31 00:00:00+00:00    5017.421126
         2026-04-01 00:00:00+00:00    5018.683266
         2026-04-02 00:00:00+00:00    5018.165561
         2026-04-03 00:00:00+00:00    5018.284064
         2026-04-04 00:00:00+00:00    5022.744148
         Freq: D, Name: predicted_mean, Length: 1459, dtype: float64

In [34]: import datetime
         datetime.datetime.now()

Out[34]: datetime.datetime(2022, 4, 7, 21, 9, 46, 724750)

In [51]: pred_uc = pred_uc.predicted_mean
         #pred_uc.columns = ['predicted1','predicted2', 'predicted_n']

In [74]: df3=pred_uc.to_frame().reset_index()
         import matplotlib.pyplot as plt

In [75]: df3.columns=['Date','Predicted']
         plt.plot(df3['Date'],df3['Predicted'])
         #plt.xlabel('Date')
         plt.ylabel('Predicted')
         plt.show()
         df3.to_csv('D:\MAJOR PROJECT\PHASE-2\predictedAMZN1.csv', index=False)
```

Fig 6.2.7.2:Predicted Mean

7. SOFTWARE TESTING

It is practicable that errors occur in a software development project. There is a tendency to miss some bugs even when they are detected after every phase, like inspections. These remaining errors will inevitably show up in the code. As a result, the final code is likely to contain some requirements errors and design errors, in addition to any errors introduced during coding. Tests are to be performed so that any remaining errors from all phases can be identified. Tests, therefore, are crucial to quality assurance.

According to the ANSI/IEEE 1059 standard, finding out is defined due to the fact the gadget of analyzing a software program application item to find out the versions amongst modern-day and required conditions (i.e., defects/errors/bugs) and take a look at the software program application item's features.

7.1 Unit Testing:

Unit testing is said to be the first level of testing. Several modules are being tested during this process against specifications that have been produced during the design process. Tests are carried out primarily to improve the quality of code created during the coding phase to understand the logic built into the modules. Typically, this is done by the module's programmer. Only after a module has been unit-tested satisfactorily can it be considered for integration and use by others. Upon unit testing, our tool, on average, gave an accuracy of **95-98%**.

7.2 Integration Testing:

A typical integration test involves integrating/combining the module tested in independent tests and testing the combined behavior as a unit. This testing has as its primary purpose or goal is to check the interfaces of the units/modules. The testers generally conduct integral testing after unit testing. We combine the unit-tested modules soon, the multiple units are developed and tested, and then we begin integrating those tests. The actual goal of this testing process is to test the interfaces which are between the units/modules. Initially, each module is tested separately. As the modules are unit tested, each of them is integrated until the entire setup of blending is completed. This process allows testing the combinational behavior and validating the requirements as to whether they have been implemented correctly. It is vital to remember that Integration testing makes up a part of the development cycle rather than occurring at the end.

Therefore, there are no actual modules to test in most instances, so what makes the task challenging is to test something that doesn't exist!

The white box testing technique aims to improve the design, usability, and security of software by testing the internal structure, implementation, and coding it to verify input-output flow. White box testing is also referred to as an open box, transparent box, code-based, and black box testing because the tester sees the code.

Black field trying out specifically specializes in the capability of the gadget as whole check instances may be designed as quickly because the practical specs are completed. Tests could be accomplished from a give-up consumer's factor of view. because they give up consumer need to take delivery of the gadget. It is tough to pick out difficult inputs if the check instances aren't evolved primarily based totally on specs.

7.3 System Testing:

System Testing verifies that the software product is fully integrated and functioning as intended. A system test's objective is to evaluate the overall specifications of the whole system. This software is usually only a part of a more comprehensive computer-based system. Software/hardware systems are ultimately integrated with the software. A Systems Test consists of several different tests intended solely to examine a computer-based system as a whole.System test falls under the black box testing category of software testing.

7.4 Acceptance Testing:

Acceptance trying out evaluates software program utility to make certain it's far as much as specs and nice earlier than placing it into production. By trialing a chunk of software programs earlier than mass-produced, trojan horse fixes may be performed extra cost-effectively.

Many techniques are for use in reputation trying out, consisting of alpha/beta trying out, which rolls out the product in a preliminary alpha section to identify obvious errors, observed with the aid of using introducing it to beta testers to discover extra mild or minor mistakes.

8. RESULTS

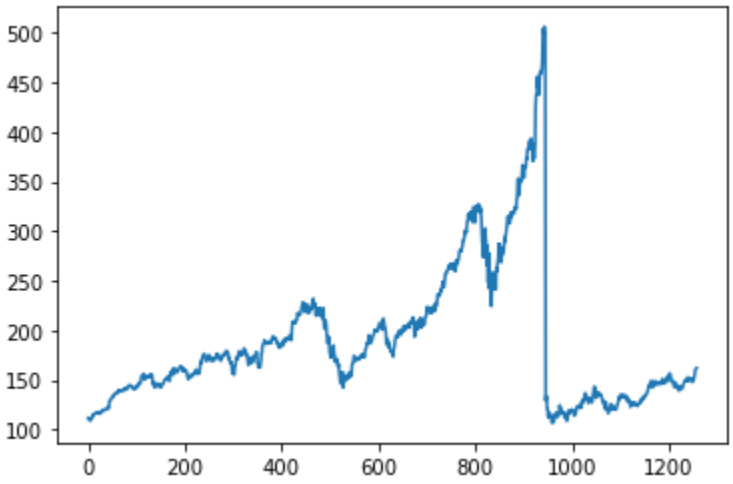

Fig 8.1:Graphical representation of live data

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100, 50)           10400
_________________________________________________________________
lstm_1 (LSTM)                (None, 100, 50)           20200
_________________________________________________________________
lstm_2 (LSTM)                (None, 50)                20200
_________________________________________________________________
dense (Dense)                (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

Fig 8.2: Model Summary After Training

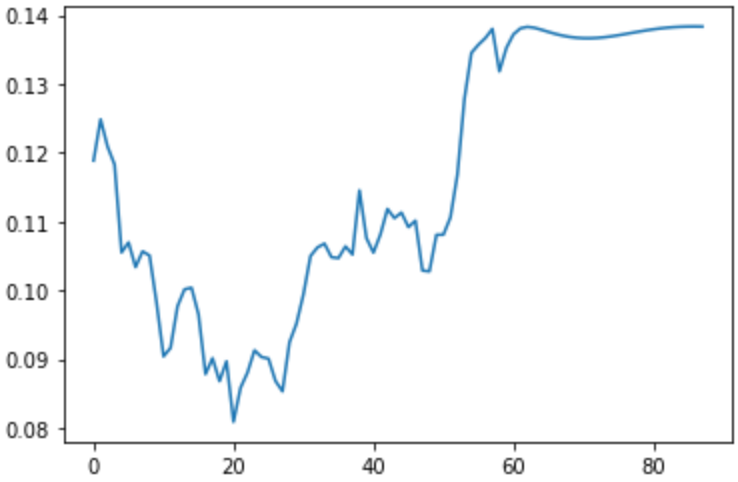

Fig 8.3:Train Predictions

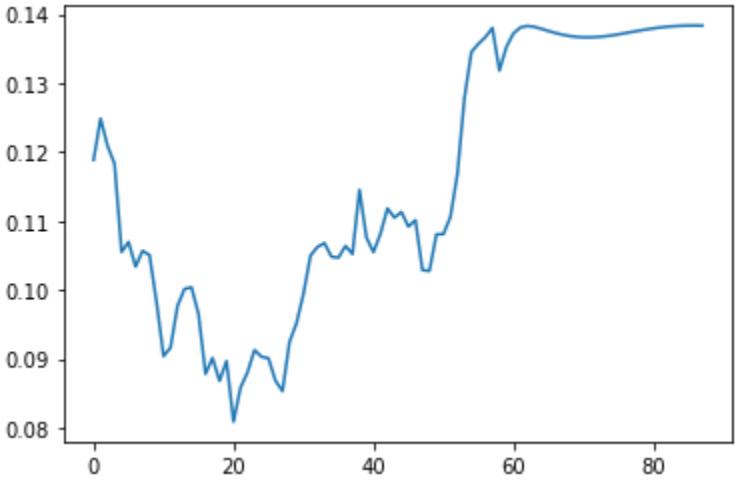

Fig 8.3.1: Final Result

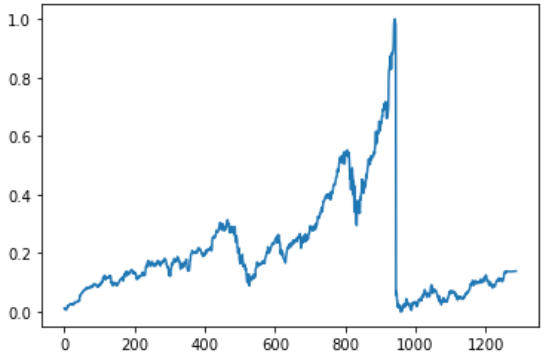

Fig 8.3.2: Final Result

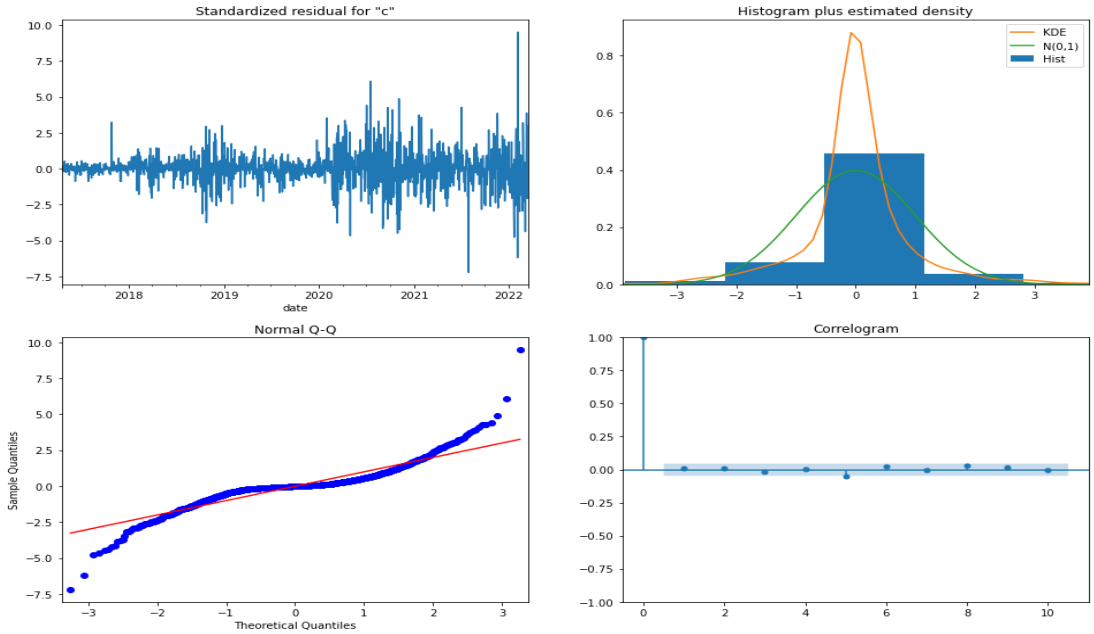

Fig 8.4: Model Diagnosis

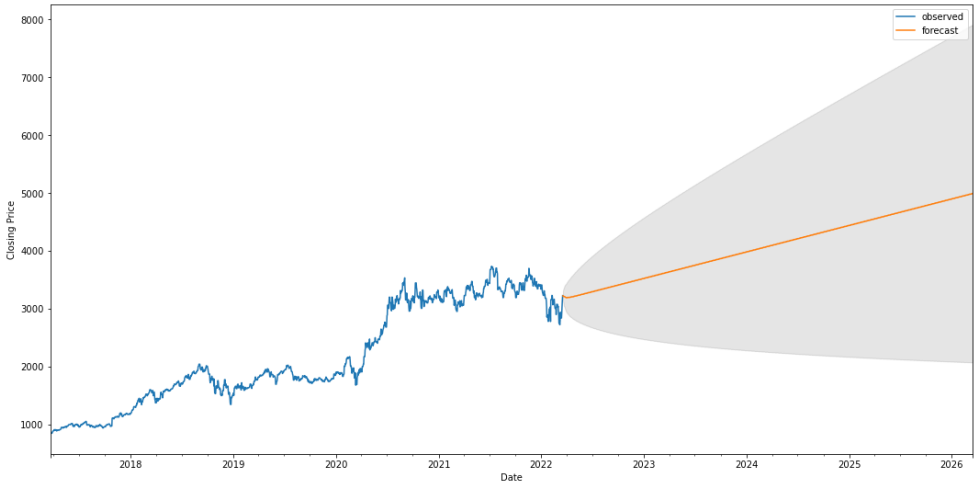

Fig 8.4.1: Model Forecast

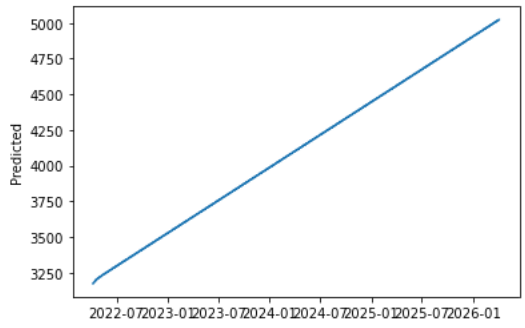

Fig 8.4.2: Predicted Mean Forecast

9. CONCLUSION AND FUTURE ENHANCEMENTS

9.1 Conclusion

Our main motivation for creating this project is to use a machine learning or deep learning algorithm to assist a person in making money by selling their shares with a prior understanding of the future scope of the shares and their share prices of a specific firm, which were predicted using live data rather than static data. As a result of our research, we have concluded that LSTM, also known as the long short-term memory approach, is the greatest fit for predicting stock prices and prices since LSTM can store historical data and ultimate outcomes over time.

9.2 Future enhancements

We were able to anticipate or forecast the share values for the following ten days with more efficiency since the live data was imported from Tiingo, an API that distributes the live data of a few big4 firms. Still, we can develop the algorithm in such a way that it works successfully for all other businesses' data from multiple APIs or from new sources in the future that offer livestock data to other firms.

10. BLIOGRAPHY

1. Atsalakis GS, Valavanis KP. Forecasting stock market short-term trends using a neuro-fuzzy-based methodology. Expert Syst Appl. 2009;36(7):10696–707.

2. Ayo CK. Stock price prediction using the ARIMA model. In: 2014 UKSim-AMSS 16th international conference on computer modeling and simulation. 2014. https://doi.org/10.1109/UKSim.2014.67.

3. Brownlee J. Deep learning for time series forecasting: predict the future with MLPs, CNNs, and LSTMs in Python. Machine Learning Mastery. 2018. https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

4. Eapen J, Bein D, Verma A. Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In: 2019 IEEE 9th annual computing and communication workshop and conference (CCWC). 2019. pp. 264–70. https://doi.org/10.1109/CCWC.2019.8666592.

5. Fischer T, Krauss C. Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res. 2018;270(2):654–69. https://doi.org/10.1016/j.ejor.2017.11.054.

6. Guyon I, Weston J, Barnhill S, Vapnik V. Gene selection for cancer classification using support vector machines. Mach Learn 2002;46:389–422.

7. Hafezi R, Shahrabi J, Hadavandi E. A bat-neural network multi-agent system (BNNMAS) for stock price prediction: case study of DAX stock price. Appl Soft Comput J. 2015;29:196–210. https://doi.org/10.1016/j.asoc.2014.12.028.

8. Halko N, Martinsson PG, Tropp JA. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. SIAM Rev. 2001;53(2):217–88.

9. Hassan MR, Nath B. Stock market forecasting using Hidden Markov Model: a new approach. In: Proceedings—5th international conference on intelligent systems design and applications 2005, ISDA'05. 2005. pp. 192–6. https://doi.org/10.1109/ISDA.2005.85.

10. Hochreiter S, Schmidhuber J. Long short-term memory. J Neural Comput. 1997;9(8):1735–80. https://doi.org/10.1162/neco.1997.9.8.1735.

11. Hsu CM. A hybrid procedure with feature selection for resolving stock/futures price forecasting problems. Neural Comput Appl. 2013;22(3–4):651–71. https://doi.org/10.1007/s00521-011-0721-4.

12. Huang CF, Chang BR, Cheng DW, Chang CH. Feature selection and parameter optimization of a fuzzy-based stock selection model using genetic algorithms. Int J Fuzzy Syst. 2012;14(1):65–75. https://doi.org/10.1016/J.POLYMER.2016.08.021.

13. Huang CL, Tsai CY. A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting. Expert Syst Appl. 2009;36(2 PART 1):1529–39. https://doi.org/10.1016/j.eswa.2007.11.062.

14. Idrees SM, Alam MA, Agarwal P. A prediction approach for stock market volatility based on time series data. IEEE Access. 2019;7:17287–98. https://doi.org/10.1109/ACCESS.2019.2895252.

15. Ince H, Trafalis TB. Short term forecasting with support vector machines and application to stock price prediction. Int J Gen Syst. 2008;37:677–87. https://doi.org/10.1080/03081070601068595.

16. Aparna Nayak, M. M. Manohara Pai and Radhika M. Pai(2016), Prediction Models for Indian Stock Market. https://www.sciencedirect.com/science/article/pii/S1877050916311619

17. Jingyi Shen and M. Omair Shafiq(2020) , Short-term stock market price trend prediction using a comprehensive deep learning system. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00333-6

18. Polamuri Subba Rao, K. Srinivas, and A. Krishna Mohan(2020), A Survey on Stock Market Prediction Using Machine Learning Techniques. https://www.researchgate.net/publication/341482418_A_Survey_on_Stock_Market_Prediction_Using_Machine_Learning_Techniques

19. Troy J. Strader, John J. Rozycki, Thomas H. Root, Yu-Hsiang (John) Huang(2020), Machine Learning Stock Market Prediction Studies: Review and Research Directions. https://scholarworks.lib.csusb.edu/cgi/viewcontent.cgi?article=1435&context=jitim

20. Jiake Li(2021), Research on Market Stock Index Prediction Based on Network Security and Deep Learning. https://www.hindawi.com/journals/scn/2021/5522375/#abstract

21. Fahim Faisal1, May 2012. Pp. 100 – 117, "Forecasting Bangladesh's Inflation Using Time Series ARIMA Models", World Review of Business Research Vol. 2. No. 3.

22. Al Wadia, Mohd Tahir Ismail S, "Selecting Wavelet Transforms Model in Forecasting Financial Time Series Data Based on ARIMA Model", Applied Mathematical Sciences, Vol. 5, 2011, no. 7, 315 – 326.

23. Kailash Chandra Pradhan, K. Sham Bhat, (2009), "An Empirical Analysis of Price Discovery, Causality and Forecasting in the Nifty Futures Markets", International Research Journal of Finance and Economics, ISSN 1450-2887 Issue 26.

24. K.K.Sureshkumar, Dr.N.M.Elango, November 2011, "An Efficient Approach to Forecast Indian Stock Market Price and their Performance Analysis", International Journal of Computer Applications (0975 – 8887), Volume 34– No.5.

25. K.K.Sureshkumar -Dr.N.M.Elango, "Exploiting data mining techniques for improving The efficiency of time series data Using spss-clementine", journal of arts, science & commerce, e-issn 2229-4686 - ISSN 2231-4172.

26. Kailash Chandra Pradhan, (2009), "An Empirical Analysis of Price Discovery, Causality and Forecasting in the Nifty Futures Markets", International Research Journal of Finance and Economics - ISSN 1450-2887 Issue 26.

27. Abhyankar, A., Copeland, L. S., & Wong, W. (1997). “Uncovering nonlinear structure in real-time stock-market indexes: The S&P 500, the DAX, the Nikkei 225, and the FTSE-100”. Journal of Business & Economic Statistics, 15, 1–14.

28. Achelis, S.B., Technical analysis from A to Z, IL: Probus Publishing, Chicago, 1995.

29. Aiken, M. and M. Bsat. “Forecasting Market Trends with Neural Networks.

30. Bartlett, MS 1964, On The Theoretical Specification of Sampling Properties of Autocorrelated Time Series,J. Roy. Stat. Soc., B8: 27–41.