# Pattern Recognition in Financial Time Series Data

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

by

Aryan Bansal (2020EEB1162), Yash Varshney (2020EEB1222)

Akshit Barnwal (2020EEB1150), Sparsh Verma (2020EEB1210)

Keshav Arora (2020EEB1177)

Under the guidance of

**Dr. Ashwani Sharma**

and

**Dr. Arun Kumar**


DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY ROPAR

2024

# Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 15 November 2023

# Abstract

Stock markets are so non-linear, that traders find it extremely difficult to make investment decisions. As a result, they are always searching for ways to forecast stock market trends ahead of time. Deep Neural networks can be used to solve this issue. In this project, predictions are made using the Long Short Term Memory (LSTM) Deep Neural Network architecture, which models sequential data. It also helps in learning long-term dependencies in complex data. The project critically analyzes and predicts the daily data for two major NSE indices, NIFTY 50 and NIFTY BANK. We have created Single-layer and Multilayer LSTM, which are evaluated based on metrics like RMSE and MAPE, and to further enhance the results, we have added different regularization techniques like dropout, early stopping, etc. The project suggests that the models made with LSTM are far more accurate than the decision tree and random forests.

# Contents

# List of Figures

# Chapter 1

# Introduction

The stock market is very unpredictable for many reasons, like unemployment rate changes, monetary policies, immigration policies, public conditions, natural calamities, economic stability, and many more. All stakeholders try to make higher profits by thorough market analysis. They want some platforms to analyze the data to reduce the associated risks. However, gathering such vast information and putting it together to construct a reliable model is challenging. Even if we do so, the accuracy matters very much in these scenarios, because many peoples hard earned money get affected by this.

Stocks are inherently noisy and non-deterministic. These are non-parametric, chaotic systems, making feature selection and prediction of future data challenging. Various studies have used technical indicators or historical data to train their models. However, the prediction of models is not competitive because of limited features. On the other hand, if we use more parameters, then the model becomes complex and difficult to interpret. Hence, choosing a good combination of features is necessary for the analysis.

There has been a lot of research in the financial industry till today. Many researchers used historical stock prices as a base to perform time series analysis. Many statistical models like Moving Average (MA), Weighted Moving Average, Auto Regression (AR), ARIMA, etc. and Non-linear models like GARCH were tried. Still, these are not entirely accurate and are not widely used.

But with the advancements in the technology field of data analysis. In the eighties, researchers started building models by harnessing the power of ANN (Artificial Neural Network) and DL (Deep Learning Techniques) for temporaly related data processing and prediction. These models are specialized to extract complex relationships relative to traditional models.

Many deep-learning models are developed to solve the issue. But In a Feedforward Neural Network, the information flows in forward direction only. Since each step is independent of the other and processed separately, it does not retain information from the previous step. Thus, these models fail because, for a stock market, we need a model that predicts the new value based on a series of prior events.

Therefore, RNNs (Recurrent Neural Networks) are modeled to help in these situations. The RNN has loops, which allow the model to store relevant information that could persist over time. However, this technique faces the problem of vanishing Gradients. This happens because parameters are spread all over the networks, and finding gradients involves finding partial derivatives of all the variables.

In RNN, gradients tend to get very small, and hence, gradients could vanish or decay back through the networks. Therefore, LSTM was designed to solve this problem. It's a typical RNN, with an additional feature to selectively forget the non-essential data. Memorizing information for an extended period is the specialty of the LSTM model. LSTMs can filter and process information effectively, avoiding the vanishing gradient in deep neural networks. Thus allowing them to learn from prior data while reacting to changing market conditions, improving forecast accuracy.

Hence, in this study, we carefully chose appropriate data to construct the model. Following that, we normalized it using min-max normalization. Then, we generated the input sequence for LSTM for a given time step. Then, we tune the hyperparameters like epochs, number of neurons, batch size, etc. Also, we used regularization techniques to avoid overfitting. After modifying the hyperparameters, the input is given to the LSTM model, which predicts the closing price of stock. We are using RMSE, MAPE, and R to evaluate the results for our model.

# Chapter 2

# Motivation

The uncertain nonlinear nature of the stock market makes it very difficult for Shareholders to make decisions about their investments. The stock price data is affected by multiple micro and macroeconomic factors like GDP, Public Sentiment, Taxes, Company Profits etc.
To predict and recognize patterns, machine learning models must be developed that can model these non-linearities accurately. Traditional machine learning methods are not accurate since they cannot consider nonlinearities, therefore deep neural networks need to be used.
All these challenges prevailing in the market was the motivation behind the project "Pattern Recognition in Financial Time Series Data" which aimed at solving the following problems:

1. **Making better decisions in the Stock Market:** by using deep neural networks to make predictions in the stock market and finding hidden trends in the time series data, shareholders can make better decisions about their investments to gain maximum profits and avoid mistakes.

2. **Comparing and Analyzing different methods of Stock Data prediction:** various machine learning strategies have been used to make predictions in the stock market, they have their own strengths and weaknesses. During the initial stages of this project, an in depth analysis of all such techniques was made and latest research in the field was looked into. This comparative analysis helped in improving the understanding of various machine learning techniques and deep learning techniques to model stock market data.

3. **Catching Trends in the Stock Markets:** using LSTM to predict future trends in the stock market beforehand so that investors can be wary of upcoming trends and make investing decisions accordingly and have better opportunities to make profits.

# Chapter 3

# Objective

The objectives of the report include:

- Evaluating the performance of a single layer and multilayer LSTM as well as transformer model architectures for predicting the Nifty index closing price.
- Analyzing the input factors and their effect on the predicted accuracy of LSTM model.
- Analyzing the effects of hyperparameters on model performance for LSTM architecture. Examine the effects of the number of neurons, epochs, learning rate, batch size, and time step on the performance.
- Investigating the regularisation strategies that might improve the model resilience and reduce overfitting in LSTM model.
- Interpreting and analyzing the usefulness of LSTM models for stock market prediction.
- Examining both LSTM topologies, determining future research directions and possible improvements to increase the precision and dependability of predictive models in financial forecasting.

# Chapter 4

# Literature Review

With Time, financial institutions are embracing Artificial intelligence in various aspects like predicting the price of a stock, trading independently and many others. Machine learning is adopted at various degrees of sophistication in the finance industry. Predicting the stock price with various deep learning algorithms can help the Trader decide when to buy and sell stocks to make a profit. With good machine learning models, we can model the interdependencies between the history of stock price data.
There are various studies on this modelling of financial time series data, we will review the previous work.

The author [1] here aims to predict the future price of stock by training on LSTM, with the predicted price, they calculate the growth of the company which can happen in future. They calculate the future growth of various companies and then plot the curve of future growth of various companies. They find the similarity between the future growth of various companies. They introduce a approach to examine which period is most optimal to predict the stock price of the company. They predict the future closing price of 5 different companies with LSTM on different periods. Then from that they predict the best period for the companies by calculating the minimum error between all the predictions. They use only date and Close price for Prediction, which might fail in taking macroeconomic factors into account.
This work [2] presents a two-stage approach to stock market data forecasting that combines independent component analysis (ICA) with support vector regression (SVR). The motivation is that by first removing noisy elements from the data with ICA, the SVR model may accomplish greater forecasting performance than if the original noisy data were used. The technique

is tested on the Nikkei 225 and TAIEX stock index datasets and found to beat SVR without ICA filtering, as well as a random walk model benchmark. Working on financial time series presents significant challenges, including noise and the need for stationary data.

The author [3] uses long short-term memory (LSTM) networks, to predict the directional changes of S&P 500 constituent equities between 1992 and 2015.The LSTM model outperforms other conventional ML models, such as random forest models, deep neural networks, and others , in terms of predicted precision and profitability. Before transaction charges, the LSTM generated 0.46% daily returns and had a Sharpe ratio of 5.8. Relative to the benchmark models, LSTM returns were less susceptible to prevalent forms of systematic risk

This research [4] uses deep learning and machine learning approaches to provide an interdisciplinary model for predicting market prices. They plan to forecast the NIFTY 50's price from December 29th, 2014 to July 31, 2020.They used training data to develop 8 regression models that forecasted NIFTY 50 open values during the test period. The authors develop four deep learning-centered models for regression that utilize Long Short-Term Memory (LSTM) networks for improved predictive power. These LSTM models have varying topologies and input structures. The authors believe that combining machine learning with deep learning models, specifically the LSTM-based method, can accurately anticipate future stock index movements.

According to efficient market theory [5], when all information about an organization and stock market events is readily available to stakeholders and investors, the impact of those events is already reflected in the stock price. The historical spot price is believed to be the most accurate predictor of future market movement, taking into account all other factors. To predict future stock price trends, we use Machine Learning (ML) methods on historical data along with other macroeconomic factors and technical indicators , taking into account all relevant elements. ML approaches can reveal previously unknown patterns and insights, leading to highly accurate forecasts.

# Chapter 5

# Project Roadmap

The Project Roadmap is a section where we will outline all the steps that we took while completing the Capstone Project. This section will be a detailed description of everything that was done to accomplish the goal of this project, along with the detailed description of everything we will also provide all the resources that we used while building the project and how we got hold of these resources.


Figure 5.1: Project Roadmap

## 5.1 Research and Discovery Phase

During the first step of the project "Pattern Recognition in Financial Time Series Data" a literature review of different ML and Deep Learning techniques currently being employed to model stock market data was conducted. It was found out that stock market data was sensitive to micro and macro-economic factors which made it highly non-linear and unpredictable at times. Therefore, it is futile to use basic machine learning techniques like regression techniques as they fail in the task of modelling non-linearities.

It was found that using modern Deep Learning techniques is suitable in this case because it can consider numerous parameters at once and can model non-linearities better than traditional machine learning models.

After this, the problem of time series data was tackled. Out of many approaches, that are mentioned in the literature review in the previous sections, LSTM which stands for Long Short Term Memory was finalized and it was found to be the best method for this task.

## 5.2 Obtaining Stock Market data

We have obtained the data for Nifty and Banknifty with the help of KiteConnect (a Python-based API) from Zerodha.
We have created a trading account with Zerodha. Then, using the credentials, we touched upon various API endpoints of the Zerodha web terminal and fetched the data.
We downloaded the day-by-day data for two indices:

- NIFTY 50
- NIFTY BANK

We also downloaded the data for India VIX (Volatility Index)

## 5.3 LSTM (Long Short Term Memory) Neural Network

After comparing various machine learning techniques, LSTM was finalized at the end because of the following benefits: LSTM neural networks have a memory that can model time series data in the best possible way. Second, instead of just forgetting the previous entries in the data, in other words, instead of reducing the significance of older entries in the time series, the LSTM model "learns to forget".
"Learns to forget" means that whenever a trend is noticed in the modelled data in such a way that current entries affect later entries greatly, It places more weight on these trends in the future while making predictions.

## 5.4 Building and Training of the Neural Network

After obtaining the data and finalizing the model the neural network was built and trained on the data to make predictions.
The coding was done in a Python library known as Tensorflow, Tensorflow is a highly versatile

modern deep learning library widely used across the world for building and training highly performant deep learning models.

## 5.5 Adding Regularization and Optimization Techniques for better training

After training the initial neural network, various Regularization and Optimization techniques were used to improve the weights of the model and improve the model accuracy on real-world data.

Regularization techniques sometimes reduce model accuracy on the training data but in turn, improve the model accuracy on real-world data by improving the generalizability of the model by preventing overfitting and better managing the bias-variance tradeoff.

Optimization of the model involves improving the efficiency, size and speed of the model without drastically sacrificing the model performance on the training data.

## 5.6 Making Final Predictions and Visualizing the Results

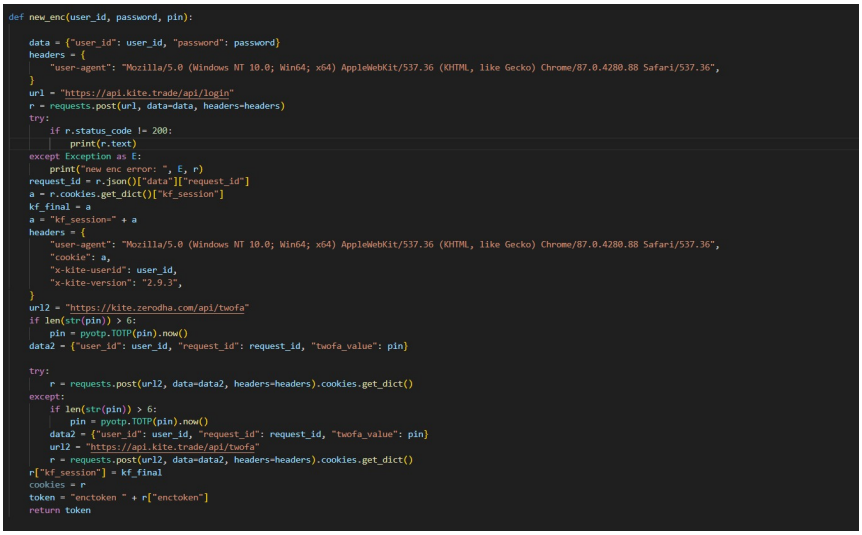
Single and Multilayer LSTM and Decision Trees models were trained. The trained models were used for making predictions. For visualizing the results that were found out, the data visualization library Matplotlib was utilized which is a low level python library for data visualization. The results that were found out were displayed in Linear charts and Heat maps for better understanding and comparing the predicted values with the original data.

# Chapter 6

# Code and Architecture

## 6.1 Data Fetching and Formating

We created a function new_enc() for receiving the encryption token from Zerodha Web Terminal. The following image depicts the whole code.



```
def new_enc(user_id, password, pin):

    data = {"user_id": user_id, "password": password}
    headers = {
        "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36",
    }
    url = "https://api.kite.trade/api/login"
    r = requests.post(url, data=data, headers=headers)
    try:
        if r.status_code != 200:
            print(r.text)
    except Exception as E:
        print("new enc error: ", E, r)
    request_id = r.json()["data"]["request_id"]
    a = r.cookies.get_dict()["kf_session"]
    kf_final = a
    a = "kf_session=" + a
    headers = {
        "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/87.0.4280.88 Safari/537.36",
        "cookie": a,
        "x-kite-userid": user_id,
        "x-kite-version": "2.9.3",
    }
    url2 = "https://kite.zerodha.com/api/twofa"
    if len(str(pin)) > 6:
        pin = pyotp.TOTP(pin).now()
    data2 = {"user_id": user_id, "request_id": request_id, "twofa_value": pin}

    try:
        r = requests.post(url2, data=data2, headers=headers).cookies.get_dict()
    except:
        if len(str(pin)) > 6:
            pin = pyotp.TOTP(pin).now()
        data2 = {"user_id": user_id, "request_id": request_id, "twofa_value": pin}
        url2 = "https://api.kite.trade/api/twofa"
        r = requests.post(url2, data=data2, headers=headers).cookies.get_dict()
    r["kf_session"] = kf_final
    cookies = r
    token = "enctoken " + r["enctoken"]
    return token
```

Figure 6.1: Code snippet for fetching encryption token from Zerodha Web Terminal

We created another method, fetchOHLC(), for fetching the OHLC of all the mentioned indices, and ETFs. OHLC denoted Open, High, Low, and Close where Open means the open price, High means the high price, Low means low price, and Close means the close price of a particular stock during a particular timeframe. For us, that essentially describes the whole day movement of the stock. The starting date of the data obtained is 2010-01-04 to maintain consistency among all the data fetched.

```python
def fetchOHLC(code, interval, duration, user,exchange, date1=None, date2=None):
    if not date1:
        date1, date2 = (dt.date.today() - dt.timedelta(days=duration)).strftime("%Y-%m-%d"), (dt.date.today()).strftime(
            "%Y-%m-%d")
    # dat1, date2 = '2021-02-01'
    print(date1, date2)
    token = new_enc(user, password, pin)
    headers = {
        "authorization": token
    }
    r = requests.get('https://kite.zerodha.com/oms/instruments/historical/' + str(code) + '/' + str(
        interval) + '?user_id=EC4598&oi=24&from=' + str(date1) + '&to=' + str(date2), headers=headers).json()
    # print(r)
    d = r['data']['candles']
    df = pd.DataFrame(d)
    try:
        df.columns = ['Time', 'Open', 'High', 'Low', 'Close', 'Volume']
    except:
        df.columns = ['TIME', 'OPEN', 'HIGH', 'LOW', 'CLOSE', 'VOLUME', 't']
        pass
    return df
```

Figure 6.2: Code snippet for fetching OHLC data

### 6.1.1 Data description of various Indices

1. NIFTY 50 data:

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | Time | Open | High | Low | Close | Volume |
| 2 | 0 | 2010-01-04T00:00:00+0530 | 5200.89 | 5238.45 | 5167.1 | 5232.2 | 0 |
| 3 | 1 | 2010-01-05T00:00:00+0530 | 5277.15 | 5288.35 | 5242.39 | 5277.9 | 0 |
| 4 | 2 | 2010-01-06T00:00:00+0530 | 5278.15 | 5310.85 | 5260.05 | 5281.8 | 0 |
| 5 | 3 | 2010-01-07T00:00:00+0530 | 5281.8 | 5302.55 | 5244.75 | 5263.1 | 0 |
| 6 | 4 | 2010-01-08T00:00:00+0530 | 5264.25 | 5276.75 | 5234.7 | 5244.75 | 0 |
| 7 | 5 | 2010-01-11T00:00:00+0530 | 5263.8 | 5287.2 | 5227.8 | 5249.4 | 0 |
| 8 | 6 | 2010-01-12T00:00:00+0530 | 5251.1 | 5300.5 | 5200.95 | 5210.39 | 0 |
| 9 | 7 | 2010-01-13T00:00:00+0530 | 5212.6 | 5239.2 | 5169.55 | 5233.95 | 0 |
| 10 | 8 | 2010-01-14T00:00:00+0530 | 5234.5 | 5272.85 | 5232.5 | 5259.9 | 0 |
| 11 | 9 | 2010-01-15T00:00:00+0530 | 5259.9 | 5279.85 | 5242.45 | 5252.2 | 0 |
| 12 | 10 | 2010-01-18T00:00:00+0530 | 5253.65 | 5292.5 | 5228.95 | 5274.85 | 0 |
| 13 | 11 | 2010-01-19T00:00:00+0530 | 5274.2 | 5287.8 | 5218.64 | 5225.64 | 0 |

Figure 6.3: The data of stock index Nifty 50

Columns Description:

- Time: It denotes the date, that's what we need, as we have to train our model on daily data.
- Open, High, Low, Close: As the name suggests it represents the stock price during the whole day.
- Volume: This column is redundant and we will drop this as we don't need the information of Volume.

2. USD-INR and Treasury Bonds data : USD-INR conversion rates denote the conversion value of Indian Rupees against the US Dollar. We have taken the data from investing.com(cite the website), which is a commonly available data. Treasury Bonds data

has been taken from investing.com, the treasury bonds are considered the safest forms of investment because they are backed by the government. They are theoretically a risk-free investment. We have taken 1-year, 10-year, and 30-year bond data.

3. MACD, RSI and ATR: MACD represents the exponential moving average, which helps us confirm trends and make bullish/bearish signals.ATR stands for Average True Range, and it helps us determine the market volatility. It helps us determine how much the stock price will move in a given period.RSI stands for Relative Strength Index, which helps us determine whether the underlying stock is overbought or underbought.MACD, ATR, and RSI are calculated by their mathematical formulas using the closing price of the stock

```
# Define window sizes for calculations
macd_fast_ema = 12
macd_slow_ema = 26
macd_signal_ema = 9
atr_period = 14
rsi_period = 14

# Calculate EMA (Exponential Moving Average)
ema = prices.ewm(alpha=1/macd_fast_ema, min_periods=macd_fast_ema).mean()

# Calculate MACD (Moving Average Convergence Divergence)
macd = ema - prices.ewm(alpha=1/macd_slow_ema, min_periods=macd_slow_ema).mean()
signal = macd.ewm(alpha=1/macd_signal_ema, min_periods=macd_signal_ema).mean()
macd_hist = macd - signal

# Calculate ATR (Average True Range)
true_range = pd.concat([prices.diff(), abs(prices - ema.shift(1)), abs(ema - ema.shift(1))], axis=1).max(axis=1)
atr = true_range.ewm(alpha=1/atr_period, min_periods=atr_period).mean()

# Calculate RSI (Relative Strength Index)
delta = prices.diff()
up, down = delta[delta > 0], delta[delta < 0]
avg_gain = up.ewm(alpha=1/rsi_period, min_periods=rsi_period).mean()
avg_loss = abs(down).ewm(alpha=1/rsi_period, min_periods=rsi_period).mean()
rs = avg_gain / avg_loss
rsi = 100 - 100 / (1 + rs)
```

Figure 6.4: Python code for calculating RSI, MACD and ATR

## 6.2 Predicting Indices prices using LSTM

### 6.2.1 Supporting Functions

We started our research by defining supporting functions for evaluating our machine-learning model. The functions are as follows :

- Error Calculations: We made functions to calculate the mean absolute percentage error, Root Mean Squared Error and R between actual and prediction values y_true and y_pred.

- DataSet Creation: This function helps us to create a sequence of data points from the given input dataset using time step value .
- Data set split : This method will split the input dataset into training and testing datasets.
- Min Max and Inverse Min-max Transformation: This function will normalize the data using the MinMaxScaler class of sci-kit-learn and re scale the converted values using inverse min-max transformation .

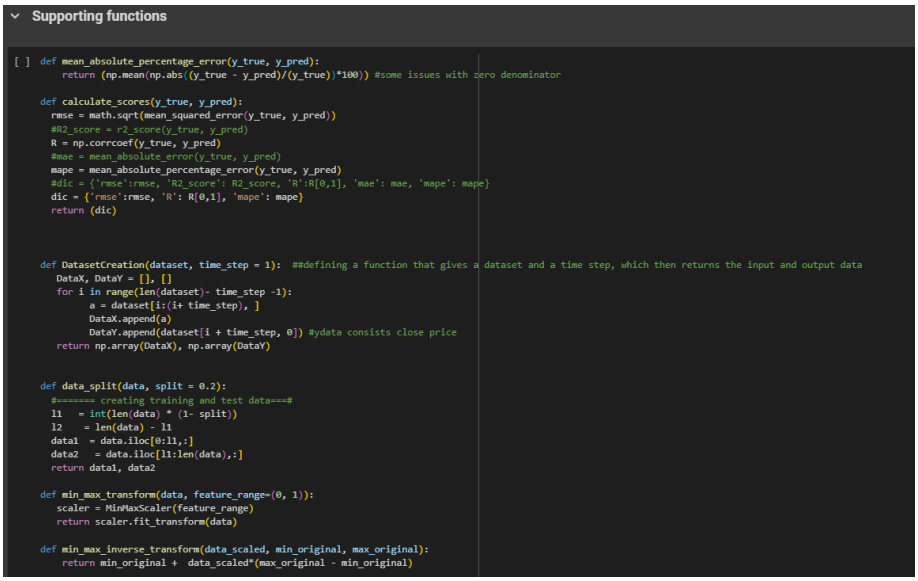



Figure 6.5: code Snippet for Supporting functions

### 6.2.2 Function For Building Model

The algorithm starts by preparing the input data, which includes the number of observations, time step, and features, as well as given hyperparameters including the optimizer, learning rate, and batch size. During initialization, epochs are set to a big value and tolerance to 5. The loop that tunes the model repeatedly tries various configurations of neurons and layers. Within this loop, the model gets trained for each range of replicate numbers, and training loss is tracked until convergence or the maximum number of epochs is achieved. The mean absolute percentage error (MAPE), the minimum, maximum, and average values of the coefficient of determination (R), and the root mean square error (RMSE) are computed after the model has been trained. Key findings are then kept for future analysis or presentation.

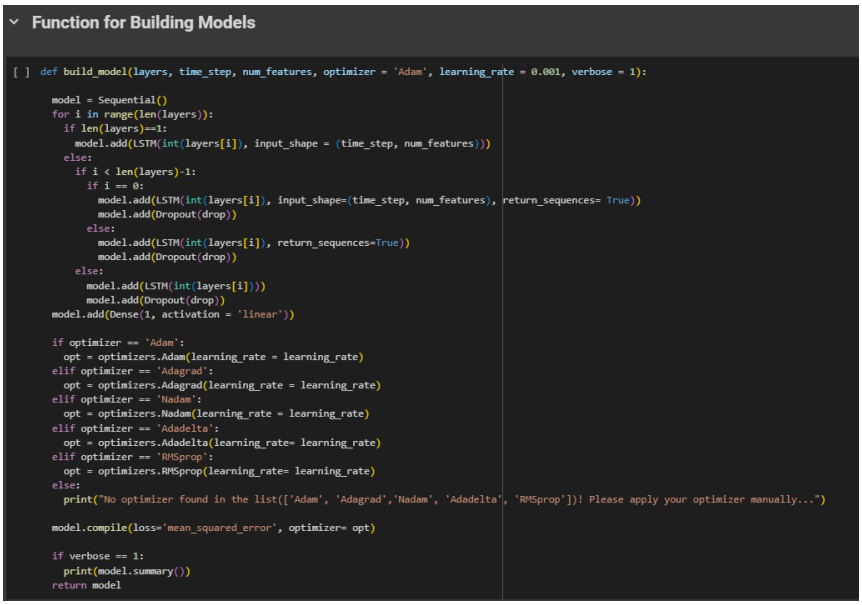



Figure 6.6: Function For Building Model

### 6.2.3 HyperParameter Tuning

Various hyperparameters such as number of neurons, epochs, time step, learning rate and optimizing function are incorporated in the model. To make the model robust and to reduce the generalization error, we incorporated functions for tuning the hyperparameters. Hyperparameters are tuned in the following way

Iteratively going through each optimizer option is how the optimization loop gets started. Iterates through various learning rates inside each optimizer. The different batch sizes are investigated for every learning rate. It trains the model by iteratively going through several replicates within each batch size and tracking validation loss until the maximum epochs are achieved or the convergence requirements are satisfied. Following this lengthy training phase, the algorithm evaluates the trained model using validation data and calculates root mean square error (RMSE) ratings. After that, these scores are totaled by figuring out the average RMSE for every combination of hyperparameters. The optimal collection of hyperparameters, their related average RMSEs, and the best average RMSE attained are finally produced by the method.

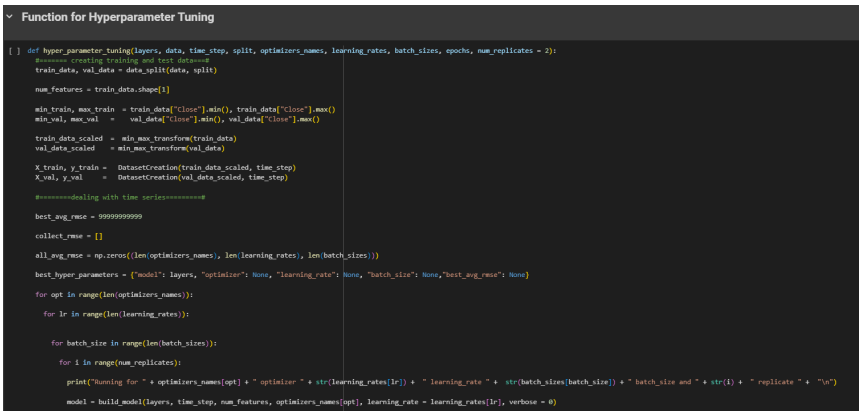



Figure 6.7: Code snippet of function tuning Hyperparameters

### 6.2.4 Training and Testing Single and Multi-Layered LSTM models

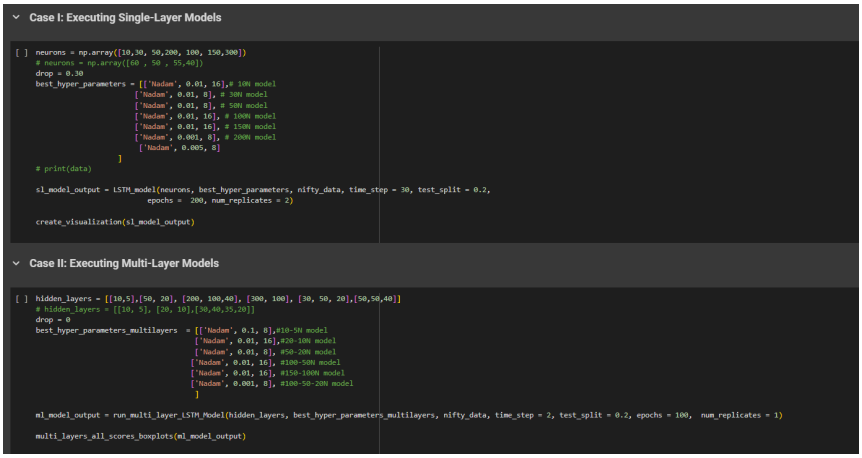



Figure 6.8: Training and Testing Single and Multi-Layered LSTM models

We trained Various LSTM models with different numbers of neurons and different depths based on the best hyperparameters for them which we got from hyperparameter tuning. Single and multilayer LSTM models were used to forecast closing prices. We compare the performance of two LSTM architectures using plots of performance metrics.

# Chapter 7

# Comparative Study Between Decision Tree and Random Forest Models

## 7.1 Decision Tree Analysis

1. NIFTY 50 data:

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | Time | Open | High | Low | Close | Volume |
| 2 | 0 | 2010-01-04T00:00:00+0530 | 5200.89 | 5238.45 | 5167.1 | 5232.2 | 0 |
| 3 | 1 | 2010-01-05T00:00:00+0530 | 5277.15 | 5288.35 | 5242.39 | 5277.9 | 0 |
| 4 | 2 | 2010-01-06T00:00:00+0530 | 5278.15 | 5310.85 | 5260.05 | 5281.8 | 0 |
| 5 | 3 | 2010-01-07T00:00:00+0530 | 5281.8 | 5302.55 | 5244.75 | 5263.1 | 0 |
| 6 | 4 | 2010-01-08T00:00:00+0530 | 5264.25 | 5276.75 | 5234.7 | 5244.75 | 0 |
| 7 | 5 | 2010-01-11T00:00:00+0530 | 5263.8 | 5287.2 | 5227.8 | 5249.4 | 0 |
| 8 | 6 | 2010-01-12T00:00:00+0530 | 5251.1 | 5300.5 | 5200.95 | 5210.39 | 0 |
| 9 | 7 | 2010-01-13T00:00:00+0530 | 5212.6 | 5239.2 | 5169.55 | 5233.95 | 0 |
| 10 | 8 | 2010-01-14T00:00:00+0530 | 5234.5 | 5272.85 | 5232.5 | 5259.9 | 0 |
| 11 | 9 | 2010-01-15T00:00:00+0530 | 5259.9 | 5279.85 | 5242.45 | 5252.2 | 0 |
| 12 | 10 | 2010-01-18T00:00:00+0530 | 5253.65 | 5292.5 | 5228.95 | 5274.85 | 0 |
| 13 | 11 | 2010-01-19T00:00:00+0530 | 5274.2 | 5287.8 | 5218.64 | 5225.64 | 0 |

Figure 7.1: The data of stock index Nifty 50

In analyzing Nifty 50 data, We're using a Decision Tree method to determine if the stock price will rise over its opening value. This strategy trains the model using historical Open, High, Low, Close, and Volume data. The goal is to analyze Decision Tree's capacity to detect patterns in data to understand market dynamics better and discover profitable trading opportunities based on expected price changes relative to the opening value.


Figure 7.2: This graph shows the Decision Tree Predicting if the Price will increase or not

We have obtained 61.06% accuracy for predicting the increased price Label. This suggests the moderate performance of the Tree Model. Also, it is not advisable for robust financial decision-making.

Then We analyzed the same data and tried to predict the closing price We considered the Opening, High and Low values for predicting the closing price.

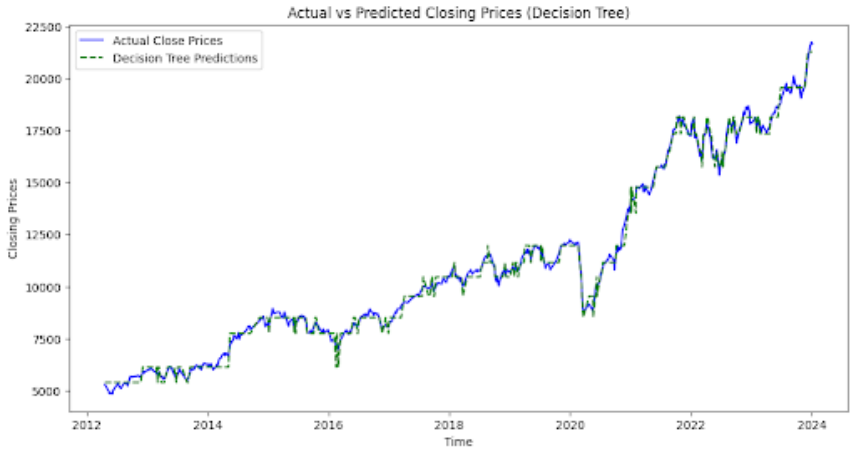



Figure 7.3: This graph shows Decision Tree Predicted Closing Price vs Actual Closing Price

This graph showed the Decision Tree is trying to cope with the noisy dynamics of the stock index, but it is lagging. We have obtained 277.045 RMSE, which shows how inefficient the Decision Tree Model is in predicting the closing price values.

2. NIFTY BANK data: For Nifty Bank data as well, we observed similar results using the Decision Tree.

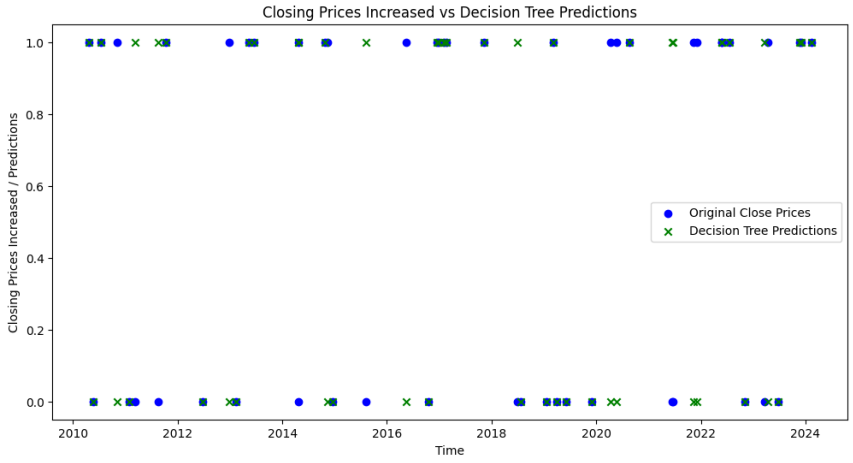



Figure 7.4: This graph showed the price increased prediction for Nifty Bank data

The accuracy observed in this case was 67.89%, which was better than Nifty 50 data, but still not good enough for its usage.

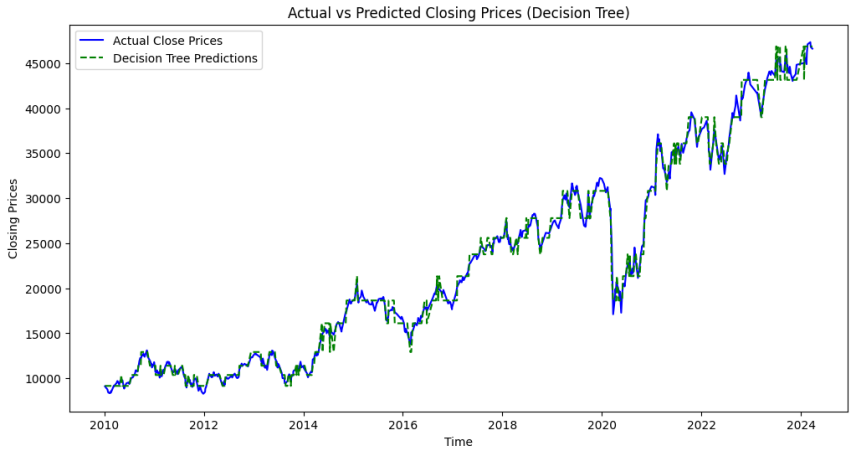



Figure 7.5: This graph showed the Actual vs Predicted Closing Prices for Nifty Bank Data

The RMSE observed in this case is 733.06, which is very large and even worse than the Nifty 50 data. Hence, we can conclude Decision Tree's Results were not optimal for predicting stock indices in any way.

## 7.2 Random Forest Analysis

1. NIFTY 50 data: Similar to Decision Tree, We're trying to use Random Forest to analyze its capacity to detect patterns in data to understand market dynamics better and discover profitable trading opportunities based on expected price changes relative to the opening value.

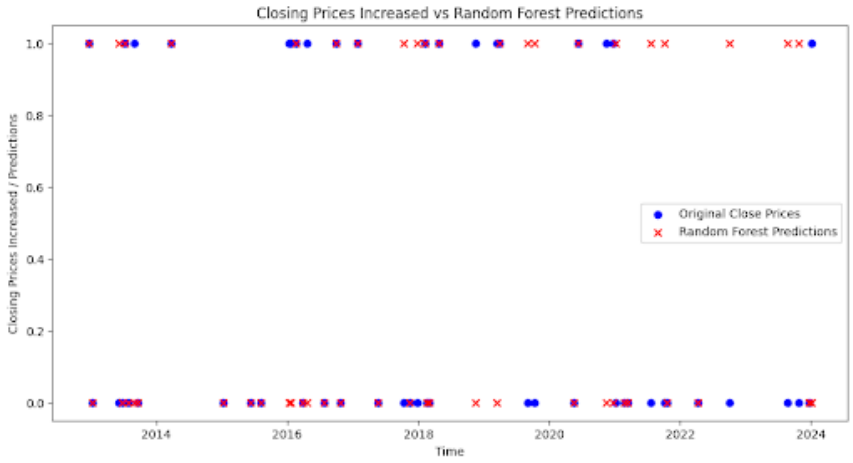



Figure 7.6: This graph shows the Random Forest Prediction if the closing price goes up compared to the opening price


Figure 7.7: This graph shows the Actual vs Random Forest Predicted values for closing prices

In this analysis, we have observed 63.98% accuracy for the former part, i.e., for predict-

ing the increased price label. For predicting the actual price, we have observed the RMSE equals 219.24, which is large and signals its inefficiency for usage in financial systems.

2. NIFTY Bank data:

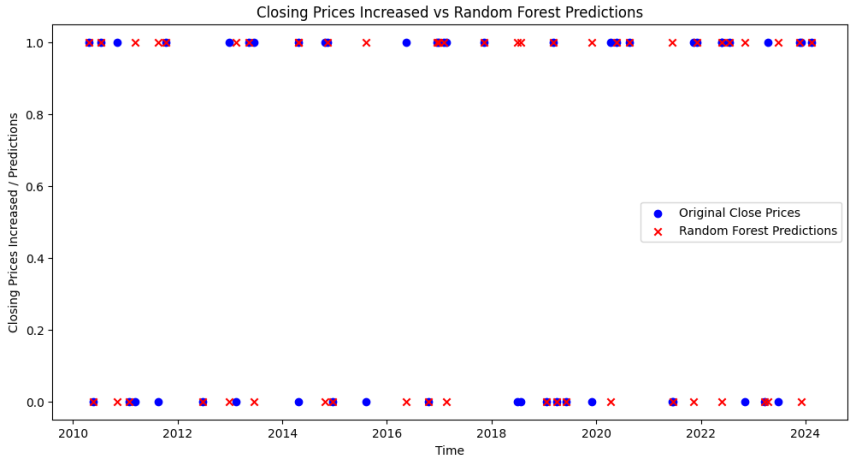



Figure 7.8: This graph shows Random Forest Predictions if price increases

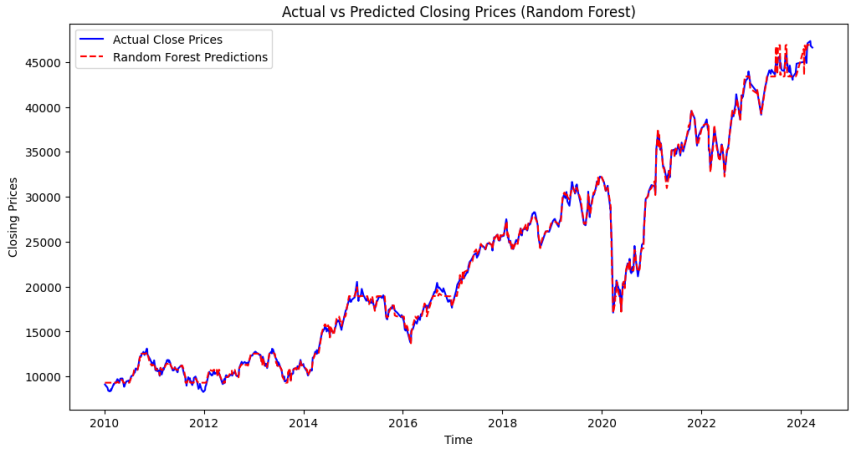



Figure 7.9: This graph shows actual vs Random Forest predicted closing prices for Nifty Bank data

In this analysis, we observed 68.45% accuracy for the former part and 395.79 RMSE for the second part. Although it's much less compared to the Decision Tree in Nifty Bank, it's still unacceptable for such dynamic purposes.

The graphs below show how decision trees and random forests are generalizing the dynamic values, leading to so much RMSE.

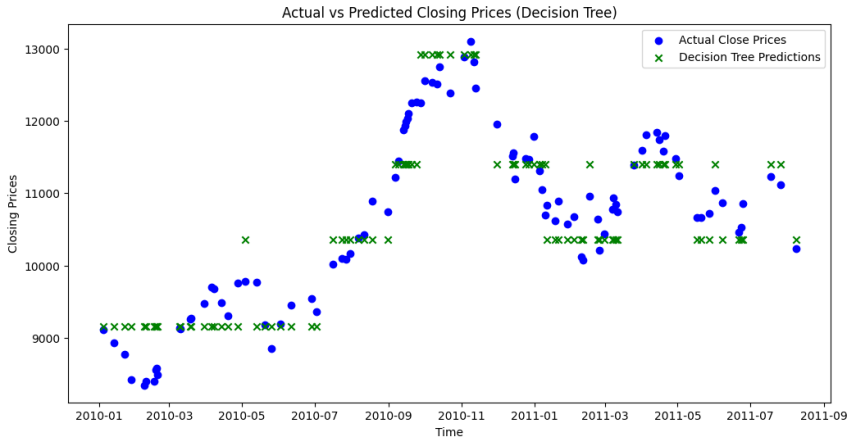



Figure 7.10: This graph shows Decision Tree generalizing the dynamic values of stocks

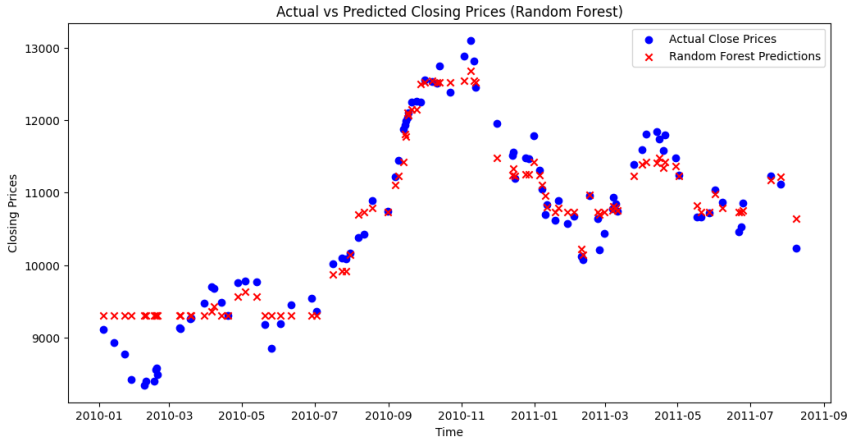



Figure 7.11: This graph shows Random Forests generalizing the dynamic values of stocks

From the Comparative study, we can see that the accuracy of decision trees and random forests is not at par with the accuracies achieved by deep learning models. Therefore we pursue further with LSTM model.

# Chapter 8

# Results

## 8.1 LSTM on NIFTY 50

A scatter plot of true values vs anticipated closing prices for test and training data is shown. Determining the model's quality of fit requires an understanding of this plot. The linear equation (y = x) fits the data the best. The best model performs a bit better for the training-data than the testing-data, which is to be expected. The forecast closing price in the test data is slightly different from the actual closing value in the 20000–23000 range, most likely as a result of the unforeseen market circumstances brought on by the COVID-19 pandemic 2020.

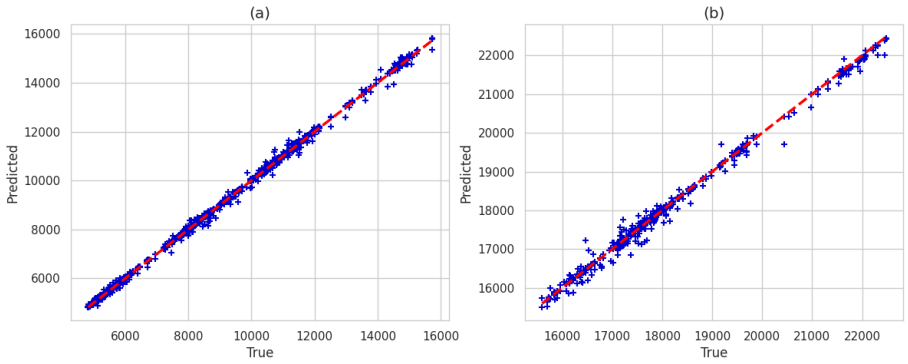



Figure 8.1: Scatter plot for Single Layer LSTM on NIFTY50 data (a) Prediction on Train- ing Data (b) Prediction on Testing Data

The chart shows the original closing price and forecasts based on the most effective single-layer model that has the least RMSE score.The closing price prediction plot for training dataset is almost identical to the actual closing price curve.This means that the most optimal model can almost flawlessly learn both the upward and downward trends of the original closing price

data.In fact the plot for prediction of close price on testing dataset is not overlaping on the True close price, the model precisely depicts the overall trend of the test data with few errors. The approach works effectively even in unexpected market conditions, like a sudden huge loss followed by a rapid V-shaped recovery.

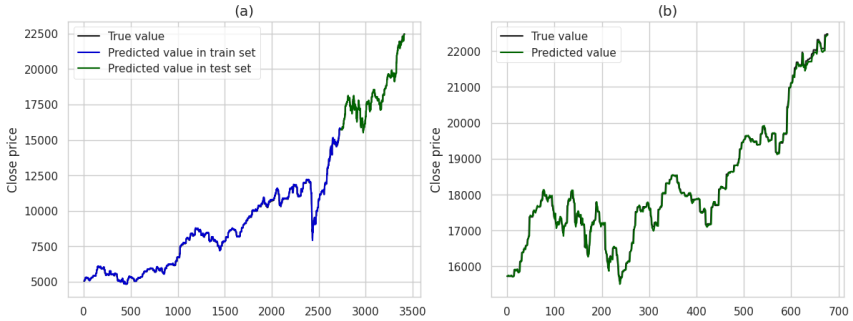



Figure 8.2: Graphs showing actual closing prices alongside their predicted values from the most optimal single-layered LSTM model with the minimum RMSE.

The figures below are plots of average performance results produced from the single and mutli-layered models.It is observed that the evaluation parameters follow a similar trend in both single and multi-layered models. In a layered plot, the RMSE in increasing a a little bit till it reaches the minimum at 200 neurons, while for multilayered the minimum error occurs at [200,100,40]. Afterwards, there was a significant increase in MAPE and RMSE values and a huge fall in R scores. Thus complex models produce bad results compared to simpler versions in out-of-sample data .

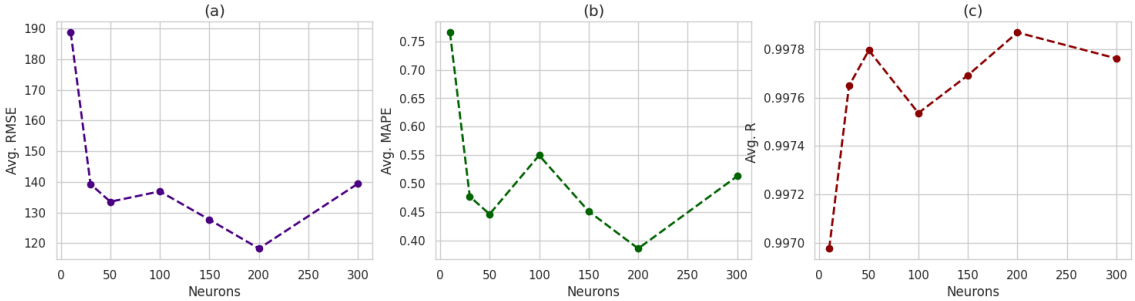



Figure 8.3: Plots: Single-layer LSTM performance, 3 replications”

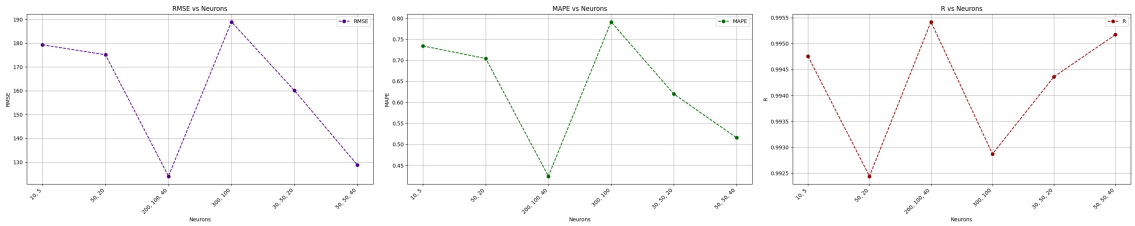



Figure 8.4: Performance metrics produced from Multi-layered LSTM model with two replications.

Table 8.1: Single-Layered LSTM Performance Metrics

| Neurons | RMSE | MAPE | R |
|---|---|---|---|
| 75 | 153.6145 | 0.6412 | 0.9974 |
| 50 | 147.622 | 0.5333 | 0.9978 |
| 200 | 162.4791 | 0.6291 | 0.9972 |
| 100 | 155.1381 | 0.5742 | 0.9976 |
| 150 | 130.0164 | 0.4709 | 0.9979 |
| 300 | 121.6914 | 0.4041 | 0.9976 |

Table 8.2: Multi-Layered LSTM Performance Metrics

| Neurons | RMSE | MAPE | R |
|---|---|---|---|
| 10,5 | 179.2998 | 0.734 | 0.9948 |
| 50, 20 | 175.1102 | 0.7042 | 0.9924 |
| 200, 100, 40 | 124.0804 | 0.4243 | 0.9954 |
| 300, 100 | 188.9507 | 0.7914 | 0.9929 |
| 30, 50, 20 | 160.0763 | 0.6197 | 0.9944 |
| 50, 50, 40 | 128.7577 | 0.5159 | 0.9952 |

## 8.2 LSTM on NIFTY BANK

We replicated the evaluation on NIFTY BANK data after applying the single layer and multi-layer LSTM models on NIFTY 50 data. As expected from the scatter plot, the prediction on the training data is far more accurate than the testing data. The training data represents far better negatively correlated predictions as the values are close to the red line. In comparison, the testing data represents weaker negatively correlated predictions because the model will perform well on training data. In comparison to the NIFTY 50 results, we get a more spread scatter plot, which suggests that the model is more accurate on the former data, which is due to the features that we have taken like INDIA VIX, Treasury Bonds yield, etc, which are affecting NIFTY 50 more than NIFTY BANK.

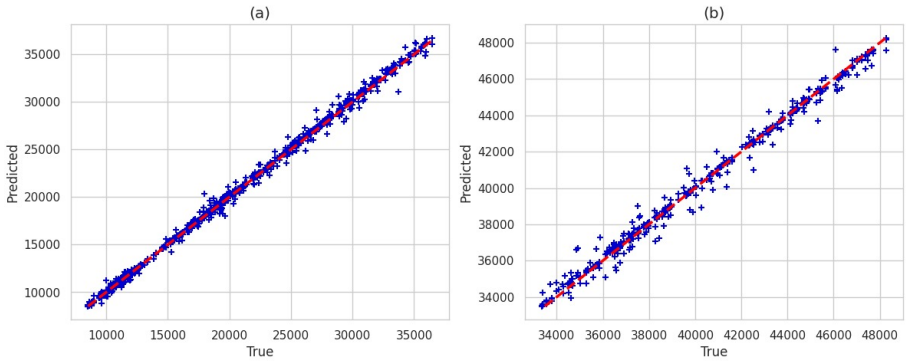



Figure 8.5: Scatter plot for Single Layer LSTM on NIFTY BANK data (a) Prediction on Training Data (b) Prediction on Testing Data

The picture below denotes the best Single-layer LSTM model that we got by varying the value of a number of neurons; it shows the true value v/s the predicted value both in training and testing data. When we closely observed the testing data curve, we found that it deviates somewhat more than NIFTY 50 predictions, as we mentioned earlier.

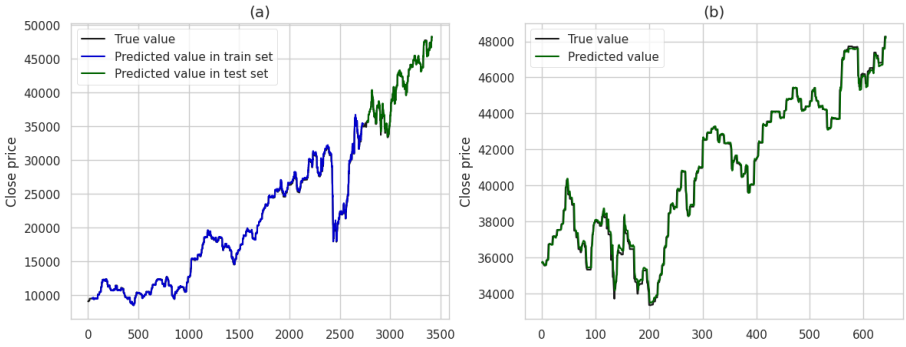



Figure 8.6: True v/s Predicted value on NIFTY BANK data (a) Training and testing combined (b) Zoomed in version for testing data

The following picture demonstrates the performance metrics for Single Layer and Multi-layer LSTM models on NIFTY BANK data; we found that increasing the number of neurons (which means increasing the complexity) leads to the increased value of RMSE, and MAPE, from which we can conclude that the complex models result in less accurate results compared to simpler models. This observation is consistent among single layer and multilayer models.

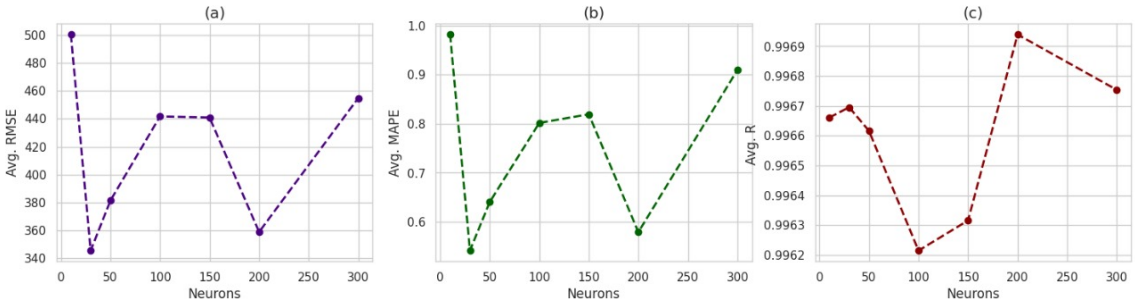



Figure 8.7: Single-layer LSTM performance metrics

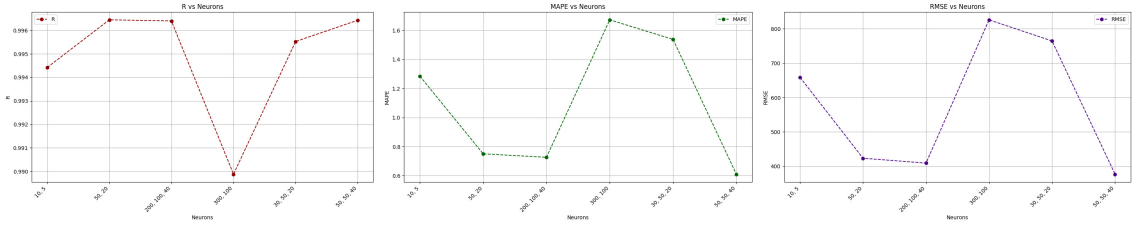


Figure 8.8: Multi-layer LSTM performance metrics

Table 8.3: Single-Layered LSTM Performance Metrics

| Neurons | RMSE | MAPE | R |
|---|---|---|---|
| 30 | 345.3789 | 0.5416 | 0.9967 |
| 50 | 381.3916 | 0.6405 | 0.9966 |
| 100 | 441.694 | 0.8017 | 0.9962 |
| 150 | 440.8821 | 0.8193 | 0.9963 |
| 200 | 358.7474 | 0.5792 | 0.9969 |
| 300 | 454.7921 | 0.9096 | 0.9968 |

Table 8.4: Multi-Layered LSTM Performance Metrics

| Neurons | RMSE | MAPE | R |
|---|---|---|---|
| [10, 5] | 658.7799 | 1.2847 | 0.9944 |
| [50, 20] | 423.5571 | 0.7505 | 0.9964 |
| [200, 100, 40] | 409.3792 | 0.7265 | 0.9964 |
| [300, 100] | 826.1117 | 1.6723 | 0.9899 |
| [30, 50, 20] | 764.6478 | 1.5374 | 0.9955 |
| [50, 50, 40] | 377.0909 | 0.6103 | 0.9964 |

# Chapter 9

# Future Scope

- **Using Ensemble Methods:** Ensemble means a group, and the Ensemble method means using a group of multiple models used all at once for predictions. Instead of using a single LSTM model to make predictions, use a group of Neural Networks. This will greatly improve the accuracy of predictions because if one of the models makes incorrect predictions, other models will act as course correctors.
- **Using More Advanced Architectures:** Using even more advanced neural network architectures like Transformers in order to make more accurate predictions. In the context of artificial intelligence, Transformers are special kinds of neural network architectures. They are also capable of processing sequential data, but instead of processing the data in a sequence they process the whole sequence in parallel by using a special mechanism known as Attention. This architecture was first introduced in a landmark artificial intelligence paper in 2017.
- **Explore more Data Sources:** In this case we are using Stock market data and indicators but we can also use modern data scraping techniques to factor in alternative data sources like news sentiment analysis, social media data, or satellite imagery. Using this we will also be able to take into account psychological and environmental factors to make predictions.
- **Do multi-step predictions:** In the project, only a single value for the next day is being predicted, a further improvement can be predicting values for multiple consecutive future days. This will help investors in planning their investing strategy for multiple upcoming days instead of using just one day to make their plan.

- **Specialized models for Different Markets and Market Segments:** since each market has its own characteristics and factors that affect it, making market specific models for different markets is a viable future improvement. This will improve results for traders who invest in specific smaller market segments.
- **Building User Customizable Models:** after training a model, unfreezing some model parameters which can be tuned by users for their own specific use case. The benefit of this will be increased model performance and generalizability of model. It will also lead to building of new models by users that might be better than the initially trained model.

# Chapter 10

# Conclusion

We have taken two machine learning models to predict:

- Decision and Random Tree
- Long Short Term Memory (LSTM)

A Decision Tree is a very naive model for predicting such a parameter which is impacted by so many values, be it national or international values; we have used open, high, low, and close prices of Nifty and Banknifty and found that the model doesn't have enough accuracy. LSTM has more advantages due to the following reasons:

- **Sequential Data Handling:** Stock data is sequential time series data, thus making LSTM a better choice in our case.
- **Long-Term Dependencies:** LSTM can address the vanishing gradient problem, which helps in capturing long-term dependencies in data, which is difficult to handle by decision trees for datasets that have complex patterns.
- **Feature learning:** Decision trees require feature engineering to perform well, while LSTM can learn the relevant features from the input data automatically.

The upper hand of LSTM over the decision tree is also visible in our research, which states that LSTM has an RMSE of around 130, whereas on the other hand, the decision tree gives us an RMSE of around 270 in Nifty50 predictions. Similar trends are noticed in both the indices, NIFTY 50 and NIFTY BANK.

Finally, we can conclude that although decision trees and random forests are helpful in predicting the prices in Indian Financial Markets, the evaluation metrics are not up to the mark and can't be used to capitalize returns in contrast to LSTM predictions

# References


[1] A. Ghosh, S. Bose, G. Maji, N. Debnath, and S. Sen, "Stock price prediction using lstm on indian share market," in *Proceedings of 32nd International Conference on Computer Applications in Industry and Engineering*, ser. EPiC Series in Computing, Q. Yuan, Y. Shi, L. Miller, G. Lee, G. Hu, and T. Goto, Eds., vol. 63. EasyChair, 2019, pp. 101–110. [Online]. Available: https://easychair.org/publications/paper/LKgn

[2] C.-J. Lu, T.-S. Lee, and C.-C. Chiu, "Financial time series forecasting using independent component analysis and support vector regression," *Decision Support Systems*, vol. 47, no. 2, pp. 115–125, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167923609000323

[3] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221717310652

[4] S. Mehtab, J. Sen, and A. Dutta, *Stock Price Prediction Using Machine Learning and LSTM-Based Deep Learning Models*. Springer Singapore, 2021, p. 88–106. [Online]. Available: http://dx.doi.org/10.1007/978-981-16-0419-5_8

[5] A. G. Ţiţan, "The efficient market hypothesis: Review of specialized literature and empirical research," *Procedia Economics and Finance*, vol. 32, pp. 442–449, 2015, emerging Markets Queries in Finance and Business 2014, EMQFB 2014, 24-25 October 2014, Bucharest, Romania. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212567115014161