Department of Computer Science and Engineering
Kalyani Government Engineering College
(Affiliated to Maulana Abul Kalam Azad University of Technology, West Bengal)
Kalyani - 741235, Nadia, WB

Project Report On

Stock Forecasting Approaches Using Hybrid LSTM Model and Its Comparison Study with Other Existing Models

(A dissertation submitted in partial fulfillment of the requirements of Bachelor of Technology in Computer Science and Engineering of the Maulana Abul Kalam Azad University of Technology)

Submitted by

Arnab Das (10201619039)
Anik Saha (10200119044)
Arkajyoti Bhattacharyay (10200119050)
Adrita Dutta (10200119026)

Under the guidance of

Prof. Deepak Kumar Jha

Faculty
Department of Computer Science and Engineering
Kalyani Government Engineering College

Session: 2019-2023

কল্যাণী - ৭৪১ ২৩৫
নদীয়া, পশ্চিমবঙ্গ

Kalyani 741 235
Nadia, West Bengal, India

পত্রাঙ্ক /Ref. No.:
তারিখ / Date :

কল্যাণী গভঃ ইঞ্জিনিয়ারিং কলেজ
Kalyani Government Engineering College
(Govt. of West Bengal)

Certificate of Approval

This is to certify that this report of B. Tech 8th Semester, 2023 project, entitled **"Stock Forecasting Approaches Using Hybrid LSTM Model and Its Comparison Study with Other Existing Models"** is a record of bona-fide work, carried out by **Adrita Dutta, Anik Saha, Arkajyoti Bhattacharyay, Arnab Das** under my supervision and guidance.

In my opinion, the report in its present form is in partial fulfillment of all the requirements, as specified by the ***Kalyani Government Engineering College*** and as per regulations of the ***Maulana Abul Kalam Azad University of Technology***. In fact, it has attained the standard, necessary for submission. To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report for the Project-III (PROJ-CS881) 8th Semester B. Tech program in Computer Science and Engineering in the year 2023.

Guide / Supervisor

____
Prof. Deepak Kumar Jha
Faculty
Department of CSE,
Kalyani Government Engineering College

____
Examiner(s)

____
Dr. Kousik Dasgupta
Head of the Department
Computer Science and Engineering
Kalyani Government Engineering College

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Prof. Deepak Kumar Jha,** Assistant Professor, Department of Computer Science and Engineering, Kalyani Government Engineering College, for his guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Dr. Kousik Dasgupta**, Head of the Department, Computer Science and Engineering, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, Kalyani Government Engineering College,** for their encouragement and cooperation to carry out this work.

We express our thanks to Project Coordinators **Dr. Anup Kumar Biswas & Dr. Supriyo Banerjee**, for their Continuous support and encouragement. We thank all **teaching faculty** of Department of CSE, whose suggestions during reviews helped us in accomplishment of our project.

At last but not the least, We would like to thank our parents, friends, and classmates for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

________________________________	Arnab Das	- 10201619039
________________________________	Anik Saha	- 10200119044
________________________________	Arkajyoti Bhattacharyay	- 10200119050
________________________________	Adrita Dutta	- 10200119026

DECLARATION

We, **Adrita Dutta, Anik Saha, Arkajyoti Bhattacharyay, Arnab Das**, students of B.Tech., CSE 4th year declare that we have submitted this report in partial fulfilment of the requirements of Bachelor of Technology in Computer Science and Engineering of Maulana Abul Kalam Azad University of Technology, West Bengal.

We solemnly declare that the project report on **"Stock Forecasting Approaches Using Hybrid LSTM Model and Its Comparison Study with Other Existing Models"** is based on our own work carried out during the course of our study under the supervision of **Professor Deepak Kumar Jha** sir.

We assert the statements made and conclusions drawn are an outcome of our research work. We further certify that,

I. The work contained in the report is original and has been done by me under the general supervision of my supervisor.

II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.

III. We have followed the guidelines provided by the university in writing the report.

IV. Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

PROJECT STUDENTS

______________________________	Arnab Das	- 10201619039
______________________________	Anik Saha	- 10200119044
______________________________	Arkajyoti Bhattacharyay	- 10200119050
______________________________	Adrita Dutta	- 10200119026

ABSTRACT

In this project, we attempt to implement various machine learning approaches to predict stock prices. Machine learning is effectively implemented in forecasting stock prices. The objective is to compare the various models and approaches to predict the future prices of a stock and then try to find the best working algorithm. We propose a stock price prediction system that integrates mathematical functions, machine learning, and other external factors for the purpose of achieving better stock prediction accuracy and issuing profitable trades. The project tests and compares the predictions of various models on stock values to show the accuracy and variations they have. At the end, we can conclude the usage and the matching that each type of model gives for the stock values, which is further used to create a prediction model. LSTMs and other neural models are very powerful in solving sequence prediction problems because they're able to store past information. The project includes such models in consideration also because the previous price of a stock is crucial in predicting its future price. While predicting the actual price of a stock is an uphill climb, we can build a model that will predict whether the price will go up or down.

CONTENTS

CHAPTER 1: INTRODUCTION

The financial market is a dynamic and composite system where people can buy and sell currencies, stocks, equities and derivatives over virtual platforms supported by brokers. The stock market allows investors to own shares of public companies through trading either by exchange or over the counter markets. Stock markets are affected by many factors causing the uncertainty and high volatility in the market. Automated trading systems (ATS) that are operated by the implementation of computer programs can perform better and with higher momentum in submitting orders than any human. However, to evaluate and control the performance of ATSs, the implementation of risk strategies and safety measures applied based on human judgements are required. Many factors are considered when developing an ATS, for instance, trading strategy to be adopted, complex mathematical functions that reflect the state of a specific stock, machine learning algorithms that enable the prediction of the future stock value, and specific news related to the stock being analyzed.

1.1 MOTIVATION FOR WORK

Businesses primarily run over customer's satisfaction and reviews about products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an un biased automated system to classify customer reviews regarding any problem. In today's environment where we're justifiably suffering from data overload, companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyze it manually without any sort of error or bias.

1.2 PROBLEM STATEMENT

Stock market prediction and analysis are some of the most difficult jobs to complete. There are numerous causes for this, including market volatility and a variety of other dependent and independent variables that influence the value of a certain stock in the market. These variables make it extremely difficult for any stock market expert to anticipate the rise and fall of the market with great precision. However, with the introduction of Machine Learning and its strong algorithms, the most recent market research and Stock Market Prediction advancements have begun to include such approaches in analyzing stock market data.

1.2.1 A SYSTEMATIC ANALYSIS & REVIEW OF DIFFERENT ALGORITHMS

The numerous methods are applied for achieving stock price prediction. Here, we will be considering the majority of those algorithms, which are broadly divided into following categories;

- **Traditional Machine Learning Methods:** Includes traditional methods such as,
    - Linear regression
    - Logistic regression
    - K-Nearest Neighbor (KNN)
    - Gaussian Naïve Bayes
    - Random forest
- **Time Series Analysis Methods:** These methods Include,
    - Simple Moving Average (SMA) with different weights
    - Autoregressive Integrated Moving Average (ARIMA)
    - Prophet
- **Deep Learning and Neural Networks:** Many of these techniques make use of RNNs and some of which are the special types of RNN,
    - Simple Recurrent Neural Network (RNN)
    - Convolutional Neural Network (CNN)
    - Gated Recurrent Unit (GRU)
    - Long Short-Term Memory (LSTM) (with various combination of Hyperparameters)
    - Bidirectional LSTM (Bi-LSTM) (with various combination of Hyperparameters)
    - Hybrid LSTM (For example: CNN-LSTM & CNN-Bi-LSTM)
    - Dense layer Forecast (For a control model)

These following algorithms will be tested, analyzed and reviewed on a same dataset of time-series stock data. By comparing the Root Mean Square Errors (RMSE) the best algorithm will be determined for the final prediction model.

1.2.2 PREDICTION WITH FINAL PREDICTION MODEL

Financial market prediction is a common variant of widely used time-series data prediction where the continuous data in a period of time is used to predict the result in the next time unit. Many algorithms have shown their effectiveness but the most common algorithms now are based on Recurrent Neural Networks' special type - Long-short Term Memory (LSTM). In this project, variants of LSTM model are mainly used to predict the future stock price.

1.2.3 PRESENTATION OF RESULTS WITH A WEB APPLICATION

A frontend made of HTML, CSS, JS is used to show the results of Python backend of the ML model.

CHAPTER 2: LITERATURE SURVEY

2.1 INTRODUCTION

Several studies have been the subject of using machine learning in the quantitative financial, predicting prices of managing and constricting entire portfolio of assets, as well as, investment process, and many other operations can be covered by machine learning algorithms. For quantitative finance and specially assets selections several models supply a large number of methods that can be used with machine learning to forecast future assets value. The combination of statistics and learning models have polished several machine learning algorithms, such as acritical neural networks, gradient boosted regression trees, support vector machines, autoregressive integrated moving average (ARIMA) and, LSTM. Neural Network prove to be very useful in financial market price prediction and forecasting. These algorithms can reveal complex patterns characterized by non-linearity as well as some relations that are difficult to detect with linear algorithms. These algorithms also prove more effectiveness and multi collinearity than the linear regressions ones.

A large number of studies is currently active on the subject of machine learning methods used in finance, some studies used tree-based models to predict portfolio returns, others used deep learning in the production of future values of financial assets. Also, using of ADaBoost algorithm or using unique decision-making model for day trading investments on the stock market is usual. The support vector machine (SVM) method, and the mean variance (MV) method is used for portfolio selection. Also, topics of return forecasting, portfolio construction, ethics, fraud detection, decision making, language processing and sentiment analysis has been covered in different algorithm studies. These models don't depend one long term memory, in this regard a class of machine learning algorithms based on Recurrent Neural Network prove to be very useful in financial market price prediction and forecasting. A paper has compared the accuracy of autoregressive integrated moving average (ARIMA) and LSTM, as illustrative techniques when forecasting time series data. These techniques were executed on a set of financial data and the results showed that LSTM was far more superior than ARIMA.

Our aim is to compare the majority of algorithms and to use ML algorithm based on hybrid LSTM to forecast the adjusted closing prices for a portfolio of assets. The main objective here is to obtain the most accurate trained algorithm, to predict future values of stocks.

2.2 EXISTING WORK

Stock Market Prediction is an area that has driven the focus of many individuals including not only companies, but also traders, market participants, data analysts, and even computer engineers working in the domain of Machine Learning (ML) and Artificial Intelligence (AI), etc. Investing funds in the market is subjected to various market risks, as the value of the shares of the company is highly dependent upon the profits and performance of the organization in the marketplace and can thus vary due to various factors such as government policies, microeconomic indicators, demand and supply etc. These variations in the market are studied to develop software and programs using various techniques such as ML, Deep Learning, Neural Networks, AI, etc.

Such systems and software can enable the investor to properly anticipate the situation of the company, on the basis of past and present data, the current condition in the market, etc. and give them a direction to make decisions so that they don't lose their valuable money and earn maximum profits.

2.2.1 Stock Market Prediction Using Traditional Machine Learning

Stock market prediction is an act of trying to determine the future value of a stock other financial instrument traded on a financial exchange. The technical and fundamental or the time series analysis is used by the most of the stockbrokers while making the stock predictions. In a Machine Learning (ML) approach model will be trained from the available stocks data and gain intelligence and then uses the acquired knowledge for an accurate prediction. In this context this study uses a machine learning technique called Support Vector Machine (SVM) to predict stock prices for the large and small capitalizations.

The authors of [1] studied the behavior of the stock market and determine the best fit model from the several traditional machine learning algorithms which included Random Forest (RF), Support Vector Machine (SVM), Naive Bayes, K-Nearest Neighbor (KNN), and SoftMax for stock market prediction. The authors conducted a comparative study of these approaches, several technical indicators were applied to the data that was gathered from different data sources including Yahoo and NSE-India. The accuracy of each model was compared and it was observed that RF gave the most satisfying results for large datasets whereas for small datasets Naive Bayesian revealed the highest accuracy. Another observation made was, as the count of technical indicators was reduced the accuracy of the models decreased.

In the paper [2] the authors study and apply different methods to predict stock prices but a high rate of accuracy is still not achieved even after analyzing major factors affecting the stock price. The authors have reviewed major techniques such as SVM, Regression, Random Forest, etc. and also analyzed hybrid models by combining two or more techniques. According to the authors, some models work better with historical data than with sentiment data. Fusion algorithms yielded results with higher predictions.

Predictions made using the Linear Regression Model have an enhanced accuracy rate after applying the Principal Component Analysis (PCA) on the data for picking out the most relevant components. SVM demonstrates high accuracy on non-linear classification data, Linear regression is preferred for linear data because of its high confidence value, a high accuracy rate was observed on a binary classification model using Random Forest Approach and the Multilayer Perceptron (MLP) yielded the least amount of error while making predictions. Many of the aforementioned techniques are not just limited to stock price prediction but can also be used broadly in the financial markets as the authors in the paper [3] conclude by studying the application of machine learning models to analyze financial trading and to design optimal strategies for the same. After performing a quantitative analysis of different techniques, the authors recommend delving further into behavioral finance to evaluate market or investor psychology to understand market fluctuations. The authors propose to make use of text mining and machine learning methods to monitor public interaction on digital financial trading platforms.

2.2.2. Graph-Based Approach

The paper [4] visualizes the stock market as a graphical network in a rather unique way and the authors have included both correlation and causation using historical price data as well as applying sentiment analysis which is highly useful in taking into account different factors that determine the stock price. The Graph Convolutional Network model proposed in this paper is vulnerable to the detonating inclination issue as nodes with more significant levels will have bigger worth in their convolved feature portrayal, while nodes with a more modest degree will have more modest worth in feature representation.

2.2.3. Forecasting the Stock Market Index Using Artificial Intelligence Techniques

The weak form of Efficient Market hypothesis (EMH) states that it is impossible to forecast the future price of an asset based on the information contained in the historical prices of an asset. The market behaves as a random walk and makes forecasting impossible due to the intrinsic complexity of the financial system. Three artificial intelligence techniques, namely, neural networks (NN), support vector machines and neuro-fuzzy systems are implemented in forecasting the future price of a stock market index based on its historical price information. Artificial intelligence techniques have the ability to take into consideration financial system complexities and they are used as financial time series forecasting tools. Two techniques are used to benchmark the AI techniques, namely, Autoregressive Moving Average (ARMA) which is linear modelling technique and random walk (RW) technique. The results showed that the three techniques have the ability to predict the future price of the Index with an acceptable accuracy, outperforming the linear model. However, the random walk method out performs all the other techniques. However, because of the transaction costs of trading in the market, it is not possible to show that the three techniques can disprove the weak form of market efficiency. The results also showed that the ranking of performances support vector machines, neuro-fuzzy systems, multilayer perceptron neural networks is dependent on the accuracy measure used.

2.2.4. Deep Learning and Neural Networks

The paper [5] proposes three different Artificial Neural Network models which include the use of multiple-input features, binary features and technical features to find the best approach to achieve the aim. The accuracy of the models was computed and revealed that the model with binary features showed the best accuracy and concluded that binary features are lightweight and are most suitable for stock prediction. However, the study has some limitations in that converting the features to binary eliminates some of the relevant information for prediction. Delving into specific techniques methods such as the Multi-Layer Perceptron Model (MLP), Sequential Minimal Optimizations and the Partial Least Square Classifier (PLS) have been studied and applied on the Stock Exchange. Paper [6] focuses on the effect of the indices in the stock price prediction. The model identifies the variables and relationship between the indices and overcomes the limitations of the traditional linear model and uses LSTM to understand the dynamics of the S&P 500 Index. The paper also analyses the sensitivity of internal memory of LSTM modelling. However, the study has some limitations, the difference between the predictive value and actual value becomes large after a certain point and thus cannot be used to develop a system to give a profitable trading strategy. [7] proposes a system that would recommend stock purchases to the buyers. The approach opted by the authors combines the prediction from historical and real-time data using LSTM for predicting. In the RNN model, latest trading data and technical indicators are given as input in the first layer, followed by the LSTM, a compact layer and finally the output layer gives the predicted value. These predicted values are further integrated with the summarized data which is collected from the news analytics to generate a report showing the percentage in change.

2.2.5. Stock Price Correlation Coefficient Prediction with Hybrid LSTM Model

Predicting the price correlation of two assets for future time periods is important in portfolio optimization. LSTM recurrent neural networks (RNN) is applied in predicting the stock price correlation coefficient of two individual stocks. RNN's are competent in understanding temporal dependencies. The use of LSTM cells further enhances its long-term predictive properties. To encompass both linearity and nonlinearity in the model, we adopt the ARIMA model as well. The ARIMA model filters linear tendencies in the data and passes on the residual value to the LSTM model. The ARIMA-LSTM hybrid model is tested against other traditional predictive financial models such as the full historical model, constant correlation model, single-index model and the multi-group model. In our empirical study, the predictive ability of the ARIMA-LSTM model turned out superior to all other financial models by a significant scale. Results implies that it is worth considering the ARIMA-LSTM model to forecast correlation coefficient for portfolio optimization.

❖ ***References in Literature Survey:***

[1] Kumar, I., Dogra, K., Utreja, C. and Yadav, P., 2018, April. A comparative study of supervised machine learning algorithms for stock market trend prediction. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 1003-1007). IEEE.

[2] Singh, Sukhman, Tarun Kumar Madan, J. Kumar and A. Singh. "Stock Market Forecasting using Machine Learning: Today and Tomorrow." 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) 1 (2019): 738-745.

[3] Vats, P. and Samdani, K., 2019, March. Study on Machine Learning Techniques In Financial Markets. In 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN) (pp. 1-5). IEEE.

[4] Patil, P., Wu, C.S.M., Potika, K. and Orang, M., 2020, January. Stock market prediction using an ensemble of graph theory, machine learning and deep learning models. In Proceedings of the 3rd International Conference on Software Engineering and Information Management (pp. 85-92).

[5] Song, Y. and Lee, J., 2019, December. Design of stock price prediction model with various configurations of input features. In Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing (pp. 1-5).

[6] Xingzhou, L., Hong, R. and Yujun, Z., 2019, July. Predictive Modeling of Stock Indexes Using Machine Learning and Information Theory. In Proceedings of the 2019 10th International Conference on E-business, Management and Economics (pp. 175-179).

[7] S. Sarode, H. G. Tolani, P. Kak and C. S. Life, "Stock Price Prediction Using Machine Learning Techniques," 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2019, pp. 177-181. doi: 10.1109/ISS1.2019.8907958

CHAPTER 3: PROPOSED WORK

Our aim is to compare Machine learning algorithm based mainly on Hybrid LSTM to the existing other widely used machine learning algorithms. The best model will forecast the adjusted closing prices of the stocks which will be presented by a web application.

3.1 Overview of Used Algorithms:

The way these algorithms work is, they're provided with an initial batch of data, and with time, as algorithms develop their accuracy, additional data is introduced into the mix. This process of regularly exposing the algorithm to new data and experience improves the overall efficiency of the machine. ML algorithms are vital for a variety of tasks related to predictive modeling, and analysis of data.

3.1.1. Traditional Machine Learning Algorithms

Today, traditional machine learning algorithms are significantly overshadowed by deep learning. But they are still well suited for many applications independently or as a support in complex scenarios,

Linear regression: Linear Regression is the supervised Machine Learning model in which the model finds the best fit linear line between the independent and dependent variable i.e. it finds the linear relationship between the dependent and independent variable.

Logistic regression: Logistic regression is a statistical method that is used for building machine learning models where the dependent variable is binary. Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables.

K-Nearest Neighbor (KNN): It is a supervised machine learning algorithm which can be used to solve both classification and regression problem statements. The number of nearest neighbors to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'.

Gaussian Naïve Bayes: Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naïve Bayes is the extension of naïve Bayes.

Random forest: The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

3.1.2. Time Series Analysis Algorithms

Simple Moving Average (SMA): SMA is the easiest moving average to construct. It is simply the average price over the specified period. The average is called "moving" because it is plotted on the chart bar by bar, forming a line that moves along the chart as the average value changes.

Autoregressive Integrated Moving Average (ARIMA): ARIMA is a statistical analysis model that uses time series data to either better understand the data set or to predict future trends. This is the combination of Auto Regression and Moving average.

Prophet: Prophet procedure is an additive regression model with four main components: A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.

3.1.3. Deep Learning and Neural Networks Algorithms

Many of these techniques make use of RNNs and some of which are the special types of RNN,

Simple Recurrent Neural Network (RNN): RNNs are a variant of the conventional feedforward artificial neural networks that can deal with sequential data and can be trained to hold knowledge about the past.

Convolutional Neural Network (CNN): CNN or ConvNet is a network architecture for deep learning that learns directly from data. CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories. They can also be quite effective for classifying audio, time-series, and signal data.

Gated Recurrent Unit (GRU): Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate.

Long Short-Term Memory (LSTM) (with various combination of Hyperparameters): LSTM is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. The unit is called a long short-term memory block because the program is using a structure founded on short-term memory processes to create longer-term memory.

<u>Bidirectional LSTM (Bi-LSTM) (with various combination of Hyperparameters):</u> A Bidirectional LSTM, or BiLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction (past to future) and the other in a backwards direction (future to past), and it's capable of utilizing information from both sides.

<u>Hybrid LSTMs:</u> Hybrid LSTM neural network structure consists of one input combination layer, zero or more intermediate combination layers and one output combination layer. **(For example: CNN-LSTM & CNN-Bi-LSTM)**

3.2. Working of Long short-term memory network (LSTM)

RNN can't store long time memory, so the use of LSTM based on "memory line" proved to be very useful in forecasting cases with long time data. In a LSTM the memorization of earlier stages can be performed trough gates with along memory line incorporated. LSTM model has three multiplicative units, i.e., input gate, output gate and forget gate. The input gate is used to memorize some information of the present, the output gate is used for output, and the forget gate is used to choose to forget some information of the past. While information enters the LSTM's network, it can be selected by rules. Only the information conforms to the algorithm will be left, and the information that does not conform will be forgotten through the forgetting gate.

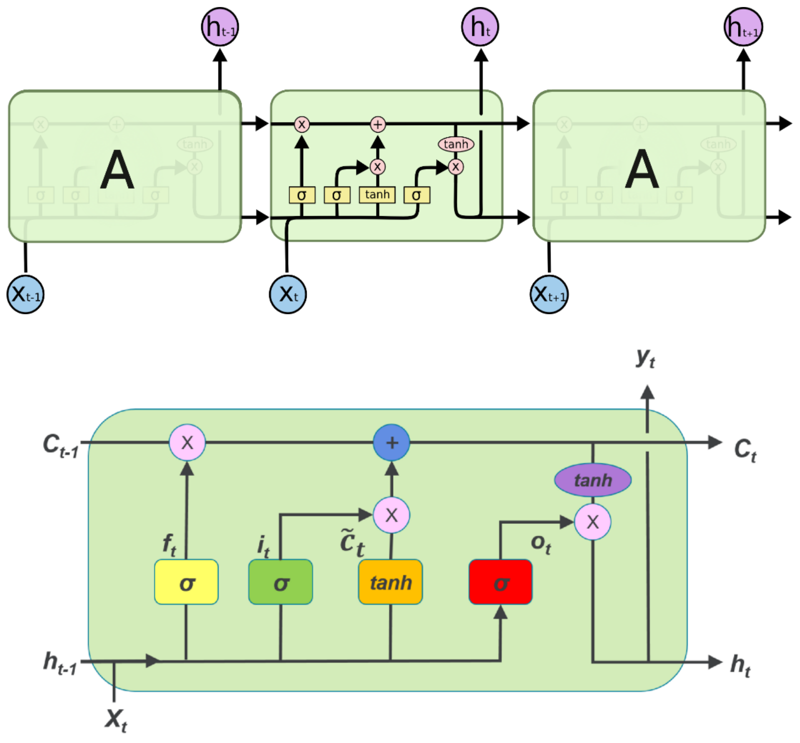


Fig.: LSTM Architecture

3.3. Working of Bidirectional Long short-term memory network (Bidirectional LSTM)

BiLSTM adds one more LSTM layer, which reverses the direction of information flow. Briefly, it means that the input sequence flows backward in the additional LSTM layer. Then we combine the outputs from both LSTM layers in several ways, such as average, sum, multiplication, or concatenation.

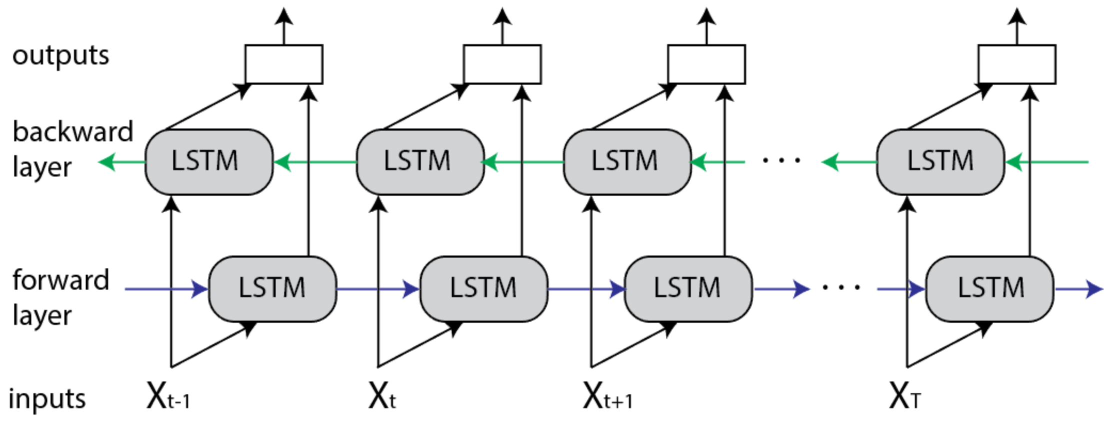


Fig.: Bidirectional LSTM Architecture

3.4. Working of Hybrid Long short-term memory networks (Hybrid LSTMs)

Hybrid LSTM neural network structure consists of one input combination layer, zero or more intermediate combination layers and one output combination layer.

- The input combination layer may include any kind of Time series or Neural network layer, may it be a CNN, LSTM or BILSTM layer (With an Activation layer and a Dropout layer).
- The intermediate combination layers include mainly Neural network layers (With or without a Dense layer, an Activation layer and a Dropout layer).
- The output combination layer includes a Dense layer and an Activation layer.

Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer. **Dropout layer** means that some neural units are temporarily discarded from the neural network according to a certain probability during the training process, preventing over-fitting.

The results show that additional training of data in Hybrid LSTM offers better predictions than BiLSTM which is better than regular LSTM-based models. More specifically, it was observed that CNN-Bi-LSTM models provide better results than hybrid ARIMA and LSTM models.

3.5. Progress of proposed work

3.5.1 Structure Chart of the Basic Models

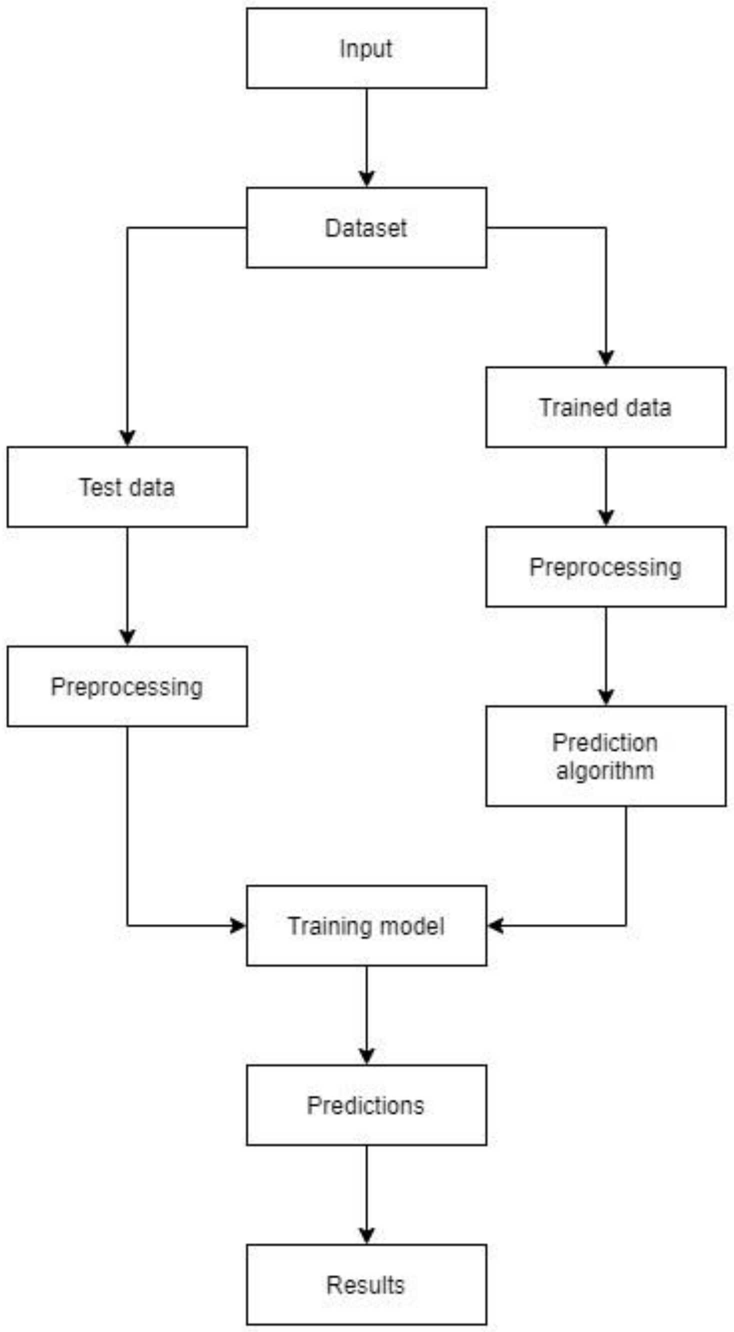


Fig.: Training and prediction

3.5.2 Implementation Steps

Step1: Raw Stock Price Dataset: Day-wise past stock prices of selected companies are collected from the official website.

Step2: Pre-processing: This step incorporates the following:

a) Data discretization: Part of data reduction but with particular importance, especially for numerical data

b) Data transformation: Normalization.

c) Data cleaning: Fill in missing values.

d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate.

Step3: Feature Selection: In this step, data attributes are chosen that are going to be fed to the neural network. In this study Date & Close Price are chosen as selected features.

Step 4: Train the model: The model is trained by feeding the training dataset. The model is initiated using random weights and biases. Different procedures are followed for different proposed traditional machine learning, time series and deep learning/ neural network algorithms to get the desired results.

For example, the final proposed LSTM model consists of a sequential input layer followed by 3 LSTM layers and then a dense layer with activation. The output layer again consists of a dense layer with a linear activation function.

Step5: Output Generation: The generated output is compared with the target values and error difference is calculated. The Backpropagation algorithm is used to minimize the error difference by adjusting the biases and weights of the neural network.

Step 6: Test Dataset Update: Step 2 is repeated for the test data set.

Step 7: Error and companies' net growth calculation: By calculating deviation we check the percentage of error of our prediction with respect to actual price.

Step 8: Visualization: Using pyplot, Keras and their function APIs the prediction is visualized.

Step 9: Investigate different time interval: We repeated this process to predict the price at different time intervals. In different time spans, we calculate the percentage of error in the future prediction.

CHAPTER 4: EXPERIMENTAL ANALYSIS AND RESULTS

4.1. System Configuration:

This project can run on commodity hardware. We ran entire project on an Intel i5 processor with 4 cores which run at the base speed of 2.3 GHz along with 8 GB RAM, 8 GB Nvidia 1050Ti Graphic Processor. First part of the is training phase which takes 15-20 mins of time and the second part is testing part which only takes few seconds to make predictions and calculate accuracy.

4.1.1. Hardware Requirements:

- RAM: 8 GB
- CPU: 2 GHz or faster
- Architecture: 32-bit or 64-bit

4.1.2. Software Requirements:

- Google Collaboratory, Anaconda & Jupyter Notebook with Python 3.5 are used for data pre-processing, model training and prediction.
- VS Code is used for creating Web Application.
- Operating System: windows 7 and above or Linux based OS or MAC OS.

4.2. Dataset Description & Analysis:

The experimental data in this project are the actual historical data downloaded from the internet. The historical stock data table contains the information of **transaction date, opening price**, **highest price**, **lowest price**, **closing price**, **adjusted closing price**, **volume** and so on. It is needed to find an optimization algorithm that requires less resources and has faster convergence speed.

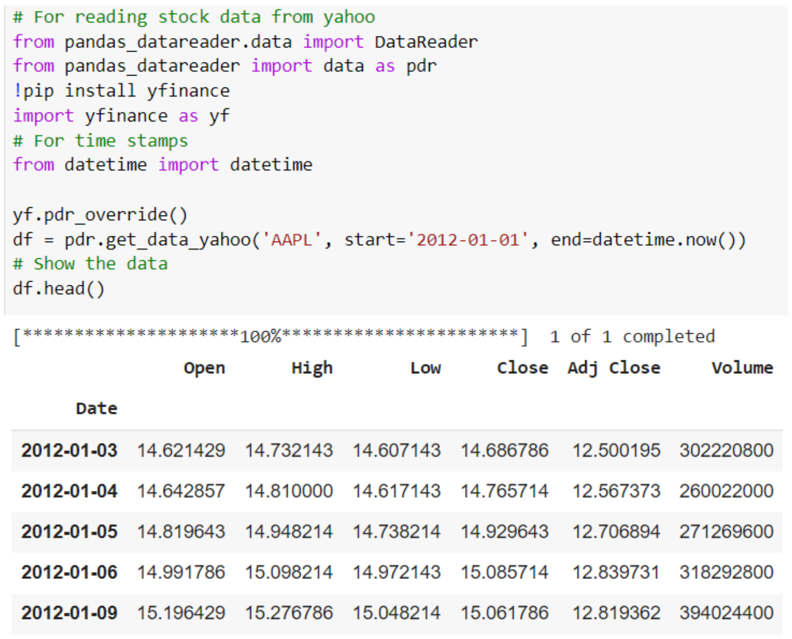
Dataset source - Yahoo Finance website: https://finance.yahoo.com

Initial models are tested by using **Apple dataset** from: https://finance.yahoo.com/quote/AAPL

For Web Application, Datasets of **Google, Apple, Tesla, Nvidia etc.** are used from the same source.

Dataset Overview & Required Data Visualization via Matplotlib Plotting:

Pandas head() method is used to return top n (5 by default) rows of a data frame or series.



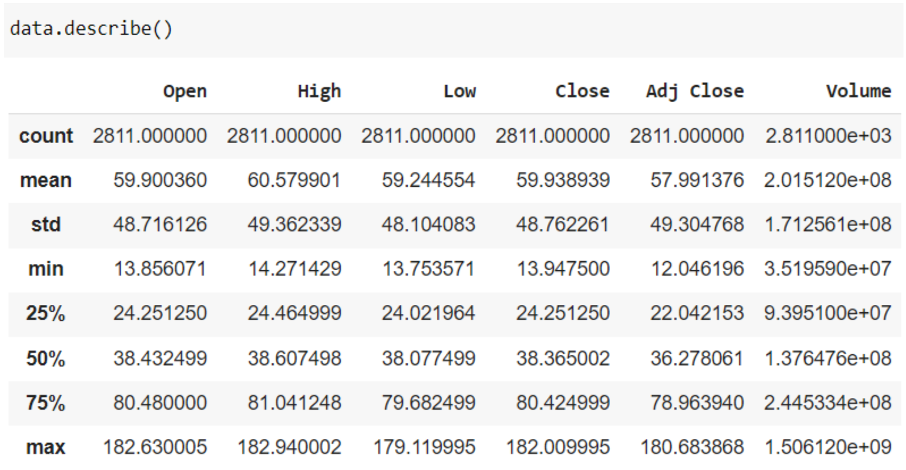
```
# For reading stock data from yahoo
from pandas_datareader.data import DataReader
from pandas_datareader import data as pdr
!pip install yfinance
import yfinance as yf
# For time stamps
from datetime import datetime

yf.pdr_override()
df = pdr.get_data_yahoo('AAPL', start='2012-01-01', end=datetime.now())
# Show the data
df.head()
```

```
[*********************100%***********************]  1 of 1 completed
                 Open       High        Low      Close  Adj Close     Volume
      Date
2012-01-03  14.621429  14.732143  14.607143  14.686786  12.500195  302220800
2012-01-04  14.642857  14.810000  14.617143  14.765714  12.567373  260022000
2012-01-05  14.819643  14.948214  14.738214  14.929643  12.706894  271269600
2012-01-06  14.991786  15.098214  14.972143  15.085714  12.839731  318292800
2012-01-09  15.196429  15.276786  15.048214  15.061786  12.819362  394024400
```

In stock trading, the high and low refer to the maximum and minimum prices in a given time period. Open and close are the prices at which a stock began and ended trading in the same period. Volume is the total amount of trading activity.

While closing price merely refers to the cost of shares at the end of the day, the adjusted closing price considers other factors like dividends, stock splits, and new stock offerings. Since the adjusted closing price begins where the closing price ends, it can be called a more accurate measure of stocks' value.

Pandas describe() method returns description of the data in the DataFrame.

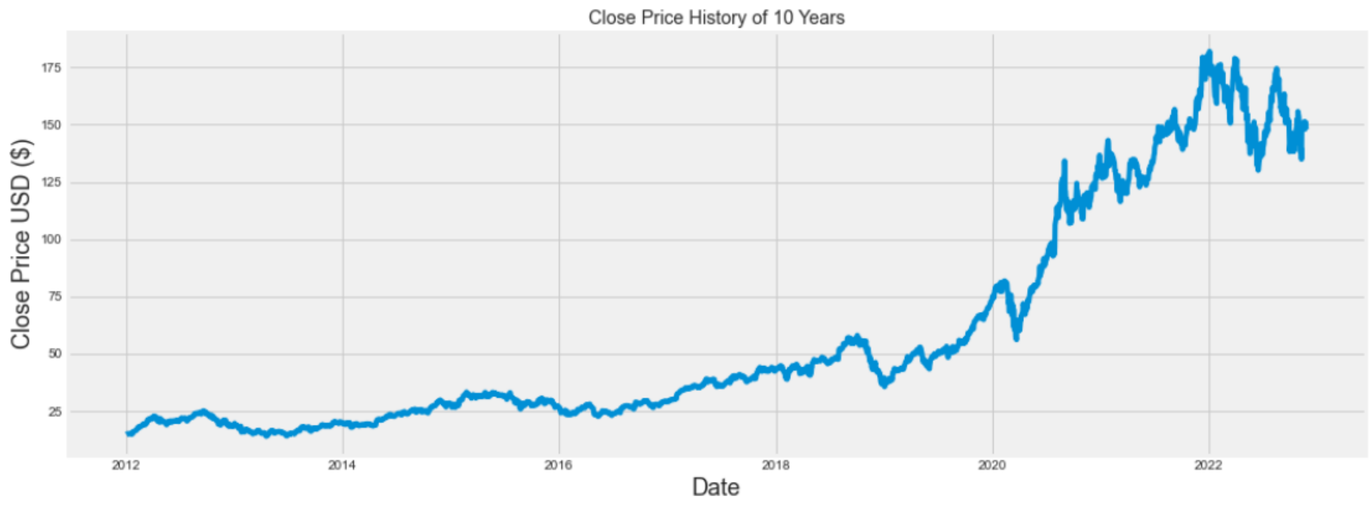


```
data.describe()
```

	Open	High	Low	Close	Adj Close	Volume
count	2811.000000	2811.000000	2811.000000	2811.000000	2811.000000	2.811000e+03
mean	59.900360	60.579901	59.244554	59.938939	57.991376	2.015120e+08
std	48.716126	49.362339	48.104083	48.762261	49.304768	1.712561e+08
min	13.856071	14.271429	13.753571	13.947500	12.046196	3.519590e+07
25%	24.251250	24.464999	24.021964	24.251250	22.042153	9.395100e+07
50%	38.432499	38.607498	38.077499	38.365002	36.278061	1.376476e+08
75%	80.480000	81.041248	79.682499	80.424999	78.963940	2.445334e+08
max	182.630005	182.940002	179.119995	182.009995	180.683868	1.506120e+09

As, the DataFrame contains numerical data, the description contains these information for each column:

count - The number of not-empty values.
mean - The average (mean) value.
std - The standard deviation.
min - the minimum value.
25% - The 25% percentile.
50% - The 50% percentile.
75% - The 75% percentile.
max - the maximum value.



4.3. Selection of the closed or the adjusted close price as primary requirement

The adjusted closing price is important because it gives investors a more current and accurate idea of the stock's price. It informs investors of any calculations after a corporate action. On the other hand, open, high etc. only state the respective point of price before it gets a fixed value i.e., the adjusted closing price. That’s why closed is mainly chosen over open, high etc. for final predictions.

4.4. Feature selection argument

Feature Selection is a method to select a feature subset from all the input features to make the constructed model better. In the practical application of machine learning, the quantity of features is normally very large, in which there may exist irrelevant features, or the features may have dependence on each other. Feature Selection can remove irrelevant or redundant features, and thus decrease the number of features to improve the accuracy of the model. The purpose of reducing the running time can also be achieved. On the other hand, selecting the really relevant features can simplify the model, and make the data generation process easy-to understand for the researchers.

4.4.1. Data selection

According to the input features commonly used in the LSTM neural network model, this project also selects five features as input: open price, highest price, lowest price, close price and adjusted close price.
In this experiment, the results of the input features will be compared by selecting only one feature at a time and training them without changing the model factors i.e., the train and test sizes, neural network parameters, number of neurons, epoch and batch sizes etc. then the prediction RMSE results will be compared and analysed respectively.
In the experiment described, a simple LSTM is chosen as the neural network to build the model. For the division of training set and test set, this test adopts 8:2 for division.

4.4.2. Model design

```
model=Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=1,epochs=3)
```

This paper chooses the sequential model of two-layer LSTM network to build the neural network model. A two-layer LSTM is added, and 50 neurons are set in each of its hidden layer. The loss function of the model chooses Mae, choosing Adam as the Optimizer, and activating the function chooses the default function. The number of iterations is 3, and we chose each batch size as 1.

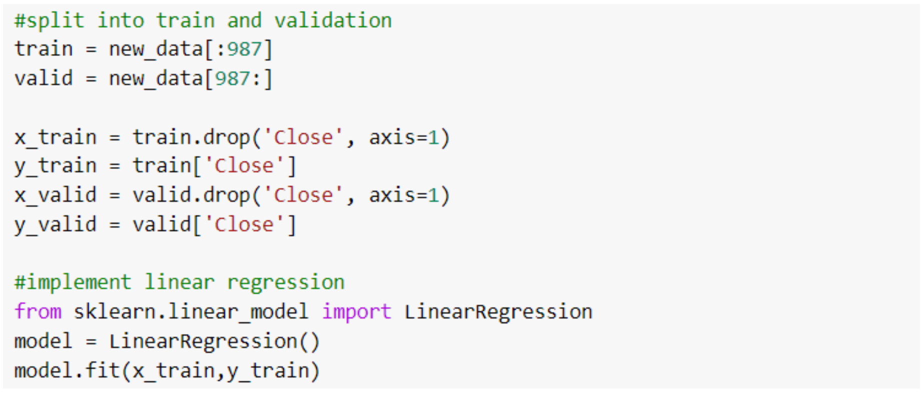
4.4.3. Result

Through the experiment of the above steps, we get the following results:
Open = 4.397517349189246
High = 18.77345060206976
Low = 6.779475154809312
Close = 9.5129595375735
Adj Close = 10.814079716011829
From the results it can be said that open price will be most accurate of all features. As, we will be mainly dealing with the closed price, it will also provide a good accuracy according to the results.

4.5. Algorithm Analysis

4.5.1 <u>Traditional Machine Learning Methods</u>

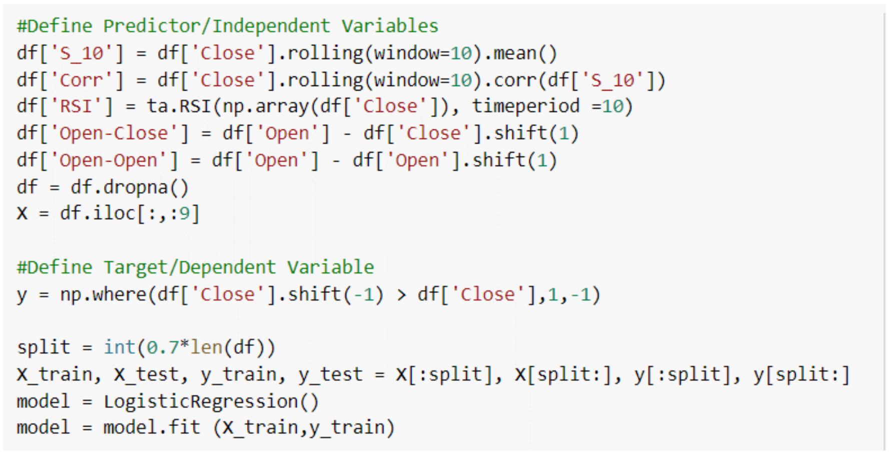
Linear Regression:



```
#split into train and validation
train = new_data[:987]
valid = new_data[987:]

x_train = train.drop('Close', axis=1)
y_train = train['Close']
x_valid = valid.drop('Close', axis=1)
y_valid = valid['Close']

#implement linear regression
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(x_train,y_train)
```

RMSE: 8.9

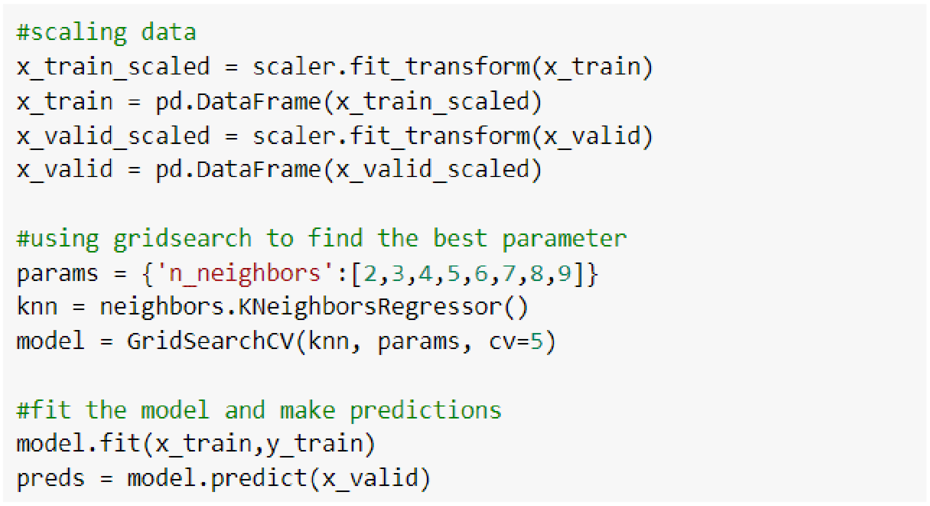
Logistic regression:



```
#Define Predictor/Independent Variables
df['S_10'] = df['Close'].rolling(window=10).mean()
df['Corr'] = df['Close'].rolling(window=10).corr(df['S_10'])
df['RSI'] = ta.RSI(np.array(df['Close']), timeperiod =10)
df['Open-Close'] = df['Open'] - df['Close'].shift(1)
df['Open-Open'] = df['Open'] - df['Open'].shift(1)
df = df.dropna()
X = df.iloc[:,:9]

#Define Target/Dependent Variable
y = np.where(df['Close'].shift(-1) > df['Close'],1,-1)

split = int(0.7*len(df))
X_train, X_test, y_train, y_test = X[:split], X[split:], y[:split], y[split:]
model = LogisticRegression()
model = model.fit (X_train,y_train)
```

RMSE:8.012

K-Nearest Neighbor (KNN):



```
#scaling data
x_train_scaled = scaler.fit_transform(x_train)
x_train = pd.DataFrame(x_train_scaled)
x_valid_scaled = scaler.fit_transform(x_valid)
x_valid = pd.DataFrame(x_valid_scaled)

#using gridsearch to find the best parameter
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
knn = neighbors.KNeighborsRegressor()
model = GridSearchCV(knn, params, cv=5)

#fit the model and make predictions
model.fit(x_train,y_train)
preds = model.predict(x_valid)
```

RMSE:29.069

Gaussian Naïve Bayes:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
```

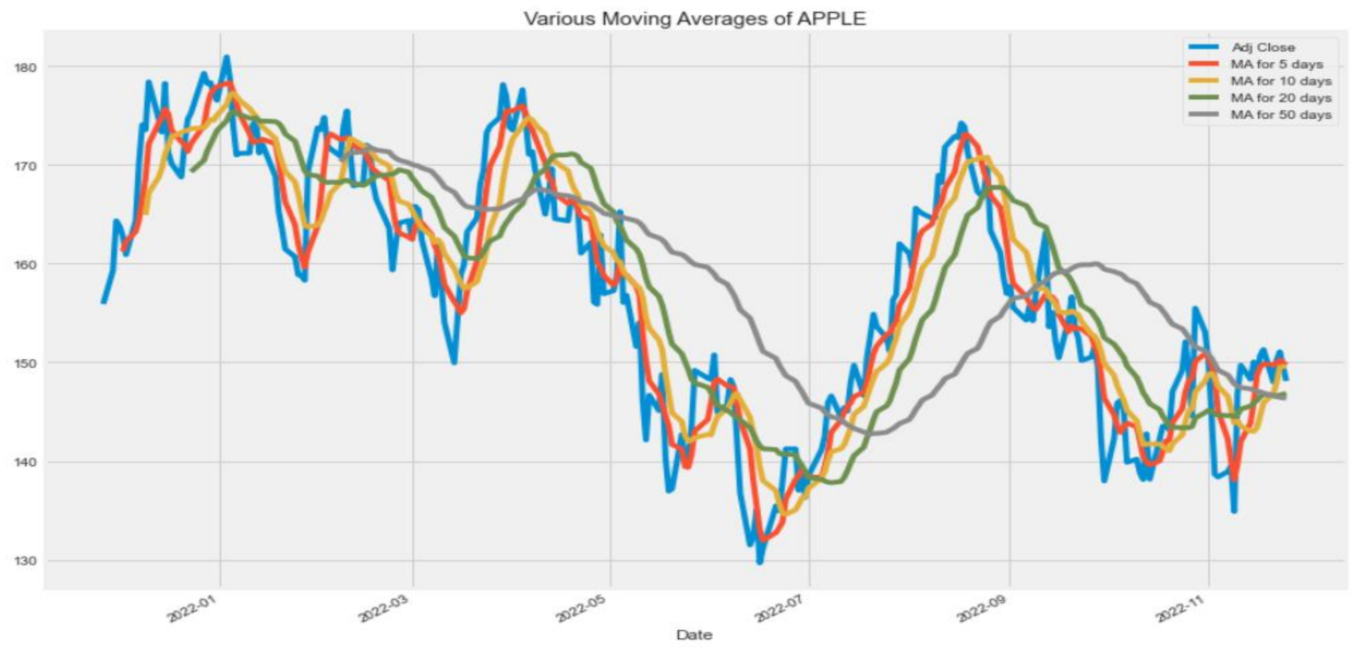
RMSE:11.07

4.5.2 Time Series Analysis Methods

Moving Averages:

Moving Averages are not true prediction models; however, it is an important topic to demonstrate. When you hear someone talk about how they want to "de-trend" or "smooth" data they are usually talking about implementing some sort of moving average. There are multiple moving average types with the most common being simple and exponential. Simple is just the average price over the desired time span. Exponential is a little more complicated as it provides a weight factor to each time step in the window. The weights are applied to make the more recent time steps more important that the later time steps. This allows the moving average to respond much more quickly to abrupt changes.

Here, simple moving average is only used.



RMSE of Weekly Moving Average (5-day SMA): 6.8635
RMSE of Biweekly Moving Average (10-day SMA): 16.6689
RMSE of Monthly Moving Average (20-day SMA): 26.54196

ARIMA:

```
from pyramid.arima import auto_arima

data = df.sort_index(ascending=True, axis=0)

train = data[:987]
valid = data[987:]

training = train['Close']
validation = valid['Close']

model = auto_arima(training, start_p=1, start_q=1,max_p=3, max_q=3,
                   m=12,start_P=0, seasonal=True,d=1, D=1, trace=True,
                   error_action='ignore',suppress_warnings=True)
model.fit(training)

forecast = model.predict(n_periods=248)
forecast = pd.DataFrame(forecast,index = valid.index,columns=['Prediction'])
```

RMSE:11.12

Prophet:

```
#importing prophet
from fbprophet import Prophet
#creating dataframe
new_data = pd.DataFrame(index=range(0,len(df)),columns=['Date', 'Close'])

for i in range(0,len(data)):
    new_data['Date'][i] = data['Date'][i]
    new_data['Close'][i] = data['Close'][i]
new_data['Date'] = pd.to_datetime(new_data.Date,format='%Y-%m-%d')
new_data.index = new_data['Date']

#preparing data
new_data.rename(columns={'Close': 'y', 'Date': 'ds'}, inplace=True)

#train and validation
train = new_data[:987]
valid = new_data[987:]
#fit the model
model = Prophet()
model.fit(train)
#predictions
close_prices = model.make_future_dataframe(periods=len(valid))
forecast = model.predict(close_prices)
```

RMSE:14.408

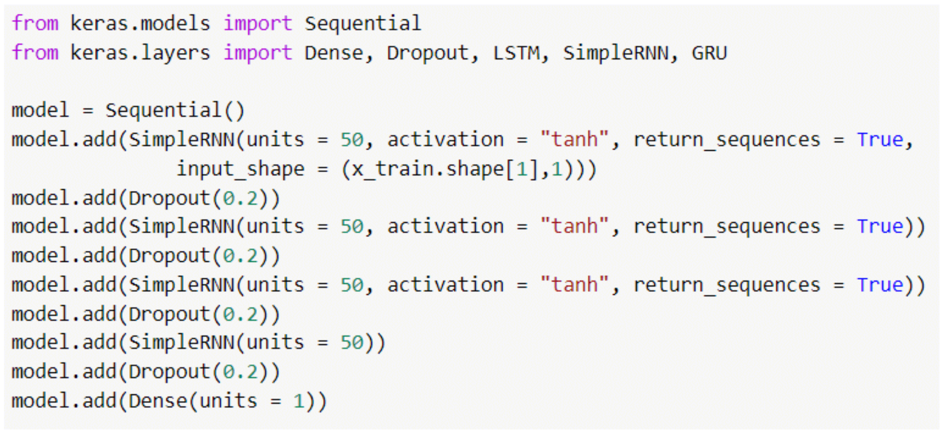
4.5.3 Deep Learning and Neural Networks

Dense forecast (For a control model):

```
# Clear back end
keras.backend.clear_session()

# Ensure reproducibility
tf.random.set_seed(42)
np.random.seed(42)

# Set Window Size
window_size = 30
train_set = window_dataset(normalized_x_train.flatten(), window_size)
valid_set = window_dataset(normalized_x_valid.flatten(), window_size)

# Build 2 layer model with 10 neurons each and 1 output layer
model = keras.models.Sequential([
  keras.layers.Dense(10, activation="relu", input_shape=[window_size]),
  keras.layers.Dense(10, activation="relu"),
  keras.layers.Dense(1)
])
```

RMSE: 14.4141

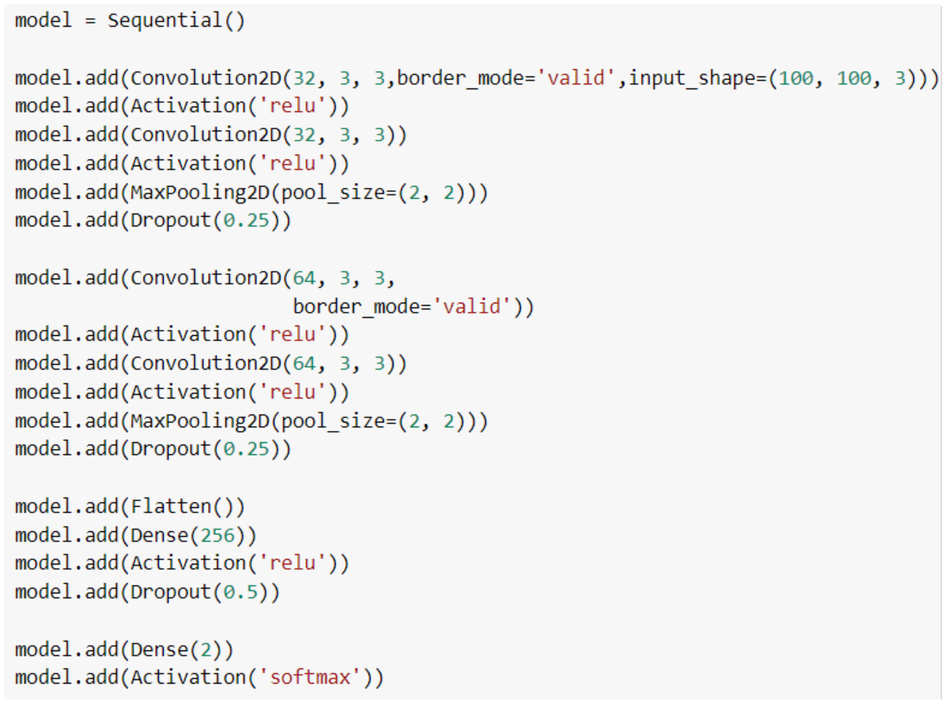
Simple Recurrent Neural Network (RNN):



```
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, SimpleRNN, GRU

model = Sequential()
model.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True,
              input_shape = (x_train.shape[1],1)))
model.add(Dropout(0.2))
model.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True))
model.add(Dropout(0.2))
model.add(SimpleRNN(units = 50, activation = "tanh", return_sequences = True))
model.add(Dropout(0.2))
model.add(SimpleRNN(units = 50))
model.add(Dropout(0.2))
model.add(Dense(units = 1))
```

RMSE: 14.4141

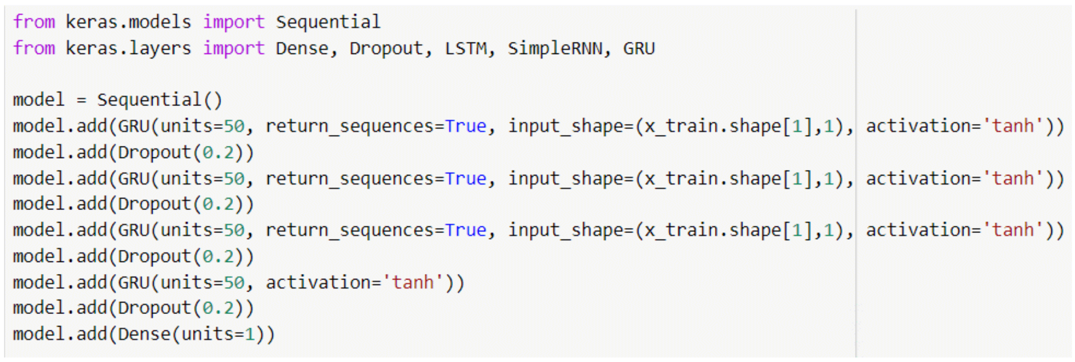
Convolutional Neural Network (CNN):



```
model = Sequential()

model.add(Convolution2D(32, 3, 3,border_mode='valid',input_shape=(100, 100, 3)))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3,
                        border_mode='valid'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dropout(0.5))

model.add(Dense(2))
model.add(Activation('softmax'))
```

RMSE: 14.4141

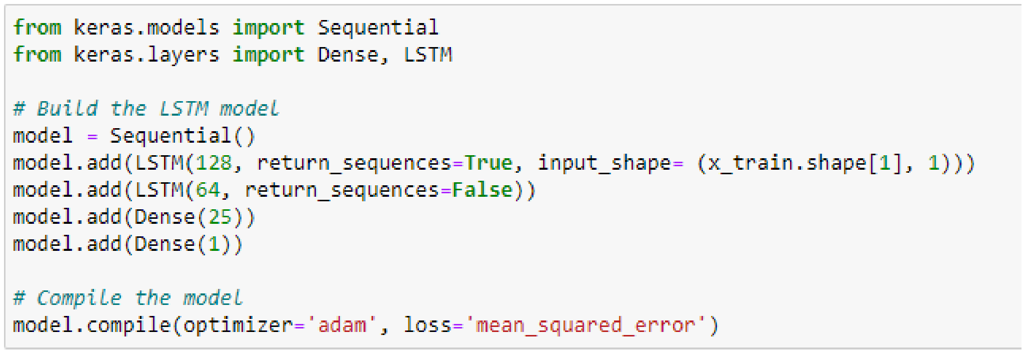
Gated Recurrent Unit (GRU):



```
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM, SimpleRNN, GRU

model = Sequential()
model.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1],1), activation='tanh'))
model.add(Dropout(0.2))
model.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1],1), activation='tanh'))
model.add(Dropout(0.2))
model.add(GRU(units=50, return_sequences=True, input_shape=(x_train.shape[1],1), activation='tanh'))
model.add(Dropout(0.2))
model.add(GRU(units=50, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(units=1))
```

RMSE: 14.4141

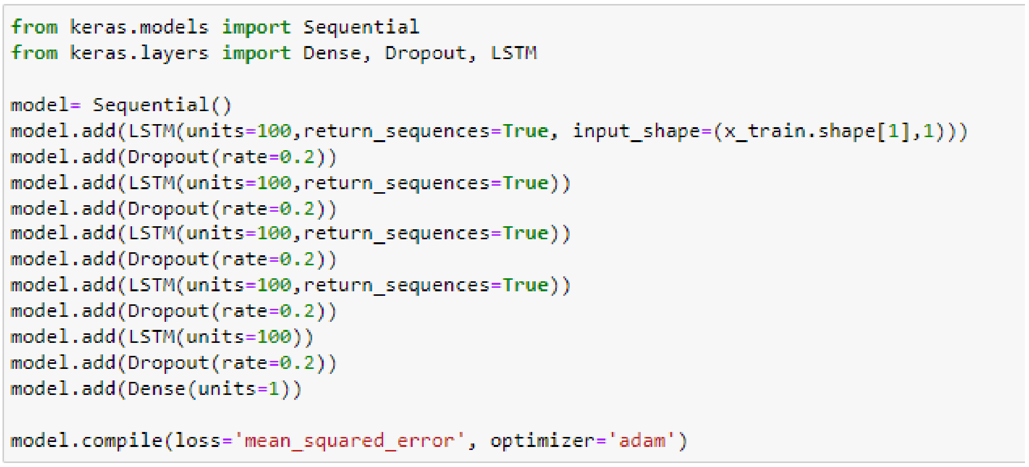
Long Short-Term Memory (LSTM) (without Dropout regularization):



```python
from keras.models import Sequential
from keras.layers import Dense, LSTM

# Build the LSTM model
model = Sequential()
model.add(LSTM(128, return_sequences=True, input_shape= (x_train.shape[1], 1)))
model.add(LSTM(64, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

RMSE: 3.3644

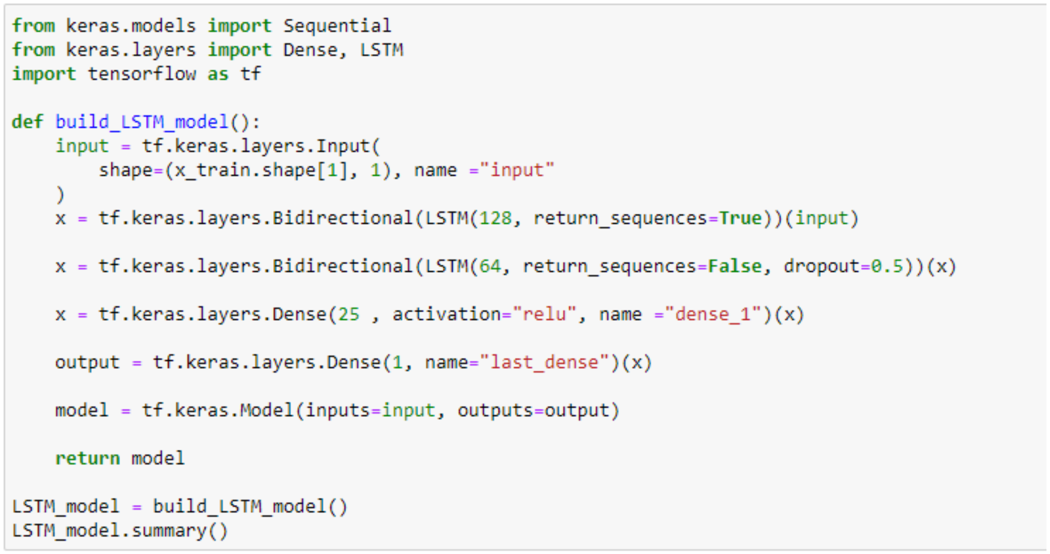
Long Short-Term Memory (LSTM) (with Dropout regularization):



```python
from keras.models import Sequential
from keras.layers import Dense, Dropout, LSTM

model= Sequential()
model.add(LSTM(units=100,return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(Dropout(rate=0.2))
model.add(LSTM(units=100,return_sequences=True))
model.add(Dropout(rate=0.2))
model.add(LSTM(units=100,return_sequences=True))
model.add(Dropout(rate=0.2))
model.add(LSTM(units=100,return_sequences=True))
model.add(Dropout(rate=0.2))
model.add(LSTM(units=100))
model.add(Dropout(rate=0.2))
model.add(Dense(units=1))

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

RMSE: 4.50203

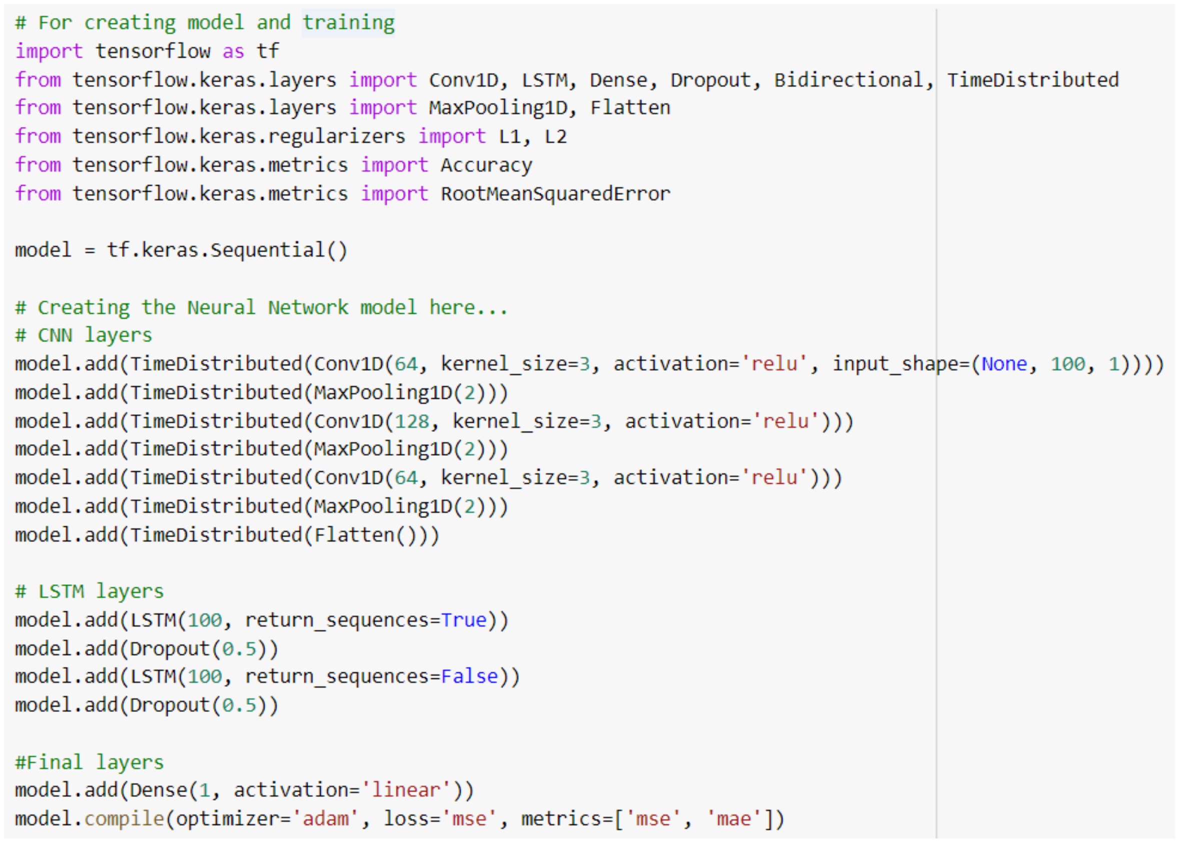
Bidirectional LSTM (Bi-LSTM) (with various combination of Hyperparameters):



```python
from keras.models import Sequential
from keras.layers import Dense, LSTM
import tensorflow as tf

def build_LSTM_model():
    input = tf.keras.layers.Input(
        shape=(x_train.shape[1], 1), name ="input"
    )
    x = tf.keras.layers.Bidirectional(LSTM(128, return_sequences=True))(input)

    x = tf.keras.layers.Bidirectional(LSTM(64, return_sequences=False, dropout=0.5))(x)

    x = tf.keras.layers.Dense(25 , activation="relu", name ="dense_1")(x)

    output = tf.keras.layers.Dense(1, name="last_dense")(x)

    model = tf.keras.Model(inputs=input, outputs=output)

    return model

LSTM_model = build_LSTM_model()
LSTM_model.summary()
```

RMSE: 3.286326

Hybrid LSTM (For example: CNN-LSTM & CNN-Bi-LSTM):



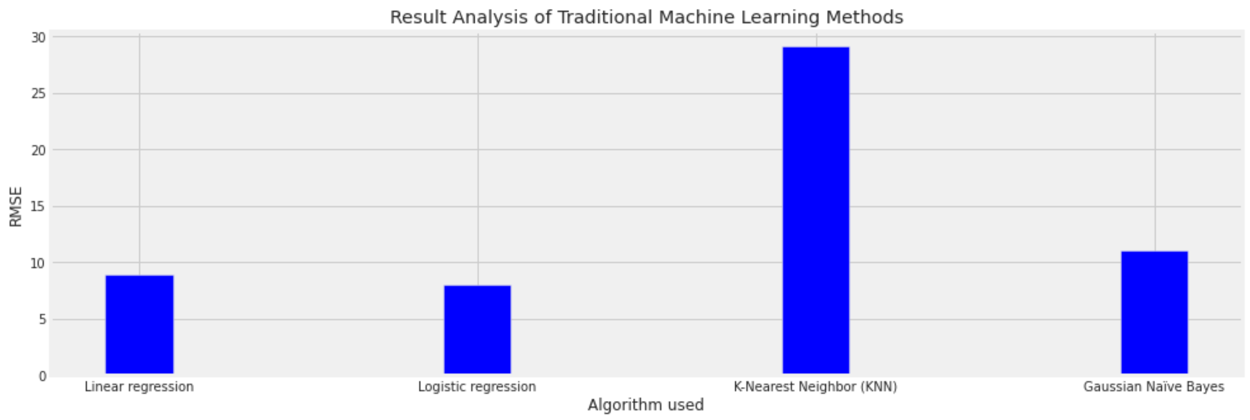
```
# For creating model and training
import tensorflow as tf
from tensorflow.keras.layers import Conv1D, LSTM, Dense, Dropout, Bidirectional, TimeDistributed
from tensorflow.keras.layers import MaxPooling1D, Flatten
from tensorflow.keras.regularizers import L1, L2
from tensorflow.keras.metrics import Accuracy
from tensorflow.keras.metrics import RootMeanSquaredError

model = tf.keras.Sequential()

# Creating the Neural Network model here...
# CNN layers
model.add(TimeDistributed(Conv1D(64, kernel_size=3, activation='relu', input_shape=(None, 100, 1))))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Conv1D(128, kernel_size=3, activation='relu')))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Conv1D(64, kernel_size=3, activation='relu')))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Flatten()))

# LSTM layers
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(100, return_sequences=False))
model.add(Dropout(0.5))

#Final layers
model.add(Dense(1, activation='linear'))
model.compile(optimizer='adam', loss='mse', metrics=['mse', 'mae'])
```

RMSE: 2.28643

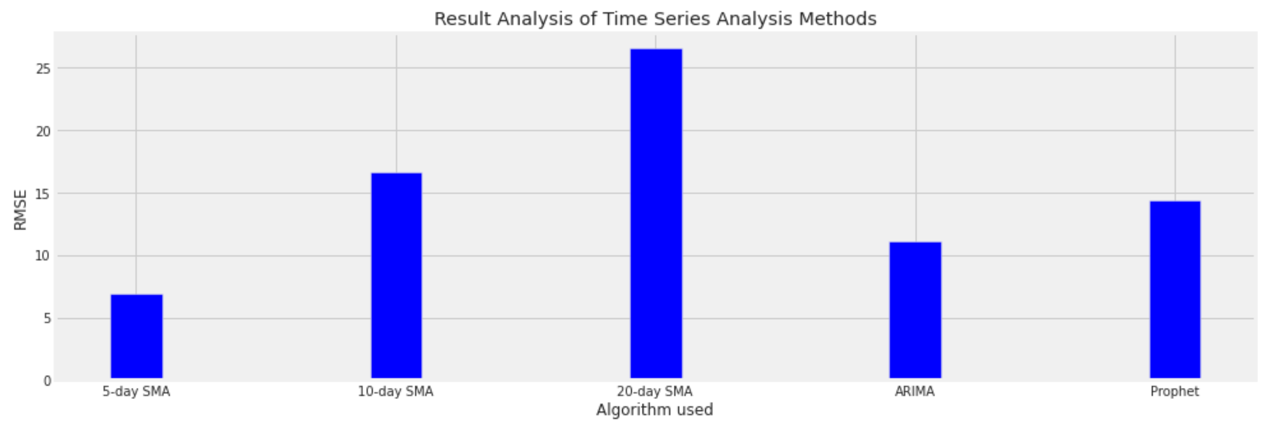
4.6. Comparison of Algorithms:

<u>RMSEs of Traditional Machine Learning Methods</u>

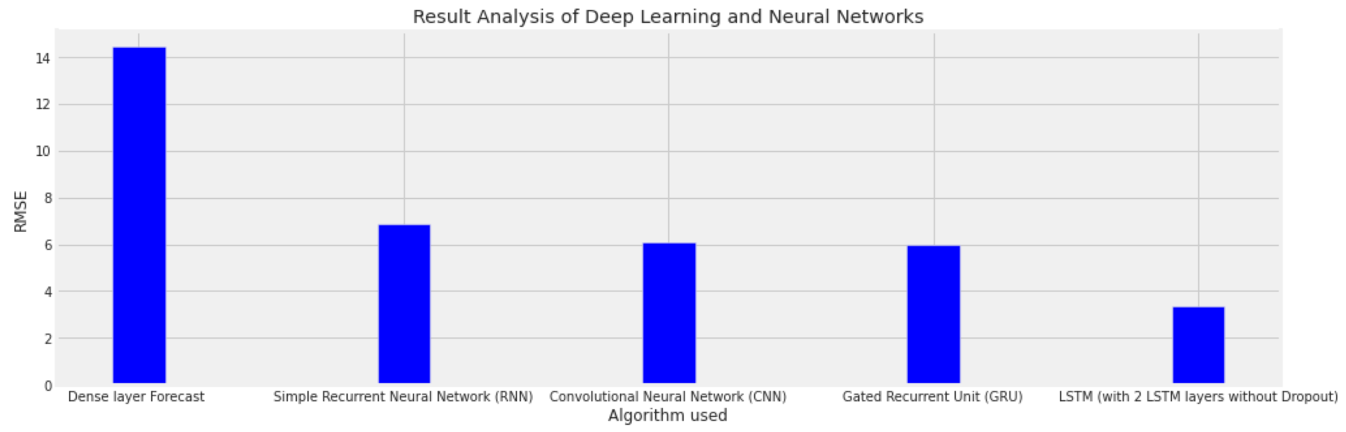
Linear regression:8.9,
Logistic regression:8.012,
K-Nearest Neighbor (KNN):29.069,
Gaussian Naïve Bayes:11.07



<u>RMSEs of Strictly Time Series Analysis Methods</u>
5-day SMA:6.8635,
10-day SMA:16.6689,
20-day SMA:26.54196,
ARIMA:11.12,
Prophet:14.408



<u>**RMSEs of Deep Learning and Neural Networks**</u>
Dense layer Forecast:14.4141,
Simple Recurrent Neural Network (RNN):6.8712,
Convolutional Neural Network (CNN):6.0812,
Gated Recurrent Unit (GRU):5.96762,
LSTM (with 2 LSTM layers without Dropout):3.36544



After analyzing so far, we have come to a conclusion that LSTM is the best model so far in these three broad categories of Algorithms. So, in the next comparisons we will be covering the variants of LSTM, BiLSTM & Hybrid LSTMs in a trial-and-error manner.

4.7. LSTM layer composition

4.7.1. Choosing the right number of nodes and layers

There is no final, definite, rule of thumb on how many nodes (or hidden neurons) or how many layers one should choose, and very often a trial-and-error approach will give you the best results for your individual problem. The most common framework for this is most likely the k-fold cross-validation. However, even for a testing procedure, we need to choose some numbers of nodes.

If the problem is simple, there are many rule-of-thumb methods for determining the correct number of neurons to use in the hidden layers, such as the following:

1. The number of hidden neurons should be between the size of the input layer and the size of the output layer.

2. The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.

3. The number of hidden neurons should be less than twice the size of the input layer.

Moreover, the number of neurons and number of layers required for the hidden layer also depends upon training cases, number of outliers, the complexity of, the data that is to be learned, and the type of activation functions used. The following formula may give a starting point in many complex cases:

$$N_h = \frac{N_s}{(\alpha * (N_i + N_o))}$$

N_i is the number of input neurons, N_o the number of output neurons, N_s the number of samples in the training data, and α represents a scaling factor that is usually between 2 and 10. We can calculate 8 different numbers to feed into our validation procedure and find the optimal model, based on the resulting validation loss.

Ideal number for any given use case will be different and is best to be decided by running different models against each other. Generally, 2 layers have shown to be enough to detect more complex features. More layers can be better but also harder to train. As a general rule of thumb, 1 hidden layer work with simple problems, like this, and two are enough to find reasonably complex features.

4.7.2. Choosing additional Hyper-Parameters

In most cases, Every LSTM layer should be accompanied by a Dropout layer. This layer will help to prevent overfitting by ignoring randomly selected neurons during training, and hence reduces the sensitivity to the specific weights of individual neurons.

After LSTM layer(s) did all the work to transform the input to make predictions towards the desired output possible, we have to reduce (or, in rare cases extend) the shape, to match our desired output. In our case, we have two output labels and therefore we need two-output units.

The final layer to add is the activation layer. Loss function and activation function are often chosen together. For choosing the optimizer, adaptive moment estimation, Adam, has been shown to work well in most practical applications and works well with only little changes in the hyperparameters.

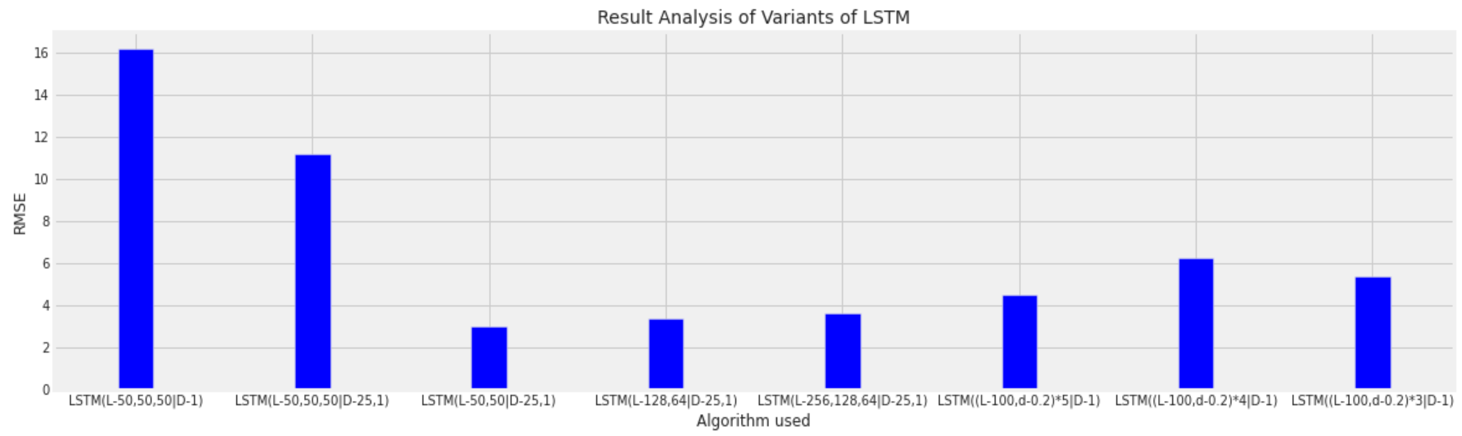
While Keras frees us from writing complex deep learning algorithms, we still have to make choices regarding some of the hyperparameters along the way. In some cases, e.g. choosing the right activation function, we can rely on rules of thumbs or can determine the right parameter based on our problem. However, in some other cases, the best result will come from testing various configurations and then evaluating the outcome. Keeping these factors in mind we will be analyzing the various composition of LSTMs.

4.8. Comparison of Variants of LSTM, BiLSTM & Hybrid LSTMs

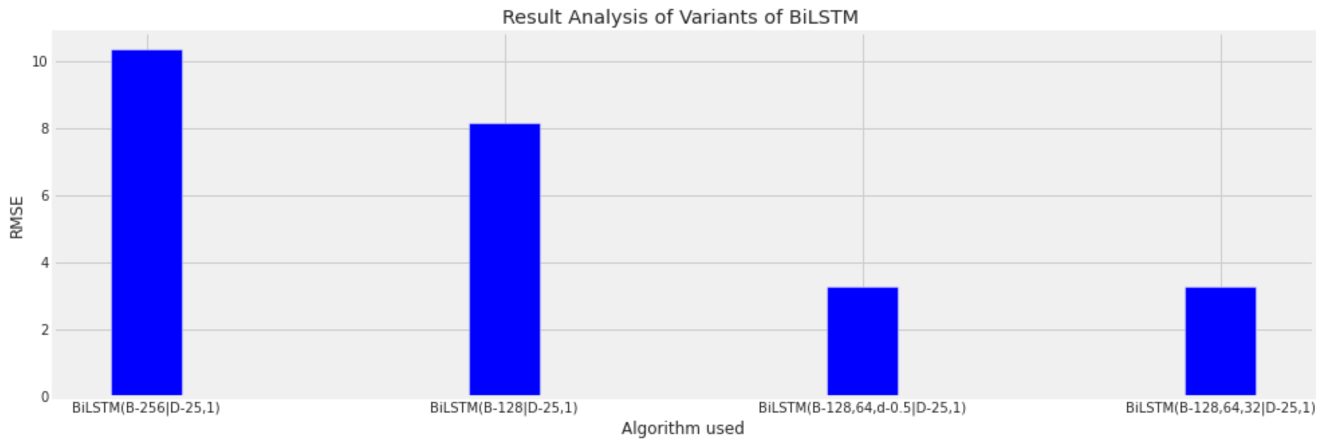
Legend: L - LSTM Layers, B - BiLSTM Layers, D - Dense Layers,

d - Dropout Layers, C - CNN Layers

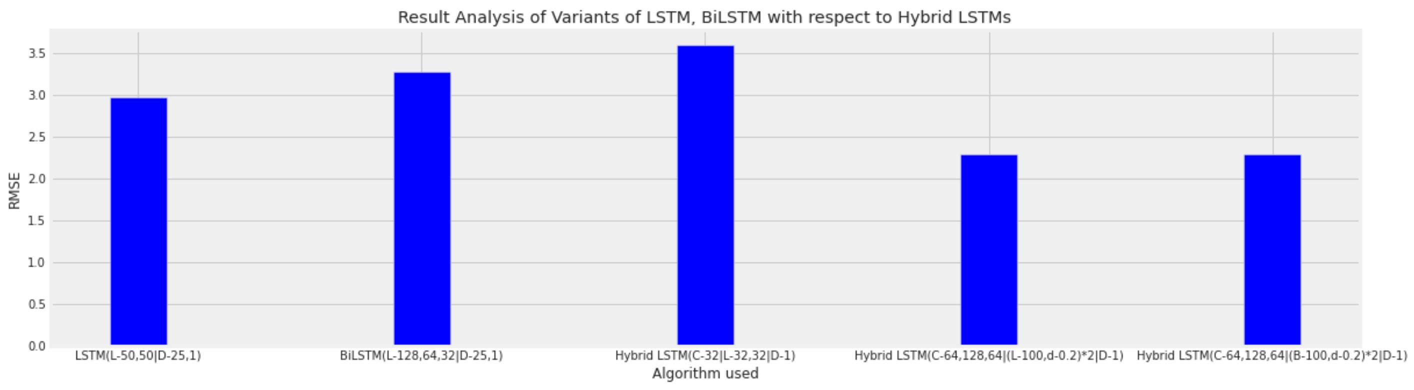
RMSEs of Variants of LSTM
LSTM(L-50,50,50|D-1):16.168,
LSTM(L-50,50,50|D-25,1):11.169778,
LSTM(L-50,50|D-25,1):2.9759,
LSTM(L-128,64|D-25,1):3.3644,
LSTM(L-256,128,64|D-25,1):3.59565,
LSTM((L-100, d-0.2)*5|D-1):4.50203,
LSTM((L-100, d-0.2)*4|D-1):6.25669,
LSTM((L-100, d-0.2)*3|D-1):5.35838



RMSEs of Variants of BiLSTM
BiLSTM(B-256|D-25,1):10.3486,
BiLSTM(B-128|D-25,1):8.12783,
BiLSTM(B-128,64,d-0.5|D-25,1):3.286326,
BiLSTM(B-128,64,32|D-25,1):3.276113



RMSEs of Variants of LSTM, BiLSTM with respect to Hybrid LSTMs
Best performed LSTM(L-50,50|D-25,1):2.9759,
Best performed BiLSTM(B-128,64,32|D-25,1):3.276113,
Hybrid LSTM(C-32|L-32,32|D-1):3.5956,
Hybrid LSTM (C-64,128,64|(L-100, d-0.2)*2|D-1):2.28643,
Hybrid LSTM (C-64,128,64|(B-100, d-0.2)*2|D-1):2.2956



Clearly, the comparison is quite difficult for the overall best model as the errors and accuracies change with datasets, timeframes and we cannot possibly try all the existing combinations of layers. But here in this case, Hybrid LSTMs performed the best.

4.9. Result Analysis

➢ **Summary of All Model Results**

After comparing Traditional Machine Learning Methods, Strictly Time Series Analysis Methods and Deep Learning and Neural Networks, we have come to a conclusion that for this dataset LSTM is the best model so far in these three broad categories of Algorithms.

So, by the further comparison of the variants of LSTM, BiLSTM & Hybrid LSTMs in a trial-and-error manner, we tried to find the most accurate combination of layers which will give the best results and the least amount of error. The difference of the errors in the best models of LSTM, BiLSTM & Hybrid LSTM are really negligible but the hybrid LSTM performed better than the other final models.

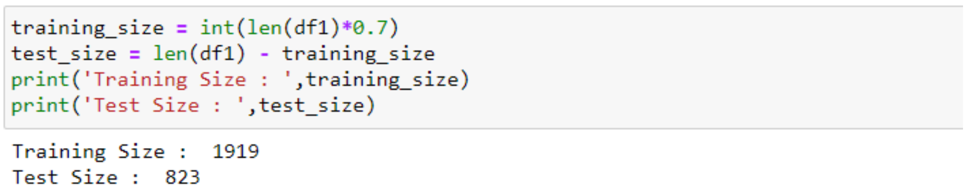
➢ **Best Model**

The results in various research papers show that additional training of data in BiLSTM-based modeling or the hybrid LSTM offers better predictions than regular LSTM-based models or the ARIMA models.

But in this case Hybrid CNN-LSTM model with dropout regularization performed the best, even better than BiLSTM variant of this particular hybrid model, which is quite surprising.

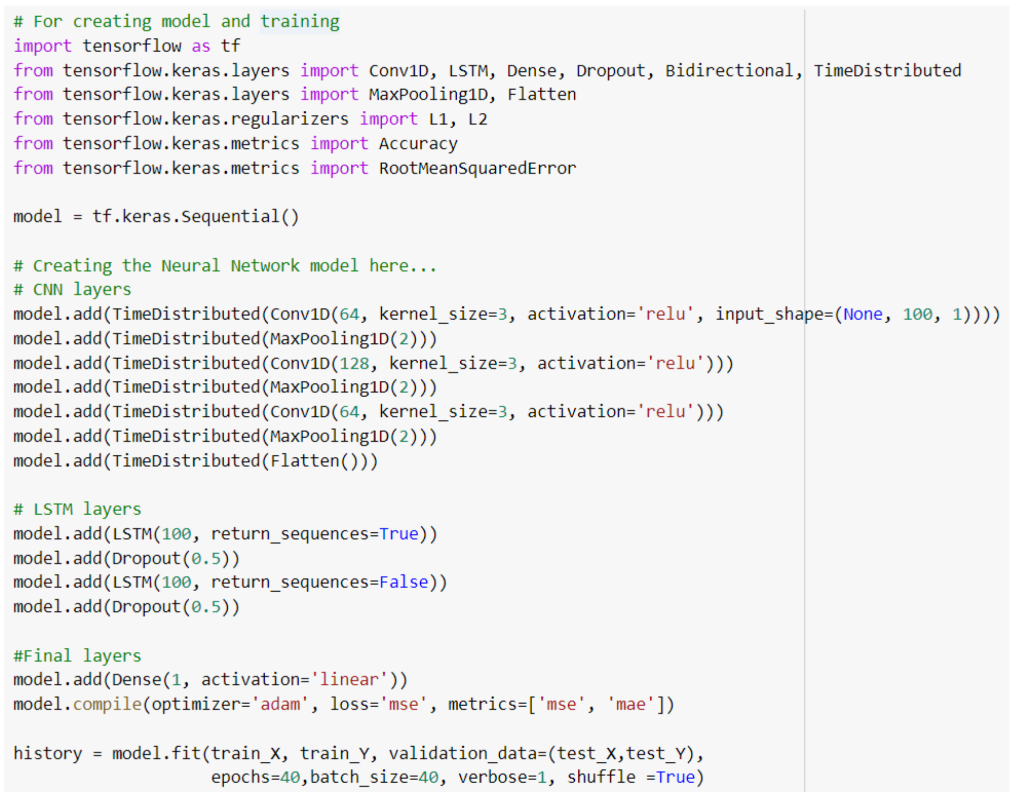
4.10. Final Predictions

➢ 30-day prediction using Hybrid CNN-LSTM



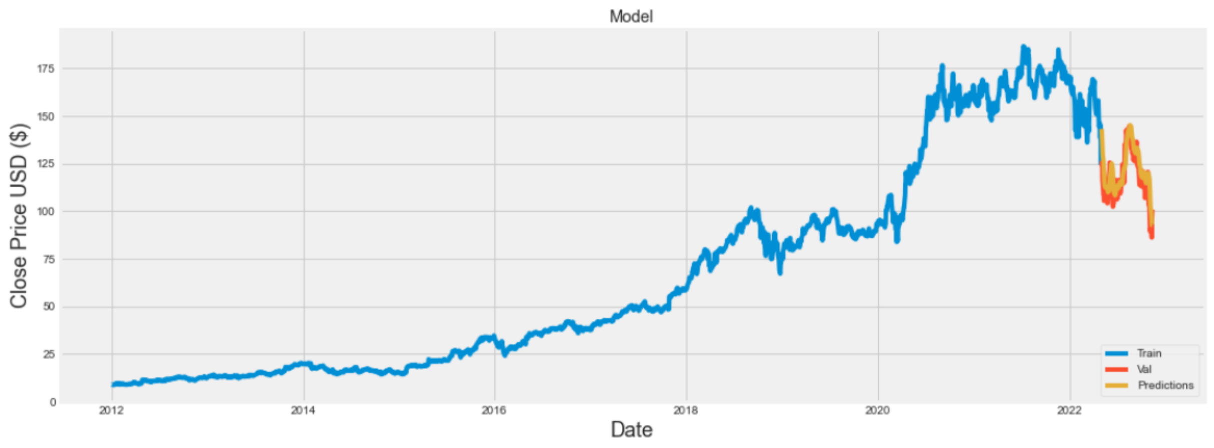
```
training_size = int(len(df1)*0.7)
test_size = len(df1) - training_size
print('Training Size : ',training_size)
print('Test Size : ',test_size)
```

```
Training Size :  1919
Test Size :  823
```

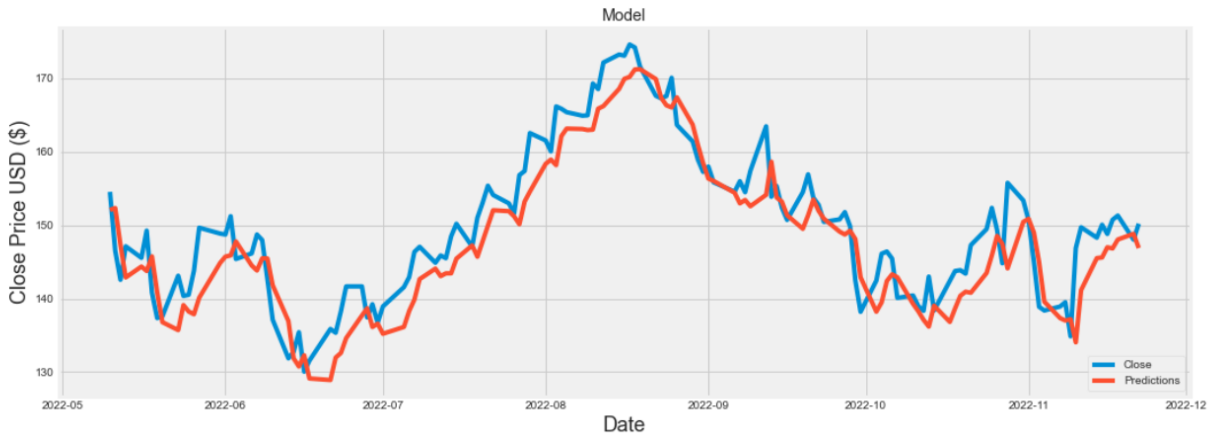


```
# For creating model and training
import tensorflow as tf
from tensorflow.keras.layers import Conv1D, LSTM, Dense, Dropout, Bidirectional, TimeDistributed
from tensorflow.keras.layers import MaxPooling1D, Flatten
from tensorflow.keras.regularizers import L1, L2
from tensorflow.keras.metrics import Accuracy
from tensorflow.keras.metrics import RootMeanSquaredError

model = tf.keras.Sequential()

# Creating the Neural Network model here...
# CNN layers
model.add(TimeDistributed(Conv1D(64, kernel_size=3, activation='relu', input_shape=(None, 100, 1))))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Conv1D(128, kernel_size=3, activation='relu')))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Conv1D(64, kernel_size=3, activation='relu')))
model.add(TimeDistributed(MaxPooling1D(2)))
model.add(TimeDistributed(Flatten()))

# LSTM layers
model.add(LSTM(100, return_sequences=True))
model.add(Dropout(0.5))
model.add(LSTM(100, return_sequences=False))
model.add(Dropout(0.5))

#Final layers
model.add(Dense(1, activation='linear'))
model.compile(optimizer='adam', loss='mse', metrics=['mse', 'mae'])

history = model.fit(train_X, train_Y, validation_data=(test_X,test_Y),
                    epochs=40,batch_size=40, verbose=1, shuffle =True)
```

- Model accuracy graph



- Model accuracy graph – detailed view

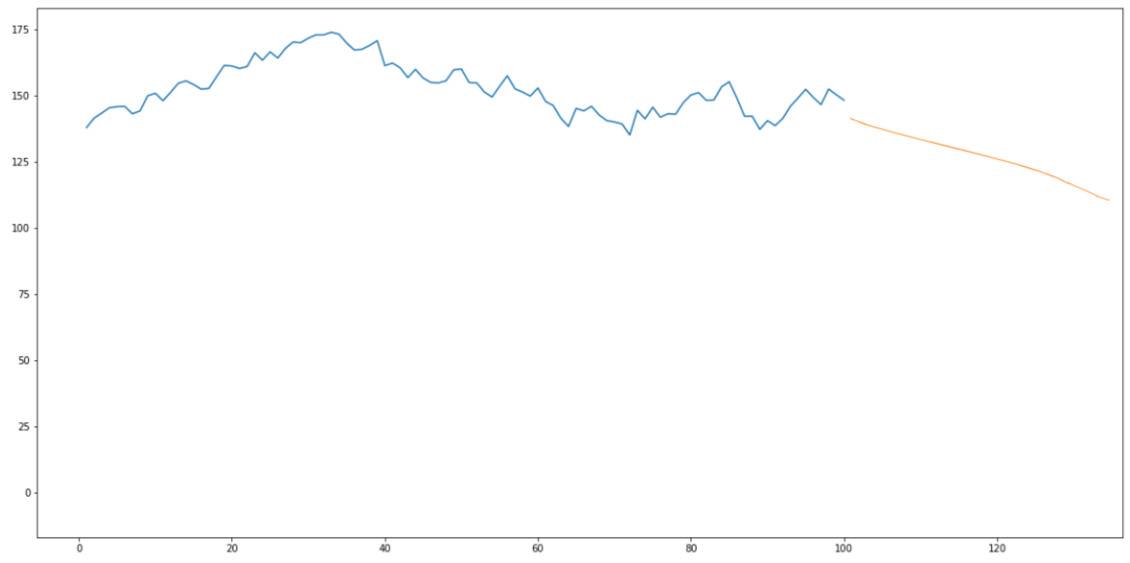


Predicting the future stock

- Plotting the predicted data (Here, for next 30 days taking past 100 days data as input)

```
plt.plot(day_new,scaler.inverse_transform(df1[len(df1)-100:]))
plt.plot(day_pred,scaler.inverse_transform(lst_output))
```

```
[<matplotlib.lines.Line2D at 0x14fcd8fc520>]
```



➢ Prediction of the next day stock price data (Here, taking the data of past 60 days as input)

```
#Get last 60 days values and convert into array
last_60_days=new_df[-60:].values

#Scale the data to be values between 0
last_60_days_scaled=scaler.transform(last_60_days)

#Create an empty list
X_test=[]
#Appemd the past 60days
X_test.append(last_60_days_scaled)

#Conver the X_test data into numpy array
X_test = np.array(X_test)

#Reshape the data
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1],1))
#Get predicted scaled price
pred_price = model.predict(X_test)
#undo the scaling
pred_price=scaler.inverse_transform(pred_price)
print(f'Closing Price of AAPL tomorrow:{pred_price}')
```

```
[*********************100%***********************]  1 of 1 completed
1/1 [==============================] - 0s 37ms/step
Closing Price of AAPL tomorrow:[[150.81215]]
```

➢ Prediction verification with respect to actual dataset

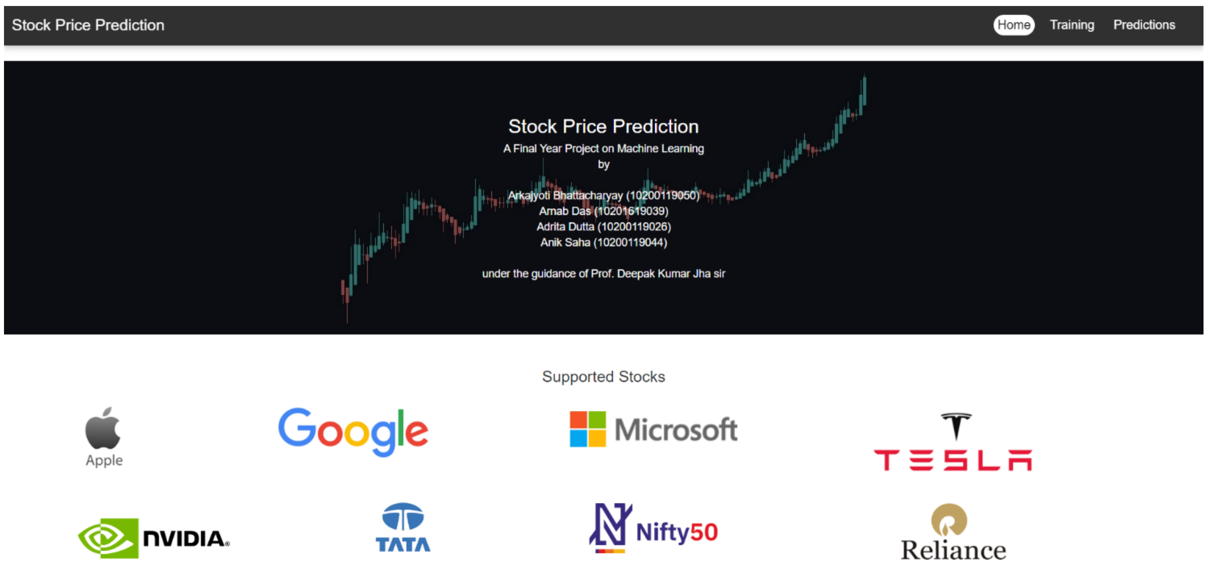
Date	Open	High	Low	Close	Adj Close	Volume
06-03-2023	153.79	156.3	153.46	153.83	153.83	87558000
07-03-2023	153.7	154.03	151.13	151.6	151.6	56182000
08-03-2023	152.81	153.47	151.83	152.87	152.87	47204800
09-03-2023	153.56	154.54	150.23	150.59	150.59	53833600
10-03-2023	150.21	150.94	147.61	148.5	148.5	68524400

Predicted Closing Price of 'tomorrow' is 150.81215 and the Actual Closing Price of 'tomorrow' is 148.50. Hence, the prediction of the final model is pretty accurate with only 0.015% error.

4.11. Web Application

4.11.1. Frontend

Home Page



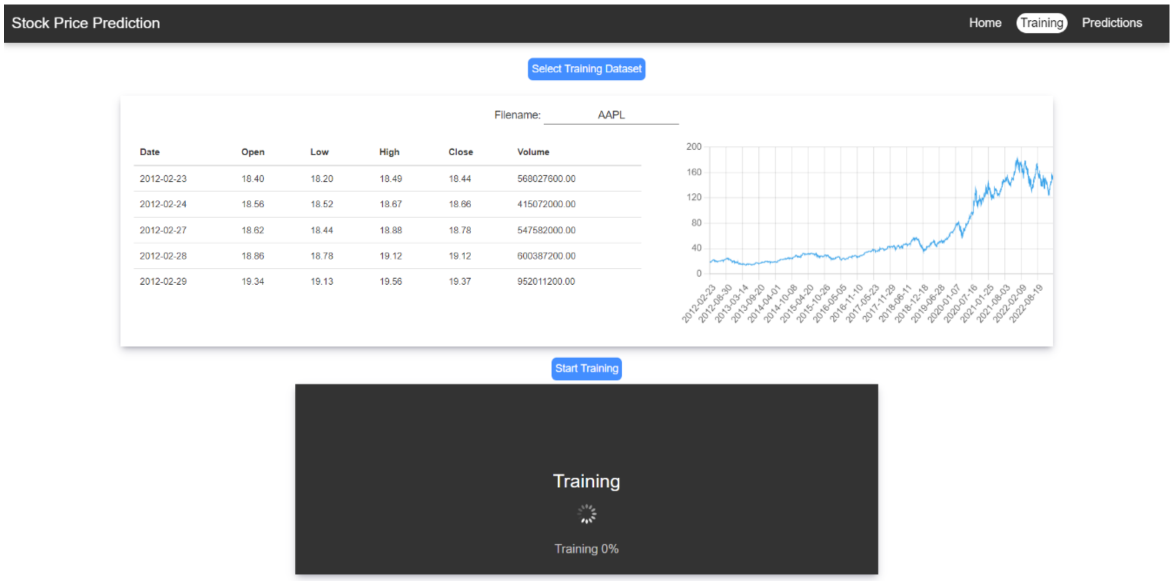
Stock Price Prediction
A Final Year Project on Machine Learning
by

Arkajyoti Bhattacharyay (10200119050)
Arnab Das (10201619039)
Adrita Dutta (10200119026)
Anik Saha (10200119044)

under the guidance of Prof. Deepak Kumar Jha sir

Supported Stocks

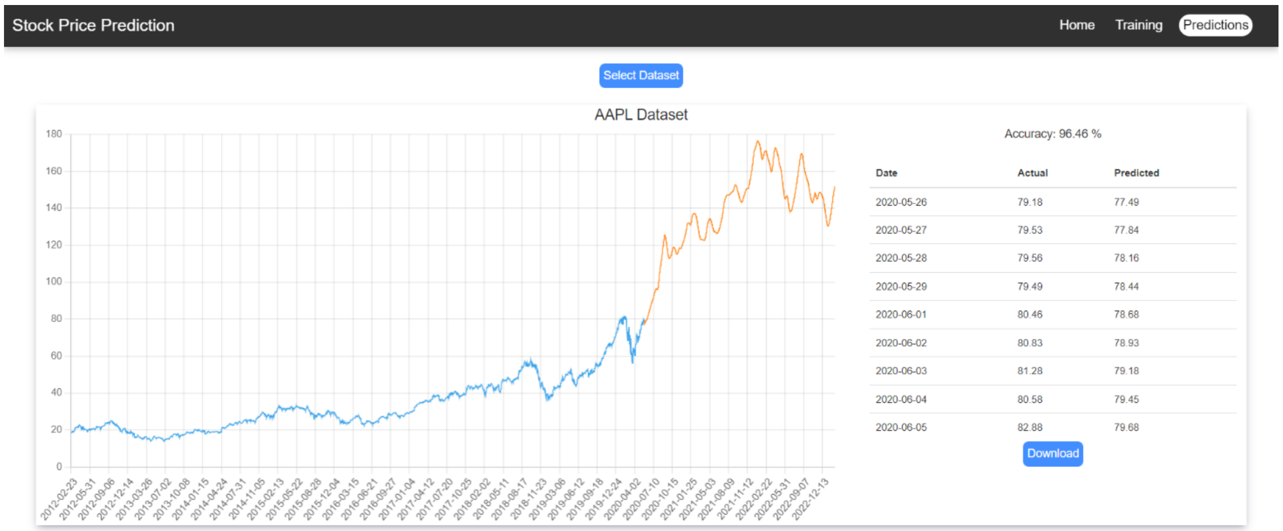
Training Page



Filename: AAPL

Date	Open	Low	High	Close	Volume
2012-02-23	18.40	18.20	18.49	18.44	568027600.00
2012-02-24	18.56	18.52	18.67	18.66	415072000.00
2012-02-27	18.62	18.44	18.88	18.78	547582000.00
2012-02-28	18.86	18.78	19.12	19.12	600387200.00
2012-02-29	19.34	19.13	19.56	19.37	952011200.00

Prediction Page



Accuracy: 96.46 %

Date	Actual	Predicted
2020-05-26	79.18	77.49
2020-05-27	79.53	77.84
2020-05-28	79.56	78.16
2020-05-29	79.49	78.44
2020-06-01	80.46	78.68
2020-06-02	80.83	78.93
2020-06-03	81.28	79.18
2020-06-04	80.58	79.45
2020-06-05	82.88	79.68

4.11.2. Backend

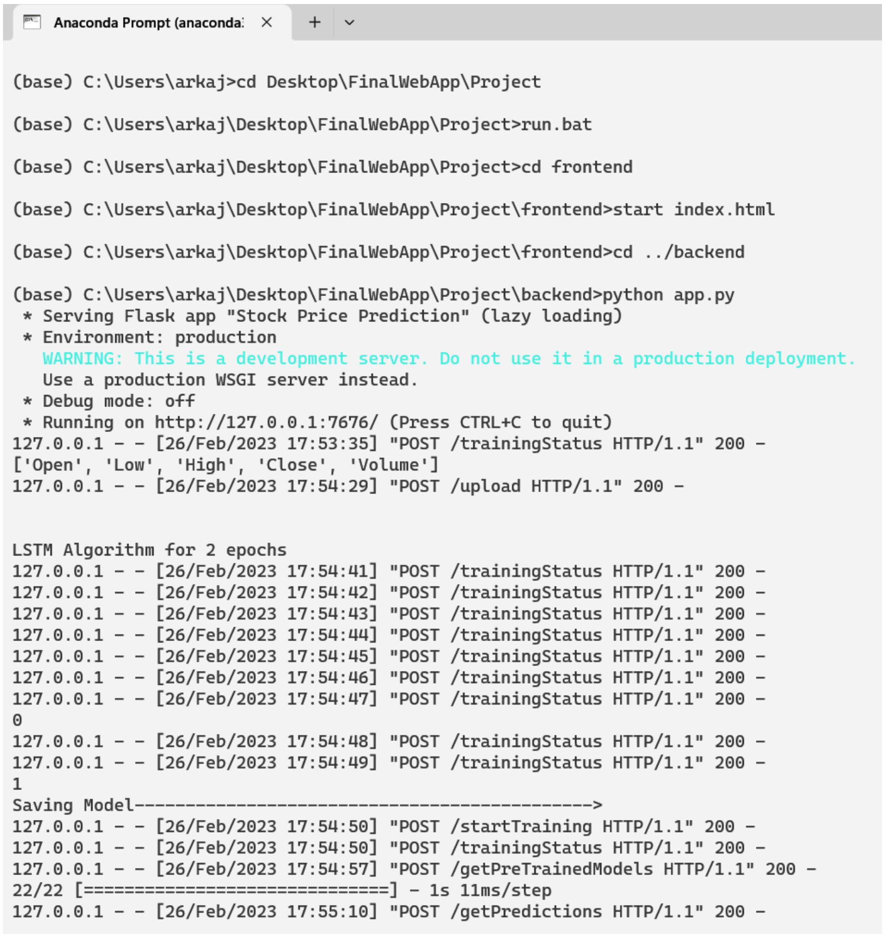
api.py file with Hybrid CNN-LSTM model is made.

app.py flask file with the execution parameters is created to run the model successfully.

4.11.3. Working

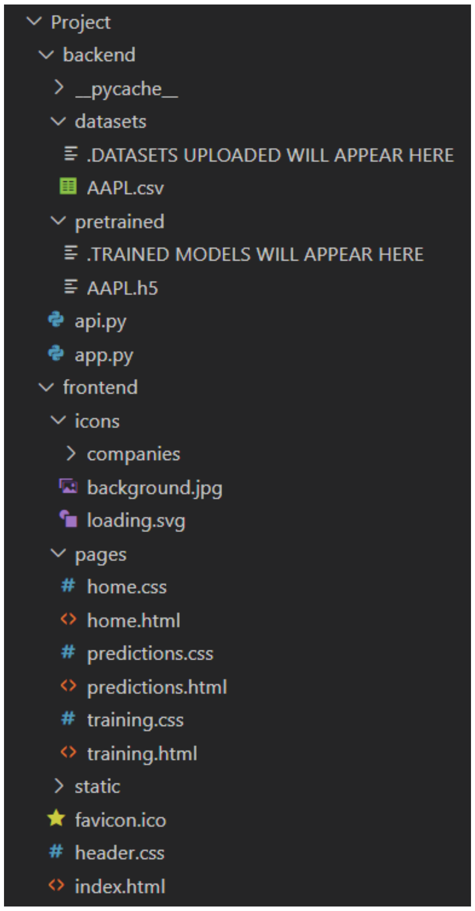
- First The python modules required for running the project i.e., pandas 1.4.4, tensorflow 2.9.2, keras 2.9.0, sklearn, flask 2.1.2, flask_cors 3.0.10 are to be downloaded.
- Then a batch file (.bat file) with the following commands is executed to synchronize the frontend and the backend.

```
cd frontend
start index.html
cd ../backend
python app.py
```



```
(base) C:\Users\arkaj>cd Desktop\FinalWebApp\Project

(base) C:\Users\arkaj\Desktop\FinalWebApp\Project>run.bat

(base) C:\Users\arkaj\Desktop\FinalWebApp\Project>cd frontend

(base) C:\Users\arkaj\Desktop\FinalWebApp\Project\frontend>start index.html

(base) C:\Users\arkaj\Desktop\FinalWebApp\Project\frontend>cd ../backend

(base) C:\Users\arkaj\Desktop\FinalWebApp\Project\backend>python app.py
 * Serving Flask app "Stock Price Prediction" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:7676/ (Press CTRL+C to quit)
127.0.0.1 - - [26/Feb/2023 17:53:35] "POST /trainingStatus HTTP/1.1" 200 -
['Open', 'Low', 'High', 'Close', 'Volume']
127.0.0.1 - - [26/Feb/2023 17:54:29] "POST /upload HTTP/1.1" 200 -


LSTM Algorithm for 2 epochs
127.0.0.1 - - [26/Feb/2023 17:54:41] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:42] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:43] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:44] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:45] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:46] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:47] "POST /trainingStatus HTTP/1.1" 200 -
0
127.0.0.1 - - [26/Feb/2023 17:54:48] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:49] "POST /trainingStatus HTTP/1.1" 200 -
1
Saving Model--------------------------------------------->
127.0.0.1 - - [26/Feb/2023 17:54:50] "POST /startTraining HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:50] "POST /trainingStatus HTTP/1.1" 200 -
127.0.0.1 - - [26/Feb/2023 17:54:57] "POST /getPreTrainedModels HTTP/1.1" 200 -
22/22 [==============================] - 1s 11ms/step
127.0.0.1 - - [26/Feb/2023 17:55:10] "POST /getPredictions HTTP/1.1" 200 -
```

Summary of the mechanism:



```
Project
  backend
    __pycache__
    datasets
      .DATASETS UPLOADED WILL APPEAR HERE
      AAPL.csv
    pretrained
      .TRAINED MODELS WILL APPEAR HERE
      AAPL.h5
    api.py
    app.py
  frontend
    icons
      companies
      background.jpg
      loading.svg
    pages
      home.css
      home.html
      predictions.css
      predictions.html
      training.css
      training.html
    static
    favicon.ico
    header.css
    index.html
```

As this is a local web application, the files should be properly organized (as in the directory overview).

Starting the index.html activates the frontend and
Starting the app.py executes the python backend model.

Frontend has three main pages,

- home.html with basic information
- training.html which has access to the dataset
- predictions.html which shows the result of the model

Backend model of the flask app is executed to train the dataset and predict the outcomes

- api.py file contains the Hybrid CNN-LSTM model
- app.py flask file with the execution parameters is executed to run the model successfully.

The results are displayed in the predictions.html page.

4.12. Streamlit Application

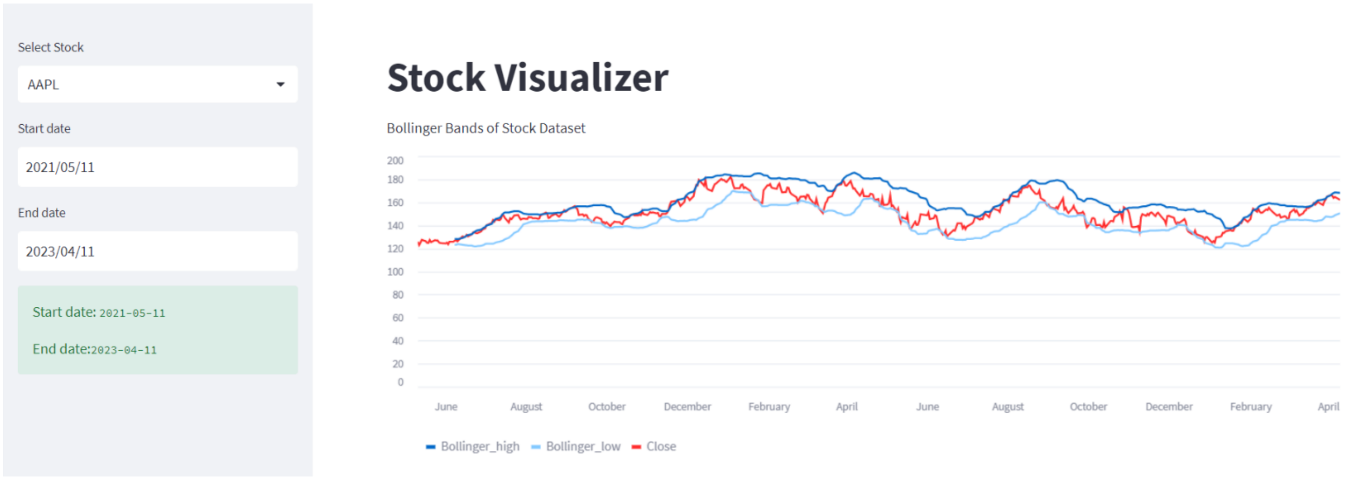
Streamlit is used make an online interactive dashboard for the visual charts related to the stock prediction. It lets users to select the stock symbols, and start and end dates in the sidebar area. It shows the stock prices with the Bollinger bands, and the MACD and RSI charts.

A sidebar is used to select stock data and to input the start date and the end date. The other parts of this streamlit app are listed below,

4.12.1. Components

1. Bollinger Bands

The Bollinger Bands are envelopes plotted at a standard deviation level above and below a simple moving average of the price. It helps to determine whether prices are high or low on a relative basis. They are used in pairs, both upper and lower bands, and in conjunction with a moving average. Further, the pair of bands is not intended to be used on its own. Use the pair to confirm signals given with other indicators. Bollinger Bands use 2 parameters, Period and Standard Deviations, StdDev. The default values are 20 for the period, and 2 for standard deviations, although the combinations may be customized.



Select Stock
AAPL
Start date
2021/05/11
End date
2023/04/11
Start date: `2021-05-11`
End date:`2023-04-11`

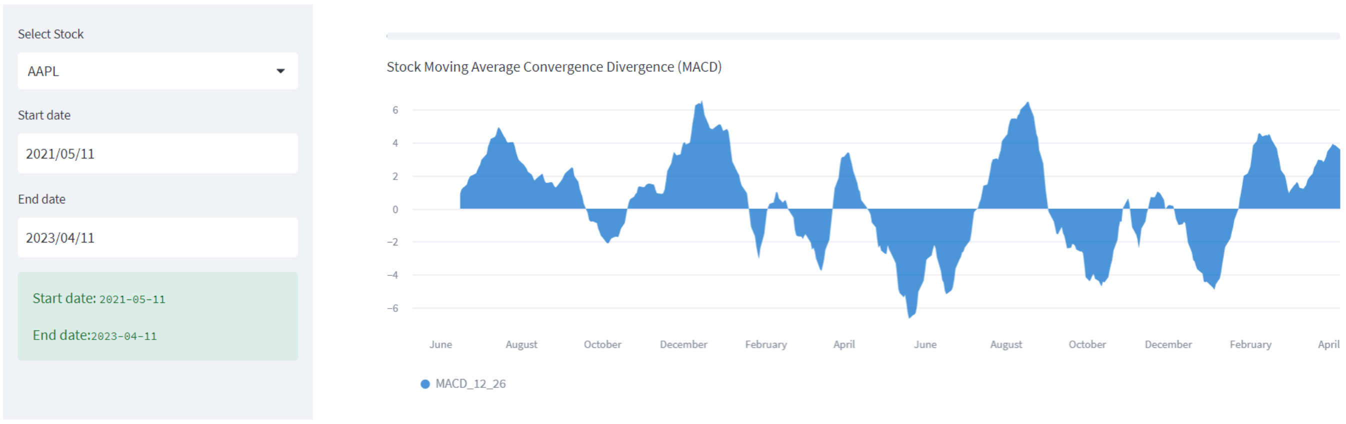
Stock Visualizer

Bollinger Bands of Stock Dataset

2. MACD (Moving Average Convergence Divergence)

The MACD can be used as a generic oscillator for any univariate series, not only price. Typically, MACD is set as the difference between the 12-period simple moving average (SMA) and 26-period simple moving average (MACD = 12-period SMA − 26-period SMA), or "fast SMA — slow SMA".

It has a positive value whenever the 12-period SMA is above the 26-period SMA and a negative value when the 12-period SMA is below the 26-period SMA. The more distant the MACD is above or below its baseline indicates that the distance between the two SMAs is growing.



Select Stock
AAPL
Start date
2021/05/11
End date
2023/04/11
Start date: `2021-05-11`
End date:`2023-04-11`

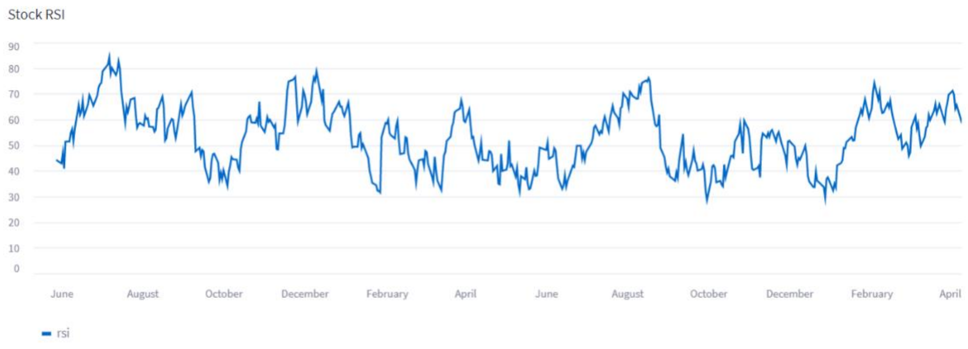
Stock Moving Average Convergence Divergence (MACD)

3. RSI (Relative Strength Index)

The relative strength index (RSI) measures the magnitude of recent price changes to evaluate overbought or oversold conditions. It is displayed as an oscillator and can have a reading from 0 to 100. The general rules are:

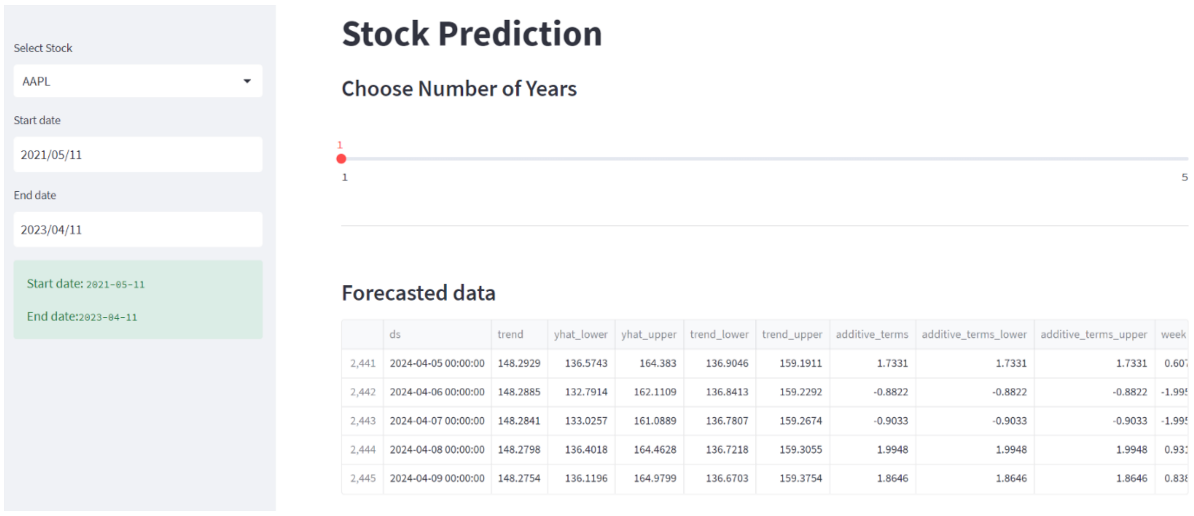
RSI >= 70: security is overvalued and may be primed for a trend reversal or corrective pullback in price.

RSI <= 30: an oversold or undervalued condition.



4. Main Prediction Graphs

Streamlit lets you plot different types of charts by using commands such as line_chart(), bar_chart(), and so on. One can also use seaborn, bokeh, or plotly for your preferences.



Select Stock

AAPL

Start date

2021/05/11

End date

2023/04/11

Start date: 2021-05-11

End date:2023-04-11

Stock Prediction

Choose Number of Years

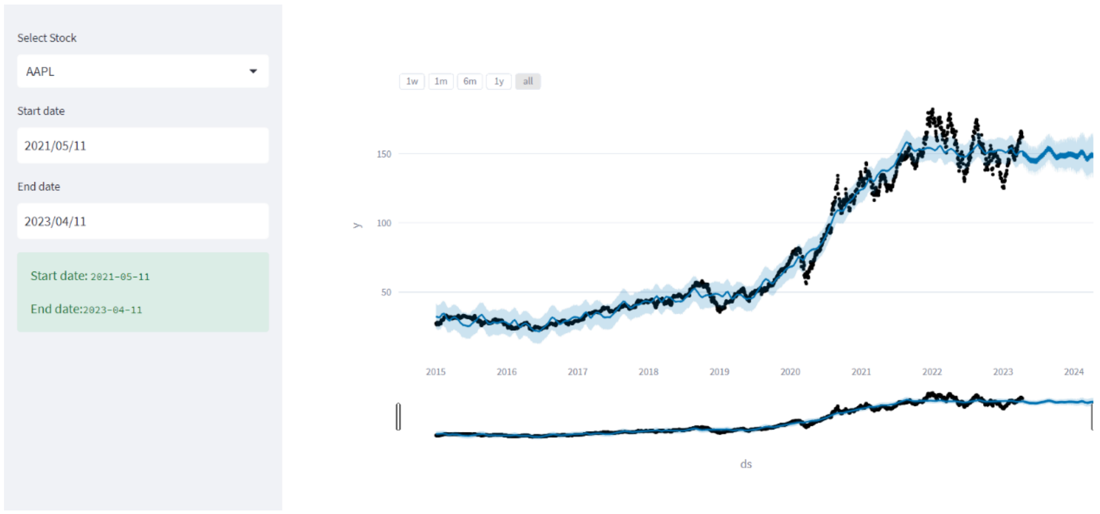
1

1 5

Forecasted data

	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	week
2,441	2024-04-05 00:00:00	148.2929	136.5743	164.383	136.9046	159.1911	1.7331	1.7331	1.7331	0.607
2,442	2024-04-06 00:00:00	148.2885	132.7914	162.1109	136.8413	159.2292	-0.8822	-0.8822	-0.8822	-1.995
2,443	2024-04-07 00:00:00	148.2841	133.0257	161.0889	136.7807	159.2674	-0.9033	-0.9033	-0.9033	-1.995
2,444	2024-04-08 00:00:00	148.2798	136.4018	164.4628	136.7218	159.3055	1.9948	1.9948	1.9948	0.931
2,445	2024-04-09 00:00:00	148.2754	136.1196	164.9799	136.6703	159.3754	1.8646	1.8646	1.8646	0.838

Fig. Prediction Values



Select Stock

AAPL

Start date

2021/05/11

End date

2023/04/11

Start date: 2021-05-11

End date:2023-04-11

Fig. Prediction Graph

4.12.2. Working

- First The python modules required for running the project i.e., streamlit 1.20.0, ta 0.10.2, yfinance 0.2.14 are to be downloaded.
- Typing "`streamlit run FinalStreamlit.py`" on command prompt opens new the application page in the browser.
- However, in a passive mode, to manually open the page in the browser by the provided output link (i.e., the common dashboard mentioned in the next section) "`streamlit run FinalStreamlit.py –- server.headless true`" will be used.

4.13. Evaluations for Streamlit and Flask Web App

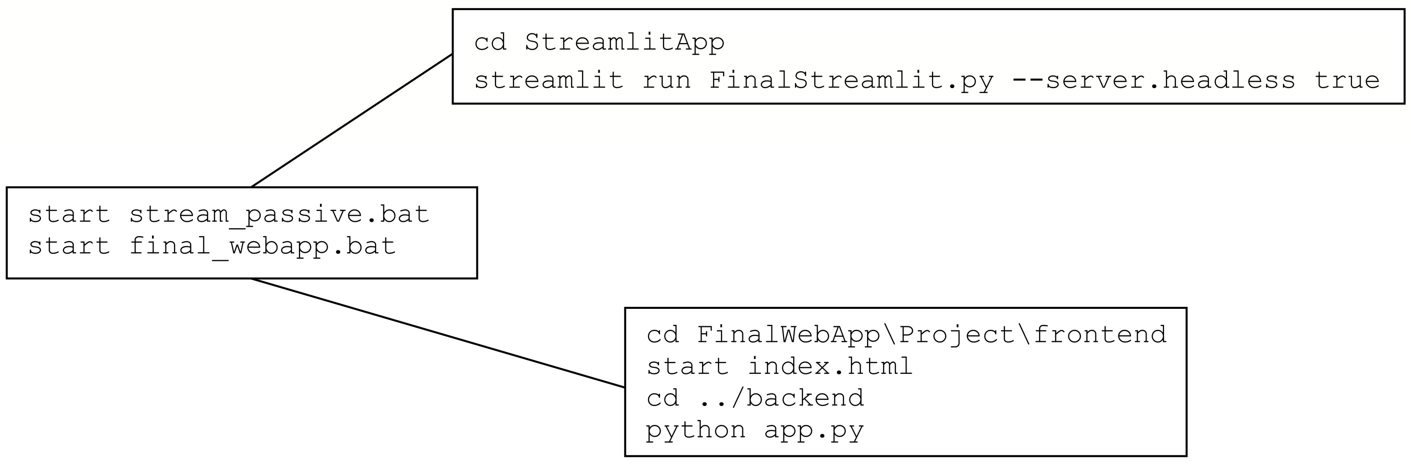
Flask very conveniently scales, can be customized to your liking and is thoroughly tested. Additionally, there is a lot of support within the community. On the downside, it can be overwhelming to begin with and the user needs to have knowledge about frontend development as Flask only provides backend support. They are easy to deploy, update, and maintain.

On the other hand, for Streamlit, there is no reason to be concerned with front-end development, it is easier to grasp and takes less time between the development and deployment stage. But it has its own drawbacks, there is no scaling, it is not yet fully developed and lacks many features of Flask as it is in beta. Additionally, it does not have the vast community and support ecosystem enjoyed by Flask. It's targeted for Machine Learning scientists to quickly deploy a machine learning model. One may not necessarily specialize in HTML, CSS, or Javascript, or be proficient in deploying in Flask, Django, or other tools. Streamlit enables data scientists to spend time on modeling.

For relatively simple apps Streamlit would suffice. But if the user requires a more secure full-fledged app then Flask would be the better option. Similarly, Streamlit can be used to develop web application when security is not needed. If you need security for your Web application, use Flask or Django packages. But Streamlit gives source code protection which web application usually doesn't.

4.14. Merging Streamlit and Flask Web App

A Common dashboard is created to access the web application and streamlit application, which are launched via a batch file which initiates the respective batch files of each application simultaneously.



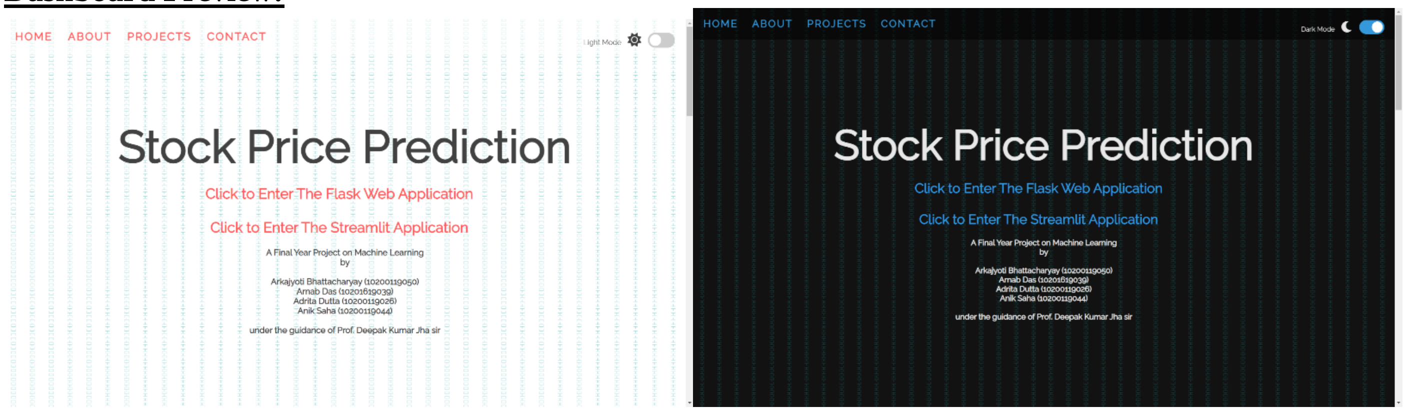
```
cd StreamlitApp
streamlit run FinalStreamlit.py --server.headless true
```

```
start stream_passive.bat
start final_webapp.bat
```

```
cd FinalWebApp\Project\frontend
start index.html
cd ../backend
python app.py
```

N.B.: Here "index.html" acts as the common dashboard.

<u>Dashboard Preview:</u>



HOME ABOUT PROJECTS CONTACT Light Mode

Stock Price Prediction

Click to Enter The Flask Web Application

Click to Enter The Streamlit Application

A Final Year Project on Machine Learning
by

Arkajyoti Bhattacharyay (10200119050)
Arnab Das (10201619039)
Adrita Dutta (10200119026)
Anik Saha (10200119044)

under the guidance of Prof. Deepak Kumar Jha sir

HOME ABOUT PROJECTS CONTACT Dark Mode

Stock Price Prediction

Click to Enter The Flask Web Application

Click to Enter The Streamlit Application

A Final Year Project on Machine Learning
by

Arkajyoti Bhattacharyay (10200119050)
Arnab Das (10201619039)
Adrita Dutta (10200119026)
Anik Saha (10200119044)

under the guidance of Prof. Deepak Kumar Jha sir

Fig.: Home Tab (Light mode & Dark mode)

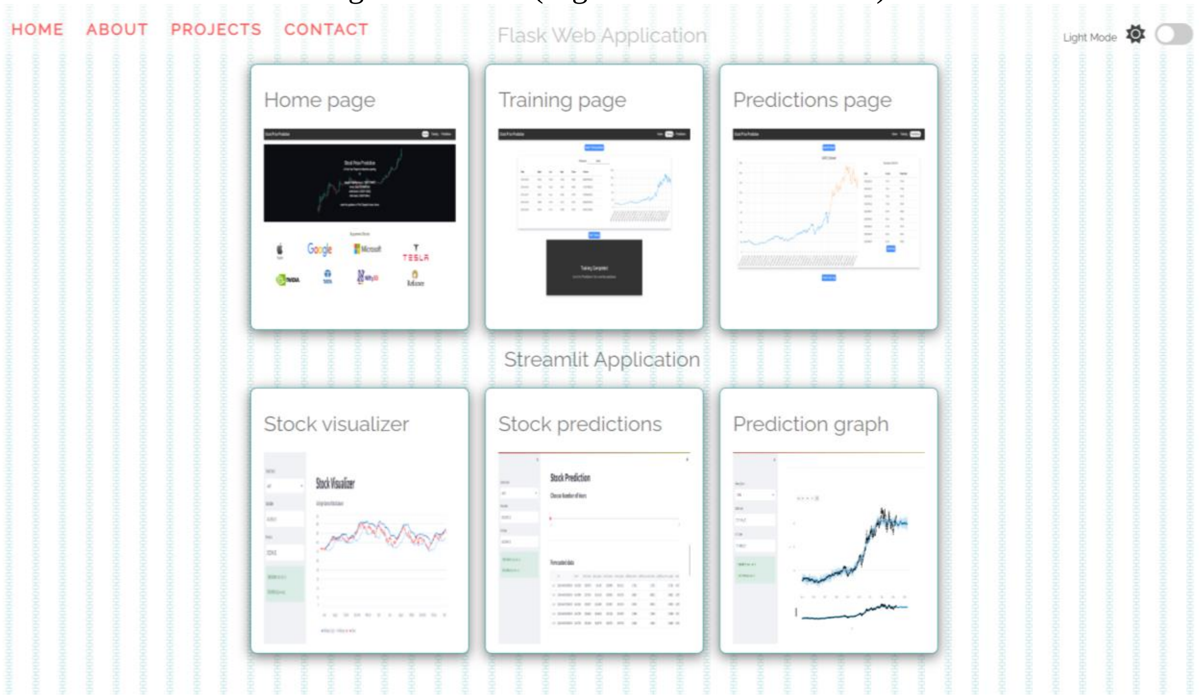
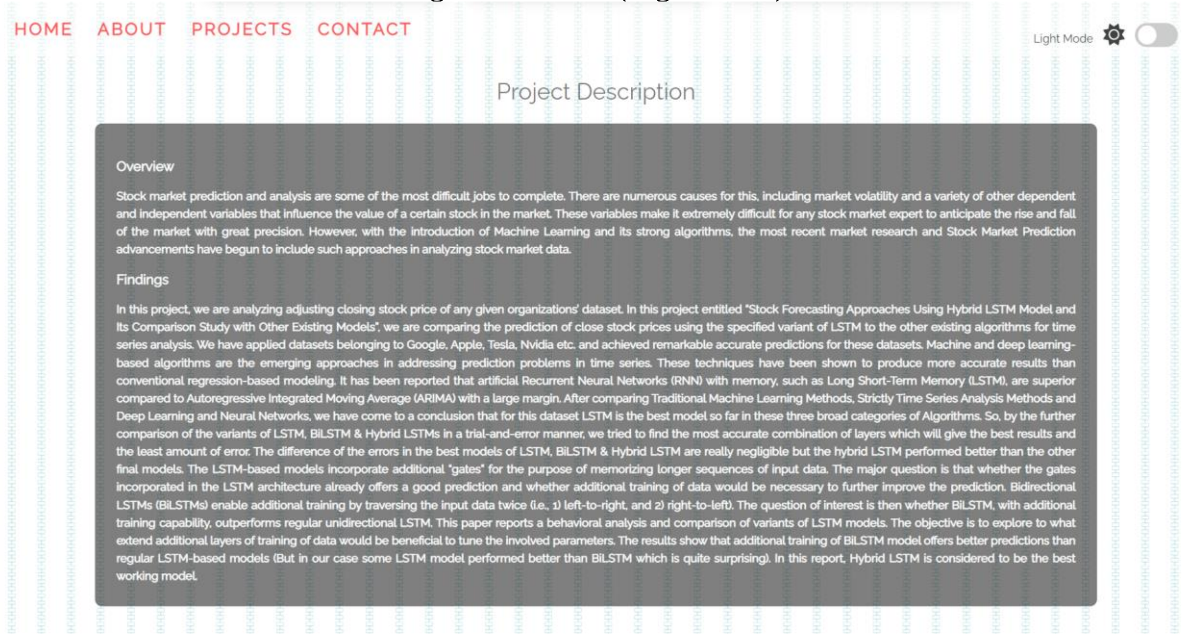


Fig.: About Tab (Light mode)



HOME ABOUT PROJECTS CONTACT Light Mode

Project Description

Overview

Stock market prediction and analysis are some of the most difficult jobs to complete. There are numerous causes for this, including market volatility and a variety of other dependent and independent variables that influence the value of a certain stock in the market. These variables make it extremely difficult for any stock market expert to anticipate the rise and fall of the market with great precision. However, with the introduction of Machine Learning and its strong algorithms, the most recent market research and Stock Market Prediction advancements have begun to include such approaches in analyzing stock market data.

Findings

In this project, we are analyzing adjusting closing stock price of any given organizations' dataset. In this project entitled "Stock Forecasting Approaches Using Hybrid LSTM Model and Its Comparison Study with Other Existing Models", we are comparing the prediction of close stock prices using the specified variant of LSTM to the other existing algorithms for time series analysis. We have applied datasets belonging to Google, Apple, Tesla, Nvidia etc. and achieved remarkable accurate predictions for these datasets. Machine and deep learning-based algorithms are the emerging approaches in addressing prediction problems in time series. These techniques have been shown to produce more accurate results than conventional regression-based modeling. It has been reported that artificial Recurrent Neural Networks (RNN) with memory, such as Long Short-Term Memory (LSTM), are superior compared to Autoregressive Integrated Moving Average (ARIMA) with a large margin. After comparing Traditional Machine Learning Methods, Strictly Time Series Analysis Methods and Deep Learning and Neural Networks, we have come to a conclusion that for this dataset LSTM is the best model so far in these three broad categories of Algorithms. So, by the further comparison of the variants of LSTM, BiLSTM & Hybrid LSTMs in a trial-and-error manner, we tried to find the most accurate combination of layers which will give the best results and the least amount of error. The difference of the errors in the best models of LSTM, BiLSTM & Hybrid LSTM are really negligible but the hybrid LSTM performed better than the other final models. The LSTM-based models incorporate additional "gates" for the purpose of memorizing longer sequences of input data. The major question is that whether the gates incorporated in the LSTM architecture already offers a good prediction and whether additional training of data would be necessary to further improve the prediction. Bidirectional LSTMs (BiLSTMs) enable additional training by traversing the input data twice (i.e., 1) left-to-right, and 2) right-to-left). The question of interest is then whether BiLSTM, with additional training capability, outperforms regular unidirectional LSTM. This paper reports a behavioral analysis and comparison of variants of LSTM models. The objective is to explore to what extend additional layers of training of data would be beneficial to tune the involved parameters. The results show that additional training of BiLSTM model offers better predictions than regular LSTM-based models (But in our case some LSTM model performed better than BiLSTM which is quite surprising). In this report, Hybrid LSTM is considered to be the best working model.

Fig.: Project Tab (Light mode)

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this project, we are analyzing adjusting closing stock price of any given organizations' dataset. In this project entitled "Stock Forecasting Approaches Using Hybrid LSTM Model and Its Comparison Study with Other Existing Models", we are comparing the prediction of close stock prices using the specified variant of LSTM to the other existing algorithms for time series analysis. We have applied datasets belonging to Google, Apple, Tesla, Nvidia etc. and achieved remarkable accurate predictions for these datasets.

Machine and deep learning-based algorithms are the emerging approaches in addressing prediction problems in time series. These techniques have been shown to produce more accurate results than conventional regression-based modeling. It has been reported that artificial Recurrent Neural Networks (RNN) with memory, such as Long Short-Term Memory (LSTM), are superior compared to Autoregressive Integrated Moving Average (ARIMA) with a large margin. After comparing Traditional Machine Learning Methods, Strictly Time Series Analysis Methods and Deep Learning and Neural Networks, we have come to a conclusion that for this dataset LSTM is the best model so far in these three broad categories of Algorithms.

So, by the further comparison of the variants of LSTM, BiLSTM & Hybrid LSTMs in a trial-and-error manner, we tried to find the most accurate combination of layers which will give the best results and the least amount of error. The difference of the errors in the best models of LSTM, BiLSTM & Hybrid LSTM are really negligible but the hybrid LSTM performed better than the other final models.

The LSTM-based models incorporate additional "gates" for the purpose of memorizing longer sequences of input data. The major question is that whether the gates incorporated in the LSTM architecture already offers a good prediction and whether additional training of data would be necessary to further improve the prediction. Bidirectional LSTMs (BiLSTMs) enable additional training by traversing the input data twice (i.e., 1) left-to-right, and 2) right-to-left). The question of interest is then whether BiLSTM, with additional training capability, outperforms regular unidirectional LSTM. This paper reports a behavioral analysis and comparison of variants of LSTM models. The objective is to explore to what extend additional layers of training of data would be beneficial to tune the involved parameters.

The results show that additional training of BiLSTM model offers better predictions than regular LSTM-based models (But in our case some LSTM model performed better than BiLSTM which is quite surprising). In this report, Hybrid LSTM is considered to be the best working model.

5.2 Future work

- Hosting the web application via hosting sites to make it robust.
- Upgrading the web application to understand the forecasting results efficiently.
- Comparing various algorithms and datasets with the existing model.
- Extending the model and application for predicting cryptocurrency trading.
- Adding sentiment analysis for better forecasting.

REFERENCES

[1] Shah, D., Isah, H. and Zulkernine, F., 2019. Stock market analysis: A review and taxonomy of prediction techniques. International Journal of Financial Studies, 7(2), p.26.

[2] Bustos, O. and Pomares-Quimbaya, A., 2020. Stock market movement forecast: A Systematic Review. Expert Systems with Applications, 156, p.113464.

[3] Jose, J., Mana, S. and Samhitha, B.K., 2019. An efficient system to predict and analyze stock data using Hadoop techniques. International Journal of Recent Technology and Engineering (IJRTE), 8(2), pp.2277-3878.

[4] Hu, Z., Zhao, Y. and Khushi, M., 2021. A survey of forex and stock price prediction using deep learning. Applied System Innovation, 4(1), p.9.

[5] Obthong, M., Tantisantiwong, N., Jeamwatthanachai, W. and Wills, G., 2020. A survey on machine learning for stock price prediction: algorithms and techniques.

[6] Yadav, A. and Vishwakarma, D.K., 2020. Sentiment analysis using deep learning architectures: a review. Artificial Intelligence Review, 53(6), pp.4335-4385.

[7] Sulandari, W., Suhartono, Subanar and Rodrigues, P.C., 2021. Exponential Smoothing on Modeling and Forecasting Multiple Seasonal Time Series: An Overview. Fluctuation and Noise Letters, p.2130003.

[8] Kumar, I., Dogra, K., Utreja, C. and Yadav, P., 2018, April. A comparative study of supervised machine learning algorithms for stock market trend prediction. In 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT) (pp. 1003-1007). IEEE.

[9] Ingle, V. and Deshmukh, S., 2016, August. Hidden Markov model implementation for prediction of stock prices with TF-IDF features. In Proceedings of the International Conference on Advances in Information Communication Technology & Computing (pp. 1-6).

[10] Singh, Sukhman, Tarun Kumar Madan, J. Kumar and A. Singh. "Stock Market Forecasting using Machine Learning: Today and Tomorrow." 2019 2nd International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT) 1 (2019): 738-745.

[11] Pahwa, K. and Agarwal, N., 2019, February. Stock market analysis using supervised machine learning. In 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon) (pp. 197-200). IEEE.

[12] Misra, Meghna, Ajay Prakash Yadav and Harkiran Kaur. "Stock Market Prediction using Machine Learning Algorithms: A Classification Study." 2018 International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering (ICRIEECE) (2018): 2475-2478.

[13] Vats, P. and Samdani, K., 2019, March. Study on Machine Learning Techniques In Financial Markets. In 2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN) (pp. 1-5). IEEE.

[14] Song, Y. and Lee, J., 2019, December. Design of stock price prediction model with various configurations of input features. In Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing (pp. 1-5).

[15] Werawithayaset, P. and Tritilanunt, S., 2019, November. Stock Closing Price Prediction Using Machine Learning. In 2019 17th International Conference on ICT and Knowledge Engineering (ICT&KE) (pp. 1-8). IEEE.

[16] Xingzhou, L., Hong, R. and Yujun, Z., 2019, July. Predictive Modeling of Stock Indexes Using Machine Learning and Information Theory. In Proceedings of the 2019 10th International Conference on E-business, Management and Economics (pp. 175-179).

[17] S. Sarode, H. G. Tolani, P. Kak and C. S. Life, "Stock Price Prediction Using Machine Learning Techniques," 2019 International Conference on Intelligent Sustainable Systems (ICISS), Palladam, India, 2019, pp. 177-181.

[18] Xiongwen Pang, Yanqiang Zhou, Pan Wang, Weiwei Lin, "An innovative neural network approach for stock market prediction", 2018

[19] Ishita Parmar, Navanshu Agarwal, Sheirsh Saxena, Ridam Arora, Shikhin Gupta, Himanshu Dhiman, Lokesh Chouhan Department of Computer Science and Engineering National Institute of Technology, Hamirpur – 177005, INDIA - Stock Market Prediction Using Machine Learning.

[20] Armano, G., Marchesi, M., and Murru, A. A. (2005) “Hybrid genetic-neural architecture for stock indexes forecasting.” Information Sciences 170 (2): 3–33.

[21] Hall, J.W. (1994) “Adaptive selection of US stocks with neural nets.” G.J. Deboeck (Ed.), Trading On the Edge: Neural, Genetic, and Fuzzy Systems for Chaotic Financial Markets, Wiley, New York: 45–65.

[22] Yaser, S.A.M. and Atiya, A.F. (1996) “Introduction to financial forecasting.” Applied Intelligence 6: 205–213.

[23] Kimoto, T., Asakawa, K., Yoda, M. and Takeoka, M. (1996) “Stock market prediction system with modular neural networks.” R.R. Trippi, E. Turban (Eds.), Neural Networks Finance Investing: Using Artificial Intelligence to Improve Real-World Performance, Irwin Publ., Chicago: 497–510.

[24] Moody, J. and Utans, J. (1994) “Architecture selection strategies for neural networks: application to corporate bond rating prediction.” A.P. Refenes (Ed.), Neural Networks in the Capital Markets, Wiley, New York: 277–300.

[25] Zhang, G.Q. and Michael, Y.H. (1998) “Neural network forecasting of the British Pound-US Dollar Exchange Rate.” Omega 26 (4): 495–506.

[26] Haykin, S. (1999) “Neural Networks: A Comprehensive Foundation.” Prentice-Hall International Inc., Englewood CliIs, NJ

[27] Chiang, W.C., Urban, T.L. and Baildridge, G. (1996) “A neural network approach to mutual fund net asset value forecasting.” Omega 24 (2): 205–215.

[28] Wen, Q., Yang, Z., Song, Y., and Jia, P. (2010) “Automatic stock decision support system based on box theory and svm algorithm.” Expert Systems with Applications 37: 1015–1022.

[29] Lin, Y., Guo, H., and Hu, J. (2013) “An svm-based approach for stock market trend prediction.” In Neural Networks (IJCNN), The 2013 International Joint Conference. IEEE: 1–7.

[30] Yu, H., Chen, R., and Zhang, G. (2014) “A svm stock selection model within pca.” Procedia computer science 31: 406–412.

[31] Gong, X., Si, Y.-W., Fong, S., and Biuk-Aghai, R. P. (2016) “Financial time series pattern matching with extended UCR suite and support vector machine.” Expert Systems with Applications 55: 284–296.

[32] Scholkopf, B., Smola, A., and Muller, K.R. (1998) “Nonlinear component analysis as a Kernel eigenvalue problem.” Neural Comput. 10: 1299–1319.

[33] Alcala, C.F. and Qin, S.J. (2010) “Reconstruction-based contribution for process monitoring with kernel principal component analysis.” Ind. Eng. Chem. Res. 49: 7849–7857.

[34] Lee, J.M., Yoo, C.K., Choi, S.W., Vanrolleghem, P.A. and Lee, I.B. (2004) “Nonlinear process monitoring using kernel principal component analysis.” Chem. Eng. Sci. 59: 223–234.

[35] Cheng, C.Y., Hsu, C.C. and Chen, M.C. (2010) “Adaptive kernel principal component analysis (KPCA) for monitoring small disturbances of nonlinear processes.” Ind. Eng.Chem. Res. 49: 2254-2262.

[36] Thomason, M. (1999) “The practitioner methods and tool.” Journal of Computational Intelligence in Finance 7 (3): 36–45.

[37] Diler, A. I. (2003) “Predicting direction of ISE national-100 index with back propagation trained neural network.” Journal of Istanbul Stock Exchange 7 (25-26): 65–81.

[38] Huang, C. L., and Tsai, C. Y. (2009) “A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting.” Expert Systems with Applications 36 (2): 1529–1539.

[39] Kim, K. (2003) “Financial time series forecasting using support vector machines.” Neurocomputing 55: 307–319.

[40] Juszczak, P., Tax, D. M. J. and Duin, R. P. W. (2002) “Feature scaling in support vector data descriptions.” Proc. 8th Annu. Conf. Adv. School Comput. Imaging.

[41] Smola, A.J. (1998) “Learning with Kernels.” Ph.D. Thesis, GMD, Birlinghoven, Germany.

[42] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Comput, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.