FORECASTING STOCK PRICE MOVEMENTS TO PIVOT POINT BASED SUPPORT/RESISTANCE LEVELS USING ARTIFICIAL INTELLIGENCE

by

GEORGE VICTOR K J, MS, PGDMLAI, BTECH

DISSERTATION

Presented to the Swiss School of Business and Management Geneva

In Partial Fulfillment

Of the Requirements

For the Degree

DOCTOR OF BUSINESS ADMINISTRATION

SWISS SCHOOL OF BUSINESS AND MANAGEMENT GENEVA

FEBRUARY, 2024

FORECASTING STOCK PRICE MOVEMENTS TO PIVOT POINT BASED SUPPORT/RESISTANCE LEVELS USING ARTIFICIAL INTELLIGENCE

by

GEORGE VICTOR K J

APPROVED BY

____
Dissertation chair

RECEIVED/APPROVED BY:

____
Admissions Director

Dedication

This dissertation is dedicated to the vast majority of individual day traders who, despite their efforts, face challenges in making any profits from the stock market. Drawing from my initial experiences as a part-time trader, I have gained an understanding of the challenges, frustrations, and persistent aspirations for profitability that are common in this field. This insight was further deepened by my realization of the limitations of day trading without adequate tools and experience, leading to my eventual transition to focus on option selling strategies. It is my aspiration that this research provides an additional resource, potentially enhancing the trading strategies and performances of individual traders.

Acknowledgements

I want to take this opportunity to thank the following individuals who played an important role during this research work.

First and foremost, I would like to express my gratitude to Prof. Derrald Stice who has been my mentor and guide for this doctoral program. Your support and guidance have been exemplary in getting this dissertation done.

I am deeply grateful to my family for being my support throughout this journey, I would especially like to express my deep gratitude to my children, Miral and Raynel, for their patience and understanding as I devoted considerable time to this research, often at the expense of my duties as a father. Insightful discussions with Miral on various facets of neural networks have been invaluable to this work. Also, I extend my heartfelt thanks to my parents, who provided me with everything during my childhood, laying the foundation for my personal and academic growth.

Finally, I want to extend my sincere appreciation to my friends and well-wishers – Jibin, Minu, Edward, and Jyothi – for believing in me, being there for me, and encouraging me through this challenging yet rewarding endeavour. Special mention must be given to Jyothi for guiding me and providing valuable suggestions, and to Jibin for recommending tools for grammar and spell-checking.

I am profoundly thankful to everyone involved in my journey for their invaluable support in reaching this significant milestone.

ABSTRACT

FORECASTING STOCK PRICE MOVEMENTS TO PIVOT POINT BASED SUPPORT/RESISTANCE LEVELS USING ARTIFICIAL INTELLIGENCE

GEORGE VICTOR K J
2024

Dissertation Chair:
Co-Chair:

Stock Trading, especially day trading, is a challenging task. Pivot Point Analysis, being a leading indicator, is an important tool for day trading, but the literature review suggests that there is a research gap in using artificial intelligence-based systems to forecast pivot levels which the price is expected to reach in future. This study aims to outline a system designed using Long Short-Term Memory (LSTM) to map the price movements along with multiple technical indicators taken as inputs to predict the next level, among the pivot levels, that the price is expected to reach in future.

The study proposes a method to map the pivot lines which change daily for day trading to a single output column so that it is possible for an artificial intelligence-based model to efficiently learn from it. The study also outlines the model architecture for designing such a system using Long Short-Term Memory networks and after conducting 44 different experiments, suggests the optimal configuration parameters for the proposed model. The research also proposes different metrics to evaluate the proposed model. The study conducts 6 more experiments to indicate that the proposed model works across different types of stock assets like indices and individual stocks, and across different timeframes for day, swing or positional trading. Finally, the research formulates an illustrative trading strategy to use the proposed model efficiently in real-world scenarios.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I:
INTRODUCTION

1.1 Introduction

In the stock market, most of the individual day traders do not make any profits (Chague, De-Losso & Giovannetti, 2019) (Jordan & Diltz, 2003) (Barber et al., 2005). There are many trading strategies which can be used for day trading, but trading strategies that are based on pivot points have more chances of making profits (Tian et al., 2012) (Frykmer & Johnsson, 2019) (Person, 2004).

There were many attempts to use machine learning and deep learning to aid the trader. Long Short-Term Memory (LSTM) based networks proved to be a better option for dealing with stock market data (Fischer & Krauss, 2017). Some of the researchers even tried using Pivot Point-based support and resistance levels as the input parameters to the LSTM to predict prices (Dwivedi et al., 2022) (Kumbhare et al., 2023).

This study addresses a research gap in the literature review by putting forward an LSTM-based system which can forecast the future level the stock price is going to reach, from the list of all the Support/Resistance levels defined by the pivot point analysis.

1.2 Research Problem

Pivot Point Analyis is a very important tool for stock traders. It provides important price levels based on the previous period's high, low and close values. These levels can be calculated before the trading session, making it a 'leading indicator' and these levels stay constant during the current trading period.

The research problem for this study is to propose an artificial intelligence-based model which takes price information as well as technical indicators and candle patterns and predicts one among these pivot levels indicating that the price is expected to reach that level shortly.

The following figure illustrates this with an example set of input features - Open, High, Low, Close, RSI, ATR, Doji - and an artificial intelligence-based model is predicting the next level as ‘s1’. Output could have been any one of the daily pivot levels from the list of pp, pb, pt, s1, s2, s3, r1, r2, r3.

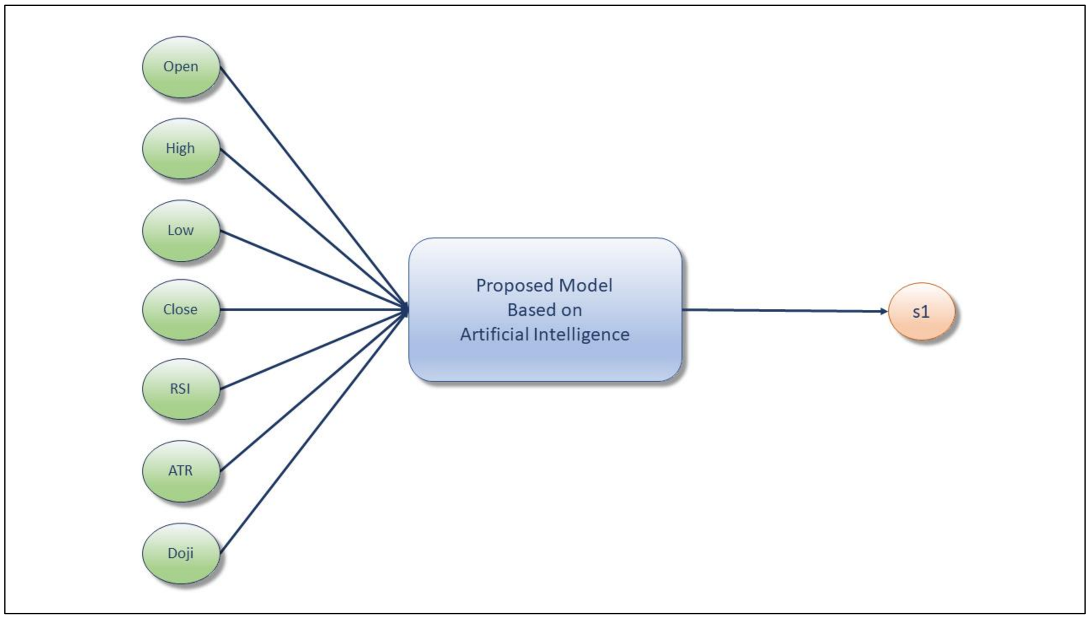


Figure 1.2: Illustration of the Research Problem

1.3 Purpose of Research

The long-term goal of the research is to develop a system using Artificial Intelligence that can forecast the price levels which has the highest probability of being touched, among a list of support/resistance levels defined by pivot point analysis. The objective of the current study is to provide a comprehensive review of the literature and outline a framework for predicting the above levels using Long Short-Term Memory (LSTM). In particular, the study has the following sub-objectives:

1. To develop a method to map the support/resistance lines defined by pivot point analysis into a dataset that the LSTM can learn from;
2. To develop a system using LSTM to forecast the future price level that the price is going to reach, from a list of support/resistance lines defined by pivot point analysis;
3. To experiment with different technical indicators and candle patterns related to the stock market and find out the optimal ones which work better for this system;
4. To experiment and find out if this system works well across different types of stock assets such as indices and individual stocks;
5. To experiment and find out if this system works across different timeframes so that it can be used for day trading and other types of trading too.

1.4 Significance of the Study

The result of this study will be valuable to the traders, especially the day traders, in making an informed decision about taking high-probability trades. It can help the automated trading systems too, by deploying these deep learning artificial intelligence models into production and making quick decisions about taking trades. This study could also aid big financial institutions engaged in having millions of financial trading transactions per month, by taking the proposed model as another indicator for automatically getting trading signals, and also to take a large number of automated trades conforming to the trading strategy and fundamental principles. Finally, this is helpful for researchers trying to bring state-of-the-art artificial intelligence-based models into financial trading.

1.5 Research Purpose and Questions

The traders, especially the day traders, can benefit vastly if she/he know the future level the price is going to reach, from a list of support/resistance levels defined by the pivot point analysis. By knowing this in advance, the trader can calculate if the trade is going to be beneficial with a good risk-to-reward ratio or not (Katz & McCormick, 2000), and place the stop-loss and target orders close to support/resistance levels of the pivot point analysis (Person, 2004).

With the help of Artificial Intelligence, we are trying to forecast the pivot level the price is expected to touch among the support/resistance levels defined by pivot point analysis. To do that, the following research questions need to be addressed:

1. Can an LSTM-based system be trained to map the price movements along with technical indicators into pivot point-based support/resistance levels?
2. Since the pivot point-based support/resistance levels change every day, how do we map these ever-changing levels into a dataset that the LSTM can learn from?
3. What are the technical indicators that help in getting the LSTM-based system to learn better?

4. Can this learning apply to different types of stock assets like indices and stocks?
5. Can this be applied to different timeframes so that it works well for other types of trading, apart from day trading?

CHAPTER II:
REVIEW OF LITERATURE

2.1 The Majority of Day Traders Incur Losses

The study conducted in the Brazilian stock market, it is found that almost 97% of individual day traders hardly make any profits (Chague, De-Losso & Giovannetti, 2019). In the US stock market, about 64.2% of the day traders lose money (Jordan & Diltz, 2003). In a large-scale study conducted in the Taiwan stock market, it was found that over 80% of day traders make losses (Barber et al., 2005).

2.2 Significance of Pivot Point Analysis in Day Trading

The pivot point analysis is a great tool for day traders as it is a leading indicator and it provides accurate entry and exit points (Person, 2004). Having trading strategies that are based on pivot points has more chances of making profits (Tian et al., 2012). Pivot points might also serve as a useful indicator of when the market will react and cause increased price volatility, (Frykmer & Johnsson, 2019).

2.3 Risk Management in Trading

A good exit strategy is more important than even the entry and it is usually accomplished using a stop-loss order or a target order (Katz & McCormick, 2000). (Person, 2004) suggests that it is a good practice to place the stop loss orders and target orders close to support and resistance levels associated with the pivot point system.

2.4 The 1:3 Risk-to-Reward Ratio: A Crucial Principle in Trading

It is very important to have a correct risk-to-reward ratio to get better overall profit in the long run and (Katz & McCormick, 2000) suggests that the optimal ratio is

1:3 where the stop loss order is placed at 1.5 Average True Range points away from the entry and the target order is placed at 4.5 Average True Range points away from the entry.

2.5 Machine Learning and Deep Learning in Trading

(Frattini et al., 2022) proposed alternative data and machine learning algorithms such as Extreme Gradient Boosting and Light Gradient Boosting Machine to predict the trend. (Ali et al., 2021) proposed machine learning using SVM and ANN with 1 layer to predict the direction of the movement price movement using 15 technical indicators as input parameters.

2.6 The LSTM Excels in the Analysis of Stock Market Data

(Fischer & Krauss, 2017) proved that LSTM-based networks provide impressive results with stock market data compared to deep neural networks, random forest and logistic regression classifiers. (Nirob & Hasan, 2023) also stated that the LSTM has high accuracy while dealing with stock market data and analyzed its accuracy by comparing it with popular conventional methods such as Simple Moving Average and Exponential Moving Average. According to (Mittal & Chauhan, 2021), LSTM with technical indicators have significantly reduced the error value by 2.42% and improved the overall performance of the system as compared to other machine learning frameworks that do not account for the effect of technical indicators. (Yadav, Jha & Sharan, 2020) used LSTM to forecast the close price of the NSE Nifty50 index. (Zaheer et al., 2023) improved by forecasting 2 output variables – close price and next day’s high price using Convolutional Neural Network, LSTM and Recurrent Neural Network along with 6 input parameters.

2.7 Prior Research on Pivot Points as Input Parameters to LSTM

(Dwivedi et al., 2022) used LSTM with various combinations of technical indicators including pivot points as input parameters and found out that at least 2 indicators should be combined to get good results. (Kumbhare et al., 2023) put forward a method with the help of LSTM for forecasting stock trends using the technical indicators Average Directional Index, Supertrend and Fibonacci Pivot Points.

2.8 Summary

Through the literature review, we can conclude that there is an important research gap which could help day traders if we can come up with a study to predict the future level that the price is going to reach among the list of pivot points levels, with the help of artificial intelligence. There were previous studies in which pivot points were taken as input parameters to predict price movements, but no study, so far, is available which predicts the pivot levels as the output of any deep learning models. This will enable the day traders to identify good trades and to place stoploss and target at efficient levels which will be close to a support or resistance level defined by the pivot point analysis. Also, they can decide whether to take the trade or not based on whether the predicted level conforms to the appropriate risk-to-reward ratio or not.

CHAPTER III:
METHODOLOGY

3.1 Overview of the Research Problem

Pivot point analysis is a famous price forecasting method used by day traders and it is a leading indicator which helps the day traders to know the important price levels even before the actual trading window opens (Person, 2004). Pivot points are mathematically calculated using the following equations (Anon, n.d.a):

Table 2.1: Mathematical Formulae to Calculate Pivot Levels. Source: (Anon, n.d.f)

Pivot level calculations based on High (h), Low (l) and Close (c) values

Pivot Line Name	Abbreviation	Mathematical Formula
Pivot Point	pp	(h + l + c) ÷ 3
Bottom Central Pivot	pb	(h + l) ÷ 2
Top Central Pivot	pt	(pp - pb) + pp
First support level	s1	(2 * pp) – h
Second support level	s2	pp – (h – l)
Third support level	s3	pp – 2 * (h – l)
First resistance level	r1	(2 * pp) – l
Second resistance level	r2	pp + (h – l)
Third resistance level	r3	pp + 2 * (h – l)

Pivot levels can be daily, weekly monthly etc., based on whether we are using the high/low/close prices of the previous day or week or month. Among these, for day trading, the most important one is daily pivot point levels (Anon, n.d.c). For this study, we are using only the daily pivot lines for easy visualization, but the same study can be extended with all three sets of pivot lines.

The following figure shows an example of pivot points-based support/resistance lines calculated for the Nifty 50 Index on the 17th of January 2023.

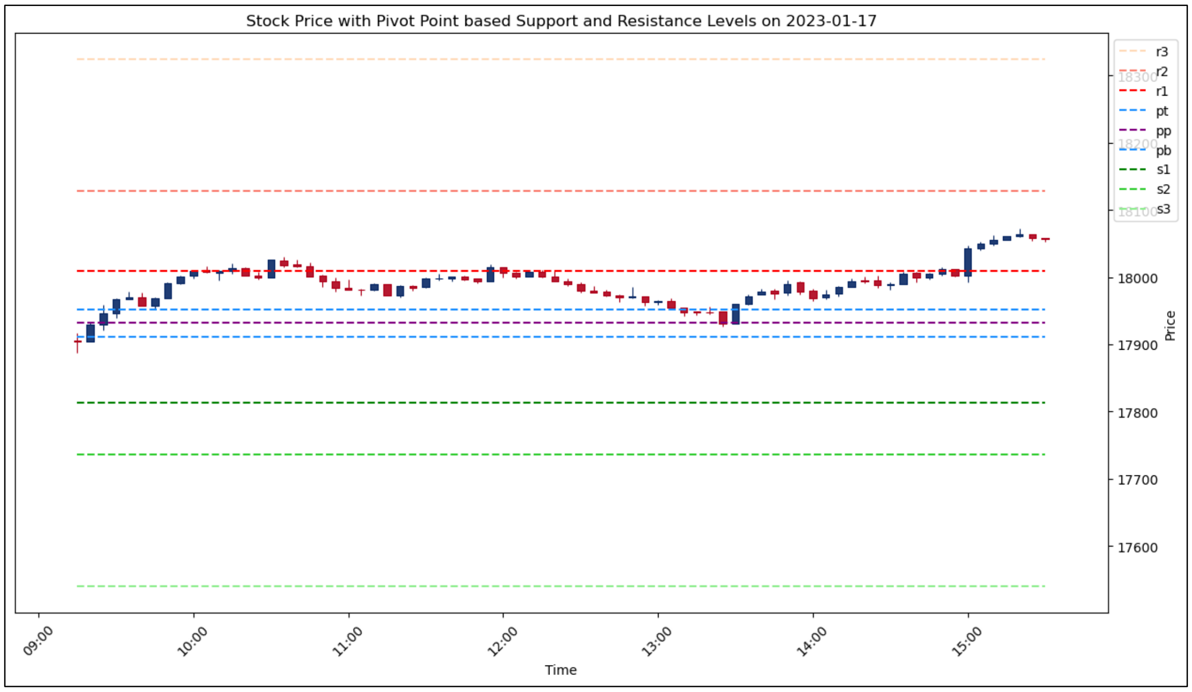


Figure 3.1 (a): Nifty 50 on 17th January 2023 with Pivot Levels, calculated daily

The research problem of this study is to develop an artificial intelligence-based system that can forecast which level, among these 9 daily pivot point levels - pp, pb, pt, s1, s2, s3, r1, r2, r3 - the price is going to reach in future. The following figure illustrates this with an example set of input features - Open, High, Low, Close, RSI, ATR, Doji - and the proposed model is predicting the next level as 's1'. Output could have been any one of the pivot levels from the list of pp, pb, pt, s1, s2, s3, r1, r2, r3.

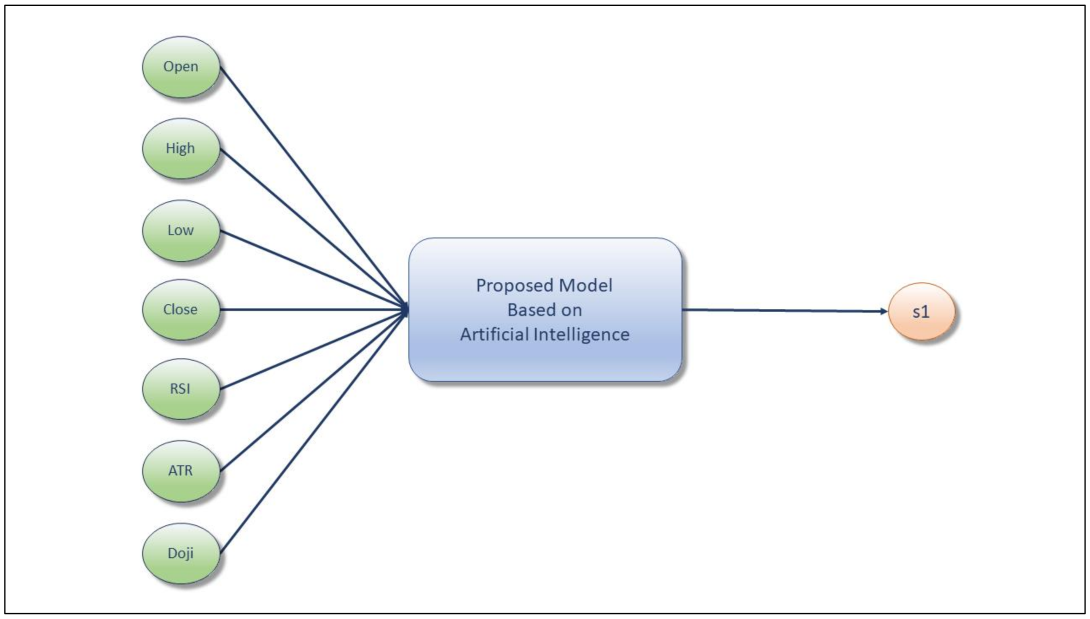


Figure 3.1 (b): Illustration of the Research Problem

3.2 Research Purpose and Questions

The purpose of the current study is to outline a framework for predicting the future level among the 9 daily pivot point levels using a Long Short-Term Memory (LSTM) based artificial intelligence system. In particular, the study has the following sub-objectives:

1. To develop a method to map the support/resistance lines defined by pivot point analysis into a dataset that the LSTM can learn from;
2. To develop a system using LSTM to forecast the future price level that the price is going to reach, from a list of support/resistance lines defined by pivot point analysis;

3. To experiment with different technical indicators and candlestick patterns related to the stock market and find out the list of technical indicators which work better for this system;
4. To experiment and find out if this system works well for different types of stock assets like indices and individual stocks;
5. To experiment and find out if this system works well in different timeframes for day, swing or positional trading;
6. To formulate an example trading strategy with the proposed model.

Following are the research questions this study is trying to answer:

1. Can an LSTM-based system be trained to map the price movements along with technical indicators into pivot point-based support/resistance levels?
2. Since the pivot point-based support/resistance levels change every day, how do we map these ever-changing levels into a dataset that the LSTM can learn from?
3. What are the technical indicators and candlestick patterns that help in getting the LSTM-based system to learn better?
4. Can this learning be applied to different types of stock assets like indices and individual stocks?
5. Can this learning be applied to different time frames?

The responses to the research questions outlined above will dictate the progression and structure of this thesis, serving as a guiding framework for the subsequent analysis and discussions.

3.3 Research Design

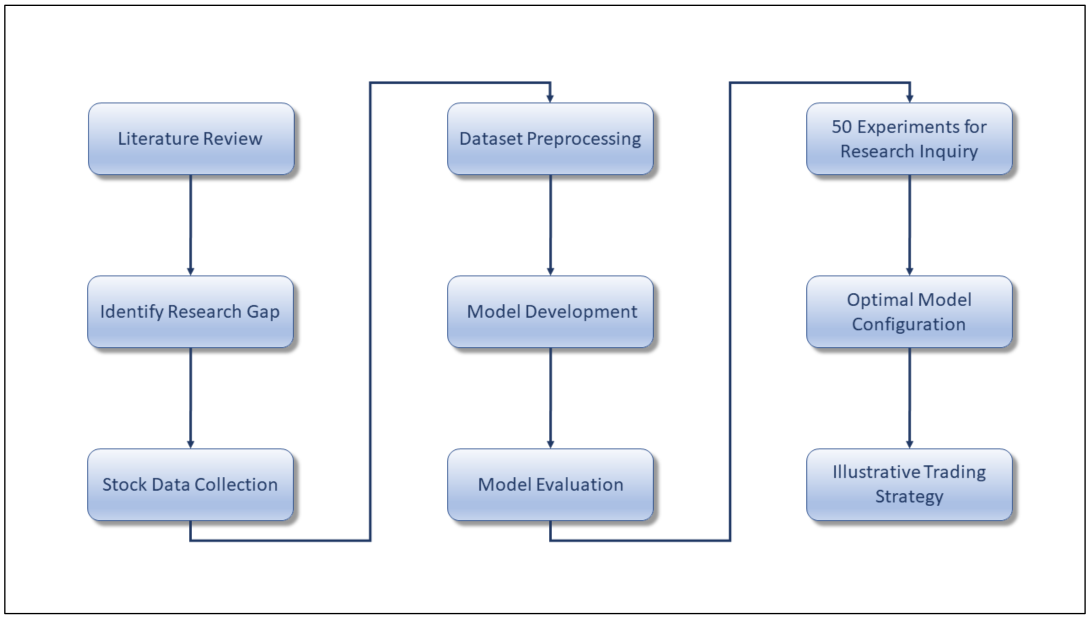


Figure 3.3: Research Design

The primary research method for this study is a literature review and modelling of a system using Long Short-Term Memory (LSTM). This study will first review previous literature to understand the research gap in studying the relationship between pivot point-based levels and price movements, and forecasting these levels as an output parameter. Based on this understanding, an appropriate stock market dataset will be collected with trade details up to every minute. Then this dataset will be modified with technical indicators as well as the pivot point levels, and then significant price movements to any levels will be marked so that a deep learning model can learn from this updated dataset.

After that, a system based on LSTM will be trained using the updated dataset which could forecast the future levels the price could reach, from the list of pivot point levels for that day. Different metrics for evaluating the model will also be proposed. Finally, once this system is in place, more experiments will be conducted to find out the best technical indicators to use in this system, if this system works for different time frames for day, swing or positional trading, and if the system works with different types of stock assets like indices and individual stocks. Based on these experiments, an optimal model configuration and an illustrative trading strategy will be proposed.

3.3.1 Data Collection

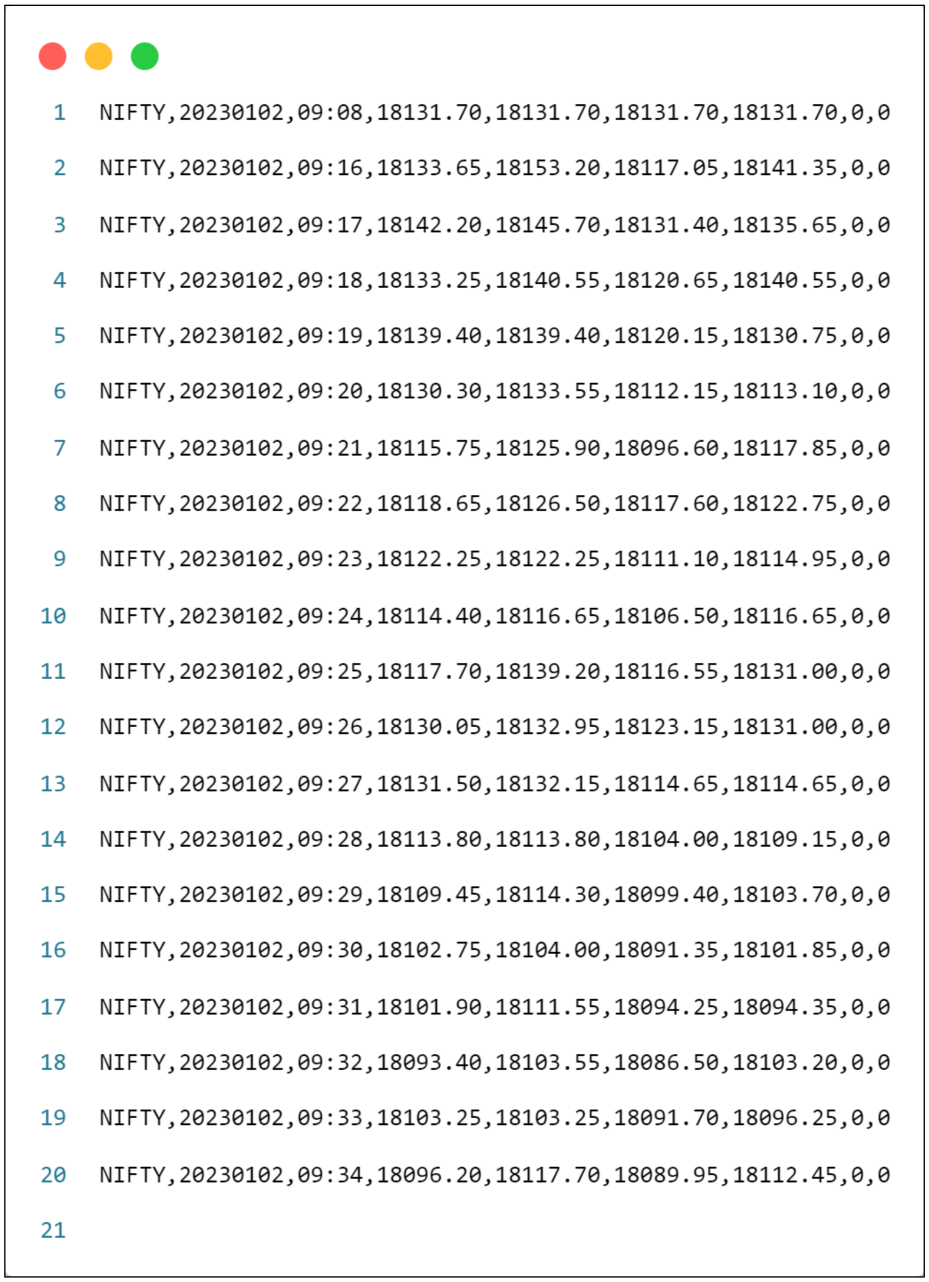
The dataset needed for this research can be collected from any source which provides stock market data with 5-minute candles. For this study, the data is collected from (rmsharma@gmail.com, n.d.). This dataset provides one-minute transactions that happened in the National Stock Exchange of India from 2012 to February 2023 and some more historical data can be found up to 2011. However the consistency of the data is not guaranteed and there is considerable disparity between different years, on what is being provided and the naming conventions being followed.

For this Study, a Python script is developed from scratch to fetch the consolidated data from 2017 to 2022. Also, the data from January and February months from 2023 are collected for testing. Data from the Nifty 50 Index, as well as 4 stocks – Infosys, Hindustan Unilever Ltd, Hero Motocorp Ltd and ITC Limited, are collected to carry out various experiments for this research study. The extracted dataset will be a comma-separated values (CSV) with the following fields in the respective order:

1. 'Symbol'
2. 'Date'

3. 'Time'
4. 'Open'
5. 'High'
6. 'Low'
7. 'Close'
8. 'Volume'
9. 'Unknown'

A sample dataset file is shown below for reference.



```
NIFTY,20230102,09:08,18131.70,18131.70,18131.70,18131.70,0,0
NIFTY,20230102,09:16,18133.65,18153.20,18117.05,18141.35,0,0
NIFTY,20230102,09:17,18142.20,18145.70,18131.40,18135.65,0,0
NIFTY,20230102,09:18,18133.25,18140.55,18120.65,18140.55,0,0
NIFTY,20230102,09:19,18139.40,18139.40,18120.15,18130.75,0,0
NIFTY,20230102,09:20,18130.30,18133.55,18112.15,18113.10,0,0
NIFTY,20230102,09:21,18115.75,18125.90,18096.60,18117.85,0,0
NIFTY,20230102,09:22,18118.65,18126.50,18117.60,18122.75,0,0
NIFTY,20230102,09:23,18122.25,18122.25,18111.10,18114.95,0,0
NIFTY,20230102,09:24,18114.40,18116.65,18106.50,18116.65,0,0
NIFTY,20230102,09:25,18117.70,18139.20,18116.55,18131.00,0,0
NIFTY,20230102,09:26,18130.05,18132.95,18123.15,18131.00,0,0
NIFTY,20230102,09:27,18131.50,18132.15,18114.65,18114.65,0,0
NIFTY,20230102,09:28,18113.80,18113.80,18104.00,18109.15,0,0
NIFTY,20230102,09:29,18109.45,18114.30,18099.40,18103.70,0,0
NIFTY,20230102,09:30,18102.75,18104.00,18091.35,18101.85,0,0
NIFTY,20230102,09:31,18101.90,18111.55,18094.25,18094.35,0,0
NIFTY,20230102,09:32,18093.40,18103.55,18086.50,18103.20,0,0
NIFTY,20230102,09:33,18103.25,18103.25,18091.70,18096.25,0,0
NIFTY,20230102,09:34,18096.20,18117.70,18089.95,18112.45,0,0

```

Figure 3.3.1: CSV Format of the extracted dataset

The Python script used for extracting and consolidating this data can be found in Appendix B.

3.3.2 Data Preprocessing

The datasets are loaded into a pandas data frame and preprocessed to achieve the following things

1. Combined ‘Date’ and ‘Time’ fields to form a Pandas DateTime Object and saved it as the column ‘DateTime’. This is set as the index of the data frame
2. Dropped the ‘Date’, ‘Time’ and ‘Unknown’ columns
3. Filtered out rows which are not within the trading window of 9:15 AM IST and 3:30 PM IST
4. Resampled and aggregated the rows to 5-minute candle data
5. Dropped all the rows with NaN values
6. Pivot point-based support and resistance levels are calculated and placed under 9 new columns - 'pp', 'pb', 'pt', 'r1', 's1', 'r2', 's2', 'r3', 's3'
7. The following different technical indicators and candle patterns are added as columns of this Pandas data frame. Please refer to Appendix A for their detailed descriptions.
   a. RSI
   b. EMA7
   c. EMA14
   d. EMA_CROSSOVER_7AND14
   e. Momentum
   f. ATR
   g. ADX, Plus DI and Minus DI
   h. Williams' %R
   i. Doji
   j. Marubozu

k. Hammer

l. Hanging Man

m. Engulfing

n. Harami

o. Morning Star

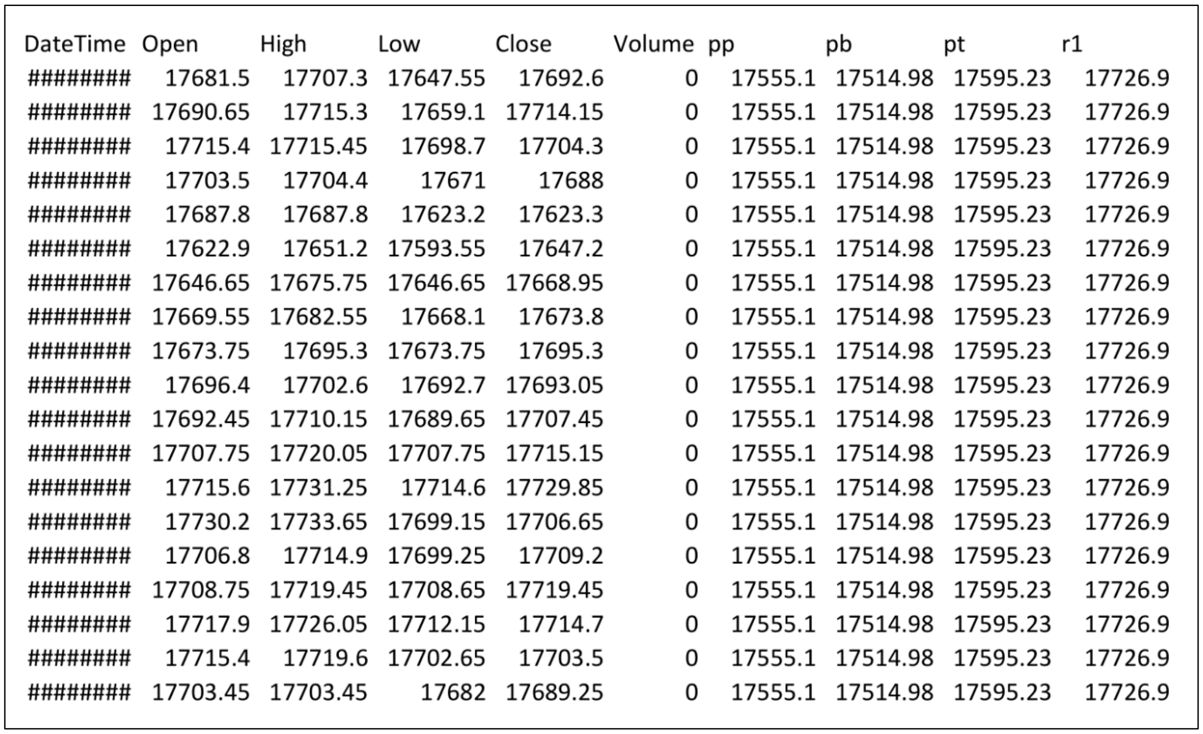
p. Evening Star

q. 3 White Soldiers

r. 3 Black Crows

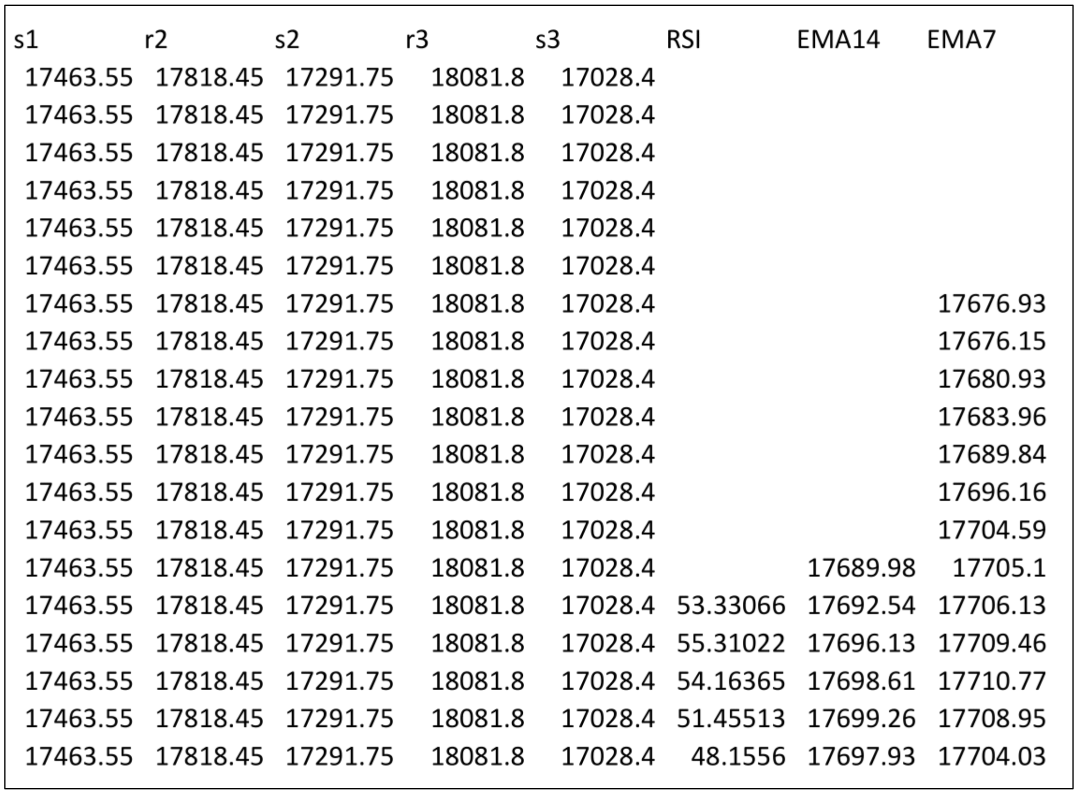
8. Marked the next level, among the 9 pivot levels, the future candles are going to touch under the column ‘NextLevel’. This will be used as the output variable while training the AI model. This is one of the most important pieces of this research study. Please refer to the Python code in Appendix C to understand how this is calculated.

The output of this preprocessing step will be the training data. A slice of the training data is given below for reference.



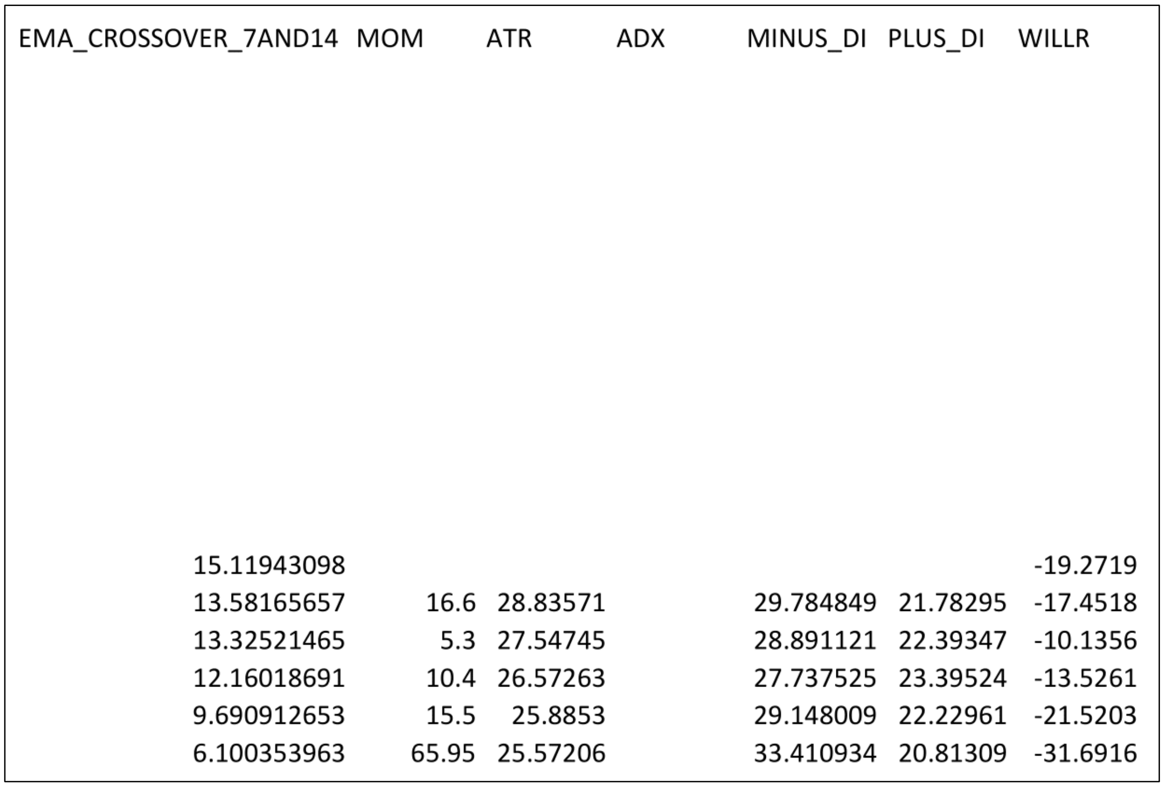
DateTime	Open	High	Low	Close	Volume	pp	pb	pt	r1
########	17681.5	17707.3	17647.55	17692.6	0	17555.1	17514.98	17595.23	17726.9
########	17690.65	17715.3	17659.1	17714.15	0	17555.1	17514.98	17595.23	17726.9
########	17715.4	17715.45	17698.7	17704.3	0	17555.1	17514.98	17595.23	17726.9
########	17703.5	17704.4	17671	17688	0	17555.1	17514.98	17595.23	17726.9
########	17687.8	17687.8	17623.2	17623.3	0	17555.1	17514.98	17595.23	17726.9
########	17622.9	17651.2	17593.55	17647.2	0	17555.1	17514.98	17595.23	17726.9
########	17646.65	17675.75	17646.65	17668.95	0	17555.1	17514.98	17595.23	17726.9
########	17669.55	17682.55	17668.1	17673.8	0	17555.1	17514.98	17595.23	17726.9
########	17673.75	17695.3	17673.75	17695.3	0	17555.1	17514.98	17595.23	17726.9
########	17696.4	17702.6	17692.7	17693.05	0	17555.1	17514.98	17595.23	17726.9
########	17692.45	17710.15	17689.65	17707.45	0	17555.1	17514.98	17595.23	17726.9
########	17707.75	17720.05	17707.75	17715.15	0	17555.1	17514.98	17595.23	17726.9
########	17715.6	17731.25	17714.6	17729.85	0	17555.1	17514.98	17595.23	17726.9
########	17730.2	17733.65	17699.15	17706.65	0	17555.1	17514.98	17595.23	17726.9
########	17706.8	17714.9	17699.25	17709.2	0	17555.1	17514.98	17595.23	17726.9
########	17708.75	17719.45	17708.65	17719.45	0	17555.1	17514.98	17595.23	17726.9
########	17717.9	17726.05	17712.15	17714.7	0	17555.1	17514.98	17595.23	17726.9
########	17715.4	17719.6	17702.65	17703.5	0	17555.1	17514.98	17595.23	17726.9
########	17703.45	17703.45	17682	17689.25	0	17555.1	17514.98	17595.23	17726.9

Figure 3.3.2 (a): A Sample from the training data – Part 1 of 5



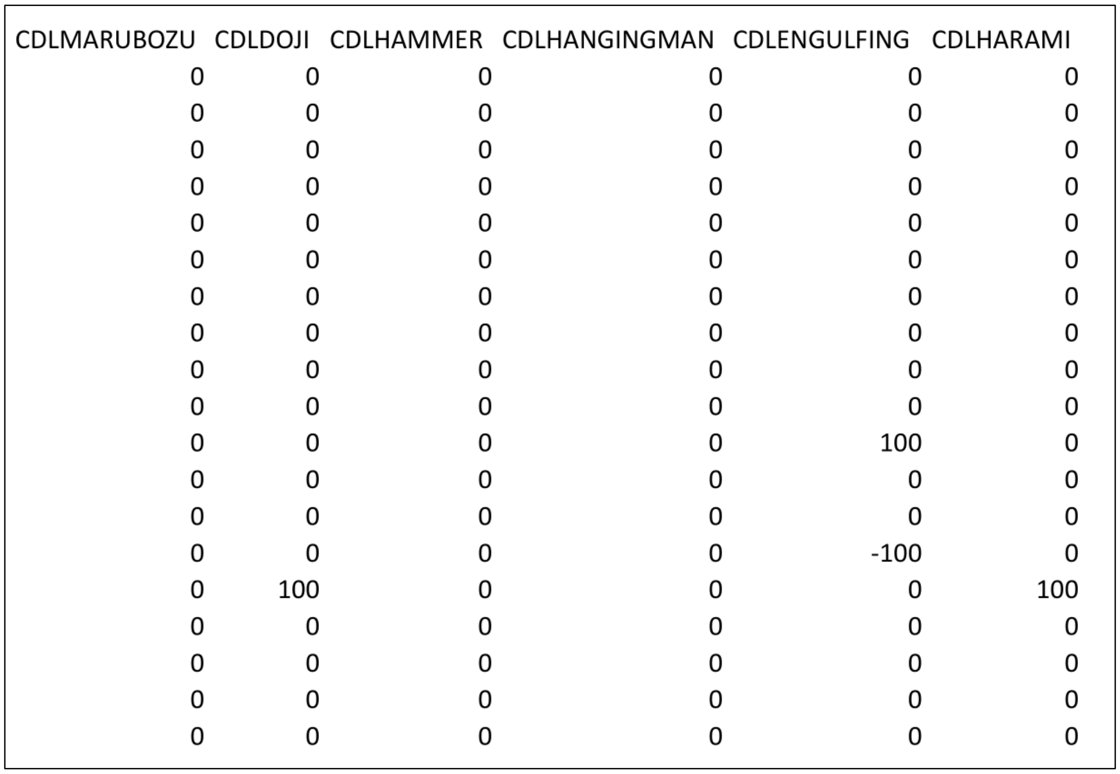
s1	r2	s2	r3	s3	RSI	EMA14	EMA7
17463.55	17818.45	17291.75	18081.8	17028.4			
17463.55	17818.45	17291.75	18081.8	17028.4			
17463.55	17818.45	17291.75	18081.8	17028.4			
17463.55	17818.45	17291.75	18081.8	17028.4			
17463.55	17818.45	17291.75	18081.8	17028.4			
17463.55	17818.45	17291.75	18081.8	17028.4			
17463.55	17818.45	17291.75	18081.8	17028.4			17676.93
17463.55	17818.45	17291.75	18081.8	17028.4			17676.15
17463.55	17818.45	17291.75	18081.8	17028.4			17680.93
17463.55	17818.45	17291.75	18081.8	17028.4			17683.96
17463.55	17818.45	17291.75	18081.8	17028.4			17689.84
17463.55	17818.45	17291.75	18081.8	17028.4			17696.16
17463.55	17818.45	17291.75	18081.8	17028.4			17704.59
17463.55	17818.45	17291.75	18081.8	17028.4		17689.98	17705.1
17463.55	17818.45	17291.75	18081.8	17028.4	53.33066	17692.54	17706.13
17463.55	17818.45	17291.75	18081.8	17028.4	55.31022	17696.13	17709.46
17463.55	17818.45	17291.75	18081.8	17028.4	54.16365	17698.61	17710.77
17463.55	17818.45	17291.75	18081.8	17028.4	51.45513	17699.26	17708.95
17463.55	17818.45	17291.75	18081.8	17028.4	48.1556	17697.93	17704.03

Figure 3.3.2 (b): A Sample from the training data – Part 2 of 5



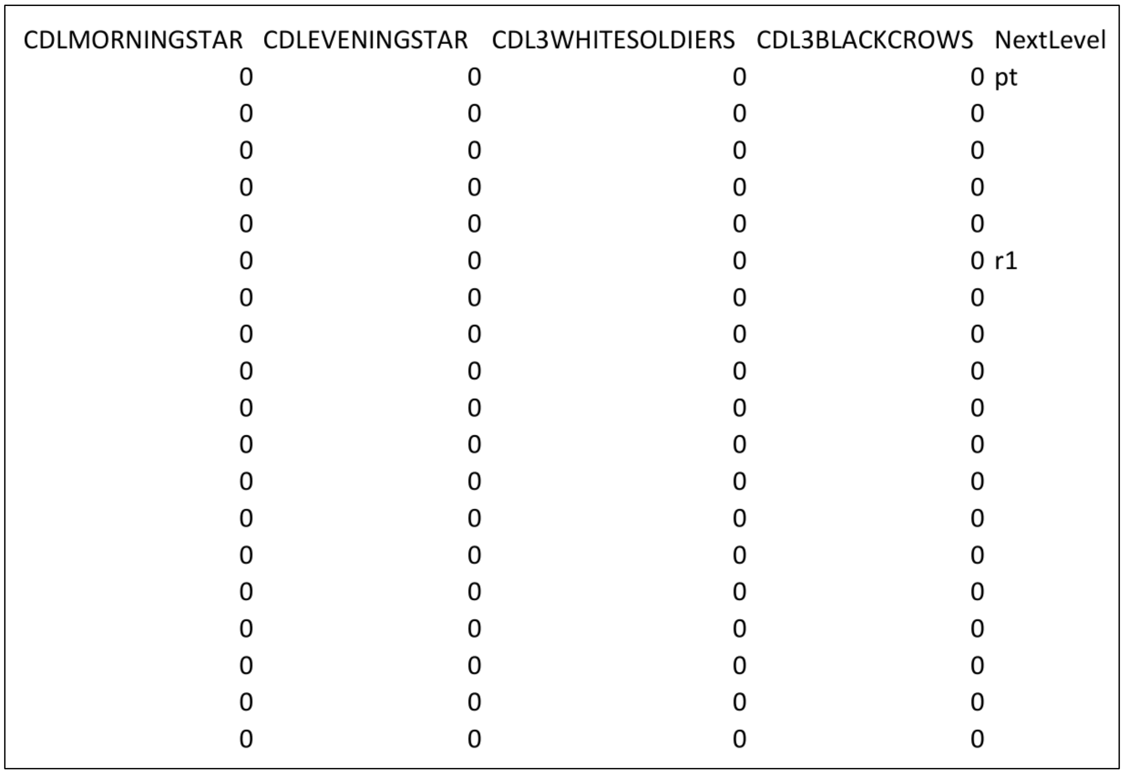
EMA_CROSSOVER_7AND14	MOM	ATR	ADX	MINUS_DI	PLUS_DI	WILLR
15.11943098						-19.2719
13.58165657	16.6	28.83571		29.784849	21.78295	-17.4518
13.32521465	5.3	27.54745		28.891121	22.39347	-10.1356
12.16018691	10.4	26.57263		27.737525	23.39524	-13.5261
9.690912653	15.5	25.8853		29.148009	22.22961	-21.5203
6.100353963	65.95	25.57206		33.410934	20.81309	-31.6916

Figure 3.3.2 (c): A Sample from the training data – Part 3 of 5



CDLMARUBOZU	CDLDOJI	CDLHAMMER	CDLHANGINGMAN	CDLENGULFING	CDLHARAMI
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	100	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	-100	0
0	100	0	0	0	100
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Figure 3.3.2 (d): A Sample from the training data – Part 4 of 5



CDLMORNINGSTAR	CDLEVENINGSTAR	CDL3WHITESOLDIERS	CDL3BLACKCROWS	NextLevel
0	0	0	0	pt
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	r1
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	
0	0	0	0	

Figure 3.3.2 (e): A Sample from the training data – Part 5 of 5

The following diagram visualizes the candle data, along with the pivot levels and the output variable ‘NextLevel’. ‘NextLevel’ will have one of the values among ['', 'pb', 'pp', 'pt', 'r1', 'r2', 'r3', 's1', 's2', 's3']. This output variable is rendered as orange dots in the figure.

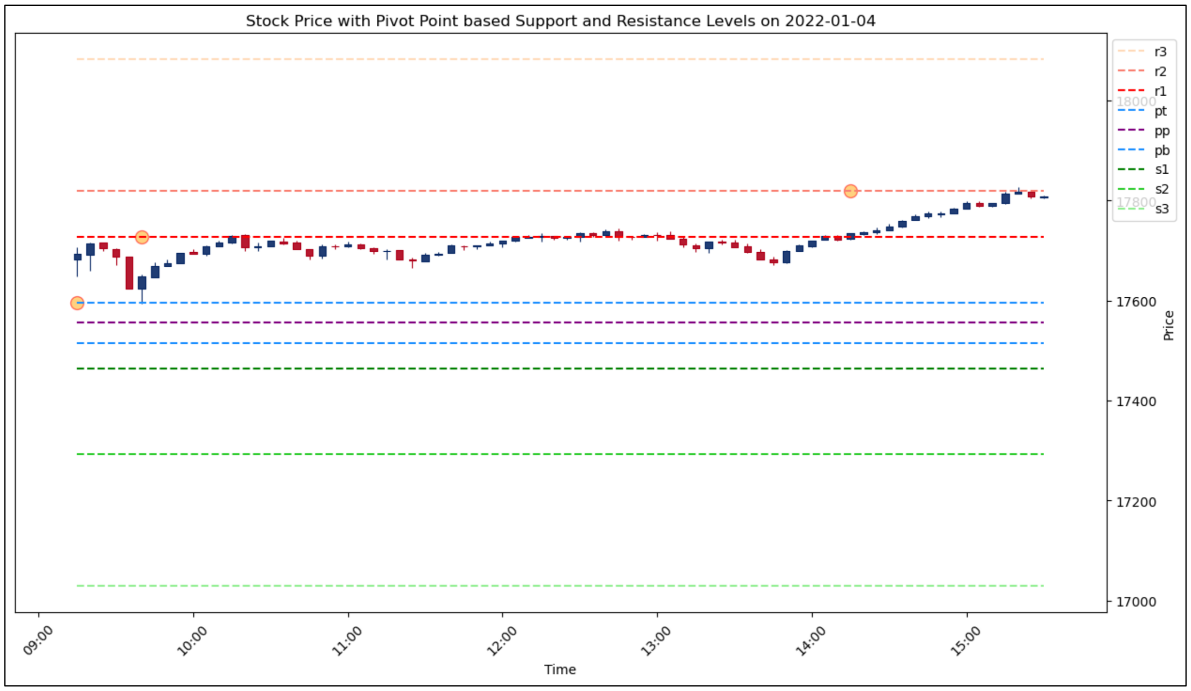


Figure 3.3.2 (f): Output Variable 'NextLevel' is marked along with candles and pivot levels

3.3.3 Model Development

As pointed out by (Fischer & Krauss, 2017), (Nirob & Hasan, 2023) and (Yan, 2016), the LSTM performs well with stock market data, hence it was chosen as the model in this research study. In this research, multiple experiments will be carried out with different sizes and numbers of hidden layers of LSTM to find the network structure better suited for our study.

3.3.3.1 Long Short–Term Memory (LSTM)

As perfectly explained by (Yan, 2016), LSTM is a type of artificial neural network with a memory unit called 'cell', which enables it to remember information from past data. This feature makes it suitable for time series data like stock market data where it needs to analyze continuous data from a time window.

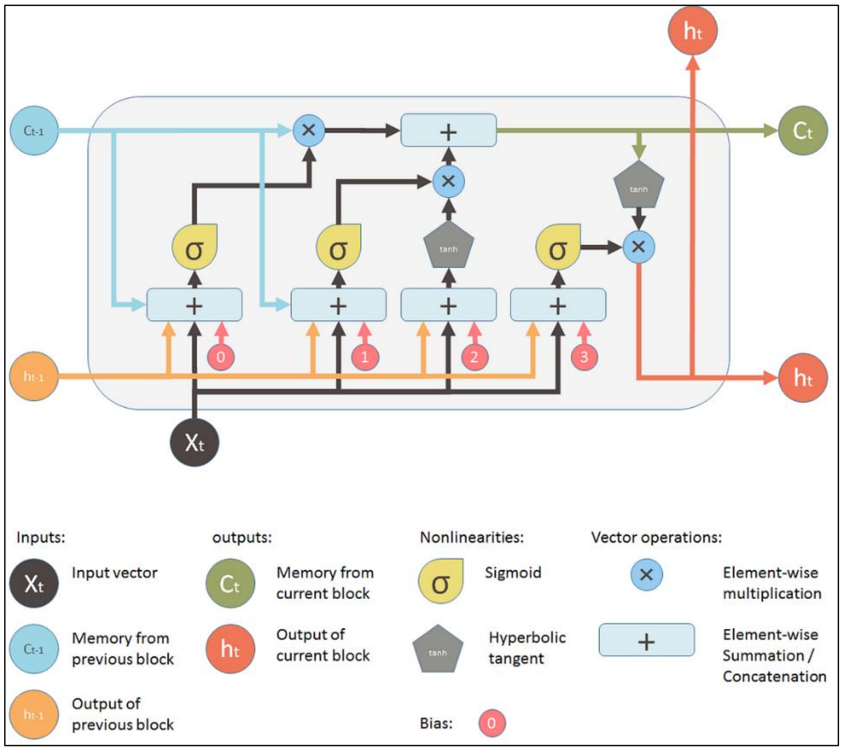


Figure 3.3.3.1: Architecture of LSTM. Source: (Yan, 2016)

In a nutshell, a single LSTM unit takes 3 inputs - the current input X_{t}, previous output h_{t-1} and previous memory C_{t-1} - and based on these, it alters its memory C_{t}, and generates a new output h_{t} (Yan, 2016).

3.3.3.2 Model Architecture

The model architecture consists of one or more layers of LSTMs and a fully connected Linear layer. The first LSTM layer takes in the input from the training data's input features. Based on the configurations there can be one or more layers of LSTM. Last LSTM's output is connected to the input of the fully connected Linear layer. The output of the Linear layer is mapped to the encoded classes of the output variable in the training data.

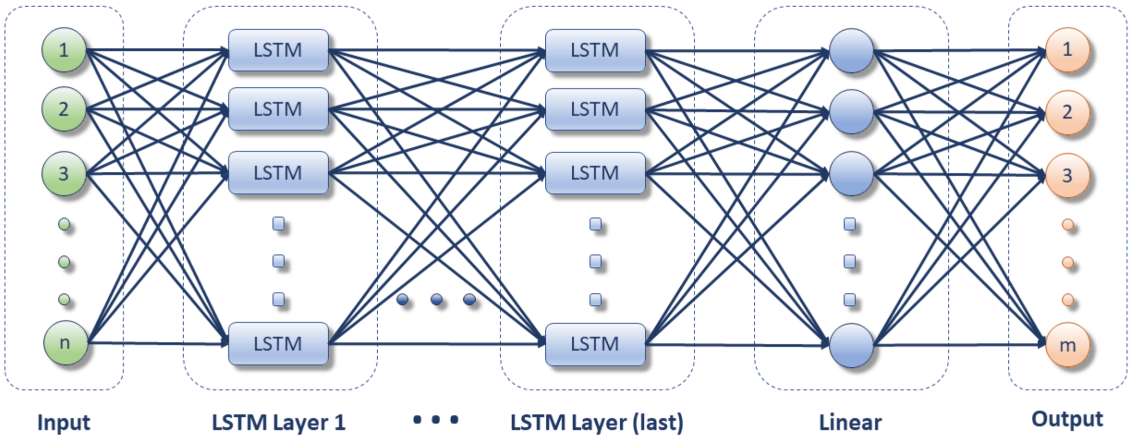


Figure 3.3.3.2 (a): Model Architecture

The model architecture consists of these:

1. Input

   This is the input data to the network with a size of 'inputSize'. To answer the research questions of this study, many experiments are done with varying combinations of input. So 'inputSize' changes based on the experiment configurations.

   All of the experiments have these mandatory input columns

   a. Open
   b. Low
   c. High
   d. Close

Based on the experiment configurations, the following columns could also be optionally added along with the above mandatory columns:

e. Volume
f. RSI
g. EMA7
h. EMA14
i. EMA_CROSSOVER_7AND14
j. Momentum
k. ATR
l. ADX, Plus DI and Minus DI
m. Williams' %R
n. Doji
o. Marubozu
p. Hammer
q. Hanging Man
r. Engulfing
s. Harami
t. Morning Star
u. Evening Star
v. 3 White Soldiers
w. 3 Black Crows

Please refer to Appendix A for the detailed descriptions of the above columns.

Before passing into the model, the input features are scaled using ‘StandardScaler’ from scikit-learn (Anon, n.d.b). This is necessary to avoid biasing because the different input features that are measured at different scales do not contribute equally towards model building (Loukas, 2020). It was also noted by Loukas that the StandardScaler removes the mean and scales each input feature to unit variance. According to him, this operation is carried out per feature and it is independent of other features.

As per (Anon, n.d.b), the standard score z is calculated as,

$z = (x - u) / s$

where

x is the input sample,

u is the mean of the training samples or 0 if with_mean=False, and

s is the standard deviation of the training samples or 1 if with_std=False.

2. LSTM layer(s)

There will be one or more layers of Long Short-Term Memory, depending on the configuration parameters. This is the layer responsible for learning the long-term dependencies and patterns in the training stock market data. Let’s take a look at the configuration parameters:

hiddenSize: determines the number of LSTM cells

numberOfLayers: determines the number of layers in the network

The larger the configuration parameters, the layer(s) could learn more complex patterns in the data but at the cost of higher probabilities of overfitting the data and more computational costs.

3. Linear Fully Connected Layer

   This layer will take the output of the last layer of the LSTM and perform a linear transformation of the same (Anon, n.d.e). This layer models the complex relationship between input and output data. According to (Anon, n.d.e), the linear transformation equation is given below:

   $y = xW + b$

   where,
   y is the output,
   x is the input,
   W is the weight matrix, which is a learnable parameter by this layer
   b is the bias, which is a learnable parameter by this layer

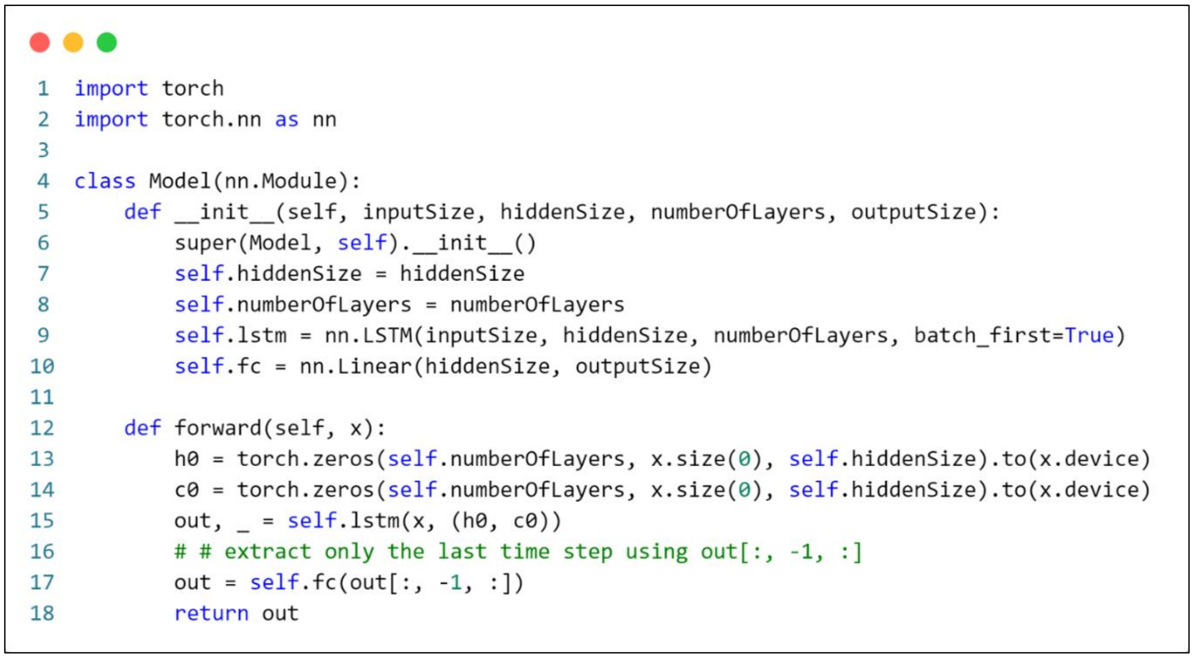
4. Output

   The output layer is the output of the fully connected layer and the dimension of this will be determined by the ‘outputSize’ parameter. This parameter will be equal to the number of unique classes under the ‘NextLevel’ column in the training data. In most cases, it would be 10 because for decent-sized training data, the classes under the ‘NextLevel’ column will be as follows:

['', 'pb', 'pp', 'pt', 'r1', 'r2', 'r3', 's1', 's2', 's3']

During the training phase, we use the category codes of ‘NextLevel’ to train the network, so this output layer will have numerical values starting from 0 to (outputSize-1).

The code to create such a model architecture is given below:



```python
import torch
import torch.nn as nn

class Model(nn.Module):
    def __init__(self, inputSize, hiddenSize, numberOfLayers, outputSize):
        super(Model, self).__init__()
        self.hiddenSize = hiddenSize
        self.numberOfLayers = numberOfLayers
        self.lstm = nn.LSTM(inputSize, hiddenSize, numberOfLayers, batch_first=True)
        self.fc = nn.Linear(hiddenSize, outputSize)

    def forward(self, x):
        h0 = torch.zeros(self.numberOfLayers, x.size(0), self.hiddenSize).to(x.device)
        c0 = torch.zeros(self.numberOfLayers, x.size(0), self.hiddenSize).to(x.device)
        out, _ = self.lstm(x, (h0, c0))
        # # extract only the last time step using out[:, -1, :]
        out = self.fc(out[:, -1, :])
        return out
```

Figure 3.3.3.2 (b): Model Class in Python

3.3.3.3 Model Training

Please refer to Appendix D, for the code to train the model.

3.3.4 Model Evaluation

After training the model, we need metrics to evaluate it. The common metrics like accuracy are good in the theoretical evaluation of the model, but since the research problem is related to the stock mark, it would be better to come up with some other metrics which can gauge the practicality as well.

In this research, we use the following 2 metrics to evaluate the trained models.

3.3.4.1 Prediction Accuracy

To compute the prediction accuracy, the accuracy_score() function from scikit-learn (Anon, n.d.g) is used. As per their documentation, this function compares the values from the validation dataset and the predicted values by the model and returns the results as a fraction, by default.

The signature of the function is as follows (Anon, n.d.g):

sklearn.metrics.accuracy_score(y_true, y_pred, *, normalize=True, sample_weight=None)

where,

y_true is the output column in the validation dataset (ground truth labels)

y_pred is the predicted labels

normalize is optional. If set to true, return the fraction of correctly classified labels. If set to false, it returns the count of the correctly classified labels. Default is True sample_weight is the optional sample weight. Defaults to None

For example, let's assume that we have a validation dataset with 100 rows. This dataset will contain all the input columns that our training dataset had, and also the output column 'NextLevel'. This output column 'NextLevel' is y_true. We will pass all the input features of all 100 rows to our trained model and it predicts output for all those rows – this is y_pred. Let us say, our model predicted 90 labels exactly the same as in the ground truth y_true. Then the accuracy_score() function will return the fraction of correctly classified labels as 0.9.

(Anon, n.d.a) provides the mathematical equation of accuracy, as indicated below,

$$\boxed{\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)}$$

Figure 3.3.4.1: Mathematical equation for calculating accuracy score. Source: (Anon, n.d.a)

In the equation above, 1(x) is the indicator function that maps elements of the subset to one and all other elements to zero (Anon, n.d.g).

3.3.4.2 Prediction Usefulness

This is a custom metric that is developed to evaluate the practicality of the model in real-world scenarios. This is limited to the current research model, as it uses the reverse concepts of marking 'NextLevel' during the data preprocessing step. The fundamental concept of this metric revolves around verifying if the predicted level is ever touched by the future candles during that trading session – in the case of day trading, that would be the market closing time on the current day. This is calculated as the percentage of useful predictions out of the total number of predictions. The mathematical equation of this metric is outlined below,

Prediction Usefulness = (useful predictions / total predictions) x 100

where,

'useful predictions' are the number of predictions where the predicted levels are touched by future candles in that trading session, and

'total predictions' are the total number of predictions done for the complete validation dataset. This includes useful as well as useless predictions

The following figure shows an example of useful predictions.

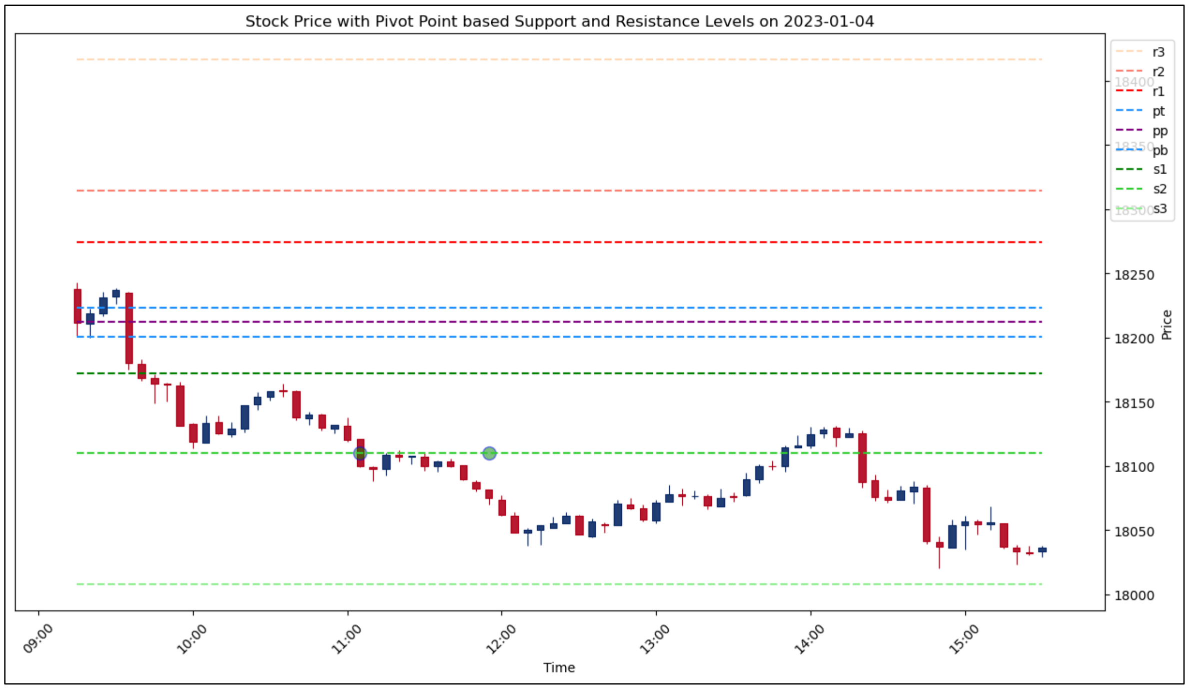


Figure 3.3.4.2 (a): Example of useful predictions

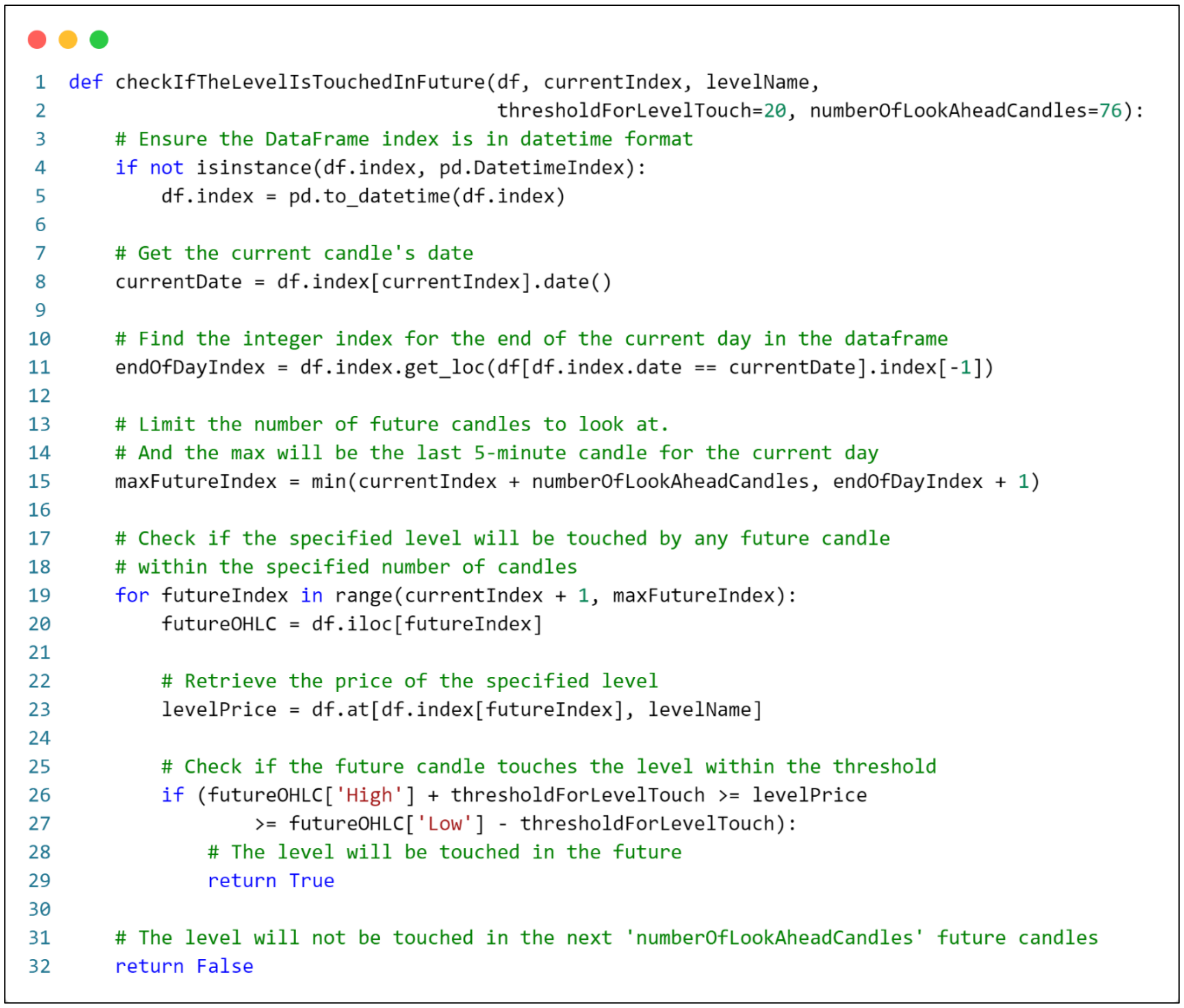
The green dot on level s2 at 11:55 depicts a prediction that level s2 will be touched in future. It can be noted that at time 13:50, a candle touched the predicted level s2. So this is a useful prediction.

Also, there is one more prediction in this diagram on s2 at 11:05. And this predicted level s2 is touched by a future candle at 11:20. So this is another useful prediction.

If the prediction usefulness metric is applied just to this day, it will be calculated as follows,

Prediction Usefulness = (2 / 2) x 100 = 100%

Python code is given below to gain insight into the primary function of this metric,



```python
def checkIfTheLevelIsTouchedInFuture(df, currentIndex, levelName,
                                     thresholdForLevelTouch=20, numberOfLookAheadCandles=76):
    # Ensure the DataFrame index is in datetime format
    if not isinstance(df.index, pd.DatetimeIndex):
        df.index = pd.to_datetime(df.index)

    # Get the current candle's date
    currentDate = df.index[currentIndex].date()

    # Find the integer index for the end of the current day in the dataframe
    endOfDayIndex = df.index.get_loc(df[df.index.date == currentDate].index[-1])

    # Limit the number of future candles to look at.
    # And the max will be the last 5-minute candle for the current day
    maxFutureIndex = min(currentIndex + numberOfLookAheadCandles, endOfDayIndex + 1)

    # Check if the specified level will be touched by any future candle
    # within the specified number of candles
    for futureIndex in range(currentIndex + 1, maxFutureIndex):
        futureOHLC = df.iloc[futureIndex]

        # Retrieve the price of the specified level
        levelPrice = df.at[df.index[futureIndex], levelName]

        # Check if the future candle touches the level within the threshold
        if (futureOHLC['High'] + thresholdForLevelTouch >= levelPrice
                >= futureOHLC['Low'] - thresholdForLevelTouch):
            # The level will be touched in the future
            return True

    # The level will not be touched in the next 'numberOfLookAheadCandles' future candles
    return False
```

Figure 3.3.4.2 (b): Python code to verify if a predicted level will be touched by future candles

3.3.4.3 Model Testing

Please refer to Appendix E for the model evaluation code.

3.3.4.4 The Need for Visual Analysis with Evaluation Metrics

Quantitative metrics like Prediction Accuracy and Prediction Usefulness give a fair amount of insights about the quality of the trained model, yet it’s always beneficial to visually analyse the Evaluation Results to understand the complex price movement patterns to better evaluate a model. Let’s assume there are 2 different models with the same Prediction Accuracy & Usefulness scores, consider the following scenarios to understand the need for visual analysis better.

Scenario 1: The first model predicts one level and then the price moves in that direction though not touching the predicted level. The second model predicts the same level, but the price moves in the opposite direction to hit the stop-loss order. In the case of the former model, at least at the end of the day, we can square off with a small profit or small loss.

Scenario 2: First one model makes only 2 predictions in a month, and the other model could be predicting 2-3 predictions each day of the month. Then it might make more sense to go with the second model for more trade opportunities and practicality.

In summary, though the visual analysis can be very subjective, it’s still useful in assessing the effectiveness of the model in real-world scenarios.

3.4 Research Design Limitations

It is essential to acknowledge that the research design for this study has certain limitations that must be taken into consideration when interpreting the results.

The research design uses only selected technical indicators and candle patterns from countless such indicators and candle patterns available. Taking all of them into consideration while developing a model is next to impossible, so this research design has been limited to using only a few from the most popular categories of technical indicators and candlesticks.

Also, the fundamental concept to mark the output variable ‘NextLevel’ in the dataset could vary depending on the use case and design philosophies. In this study, it is designed to take the farthest level in a limited window of future candles, and all the consecutive same-level predictions are omitted to keep the dataset more learnable to the model. It’s quite possible to define the concept in a completely different way, according to different use cases and design philosophies.

The research design outlined 2 metrics to evaluate the trained model. There could be more such metrics to evaluate the same. Also, it is to be noted that the Prediction Usefulness metric considers any predicted levels reached by future candles on that particular trading session as ‘useful’. But in more complex practical scenarios, it could be refined to filter out movements in the opposite directions before touching the predicted levels.

This study was conducted across different types of stock assets like indices and individual stocks, hence ‘volume’ and other volume-related technical indicators were not considered, to make it consistent across studies. Because volume will be zero for indices, whereas individual stocks will be having it. In practice, volume and other volume-related technical indicators could be used for model training exclusively for individual stocks.

3.5 Conclusion

This chapter outlined the research problem, research purpose and research design. By using an LSTM-based model architecture and a preprocessed dataset with additional columns, this study will be able to train a model to predict the pivot levels that will be touched by future candlesticks in any given session of trading. This chapter also provided the metrics to be used to evaluate the model for accuracy and practical usefulness.

CHAPTER IV:
RESULTS

4.1 Forecasting Future Candle Pivot Levels with LSTM System

This section answers the research question “Can an LSTM-based system be trained to map the price movements along with technical indicators into pivot point-based support/resistance levels?”. Training an LSTM system for forecasting the pivot levels that the future candles are going to touch, yielded positive results. Findings from the training of one of such models are given below for showcasing the results.

4.1.1 Configuration of the LSTM system

The training configuration is given below in the tabular format.

Table 4.1.1: Training Configuration of the LSTM system

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this threshold value will only be considered

4.1.2 Evaluation Results

The results of training the model using the above configuration are given below in the tabular format.

Table 4.1.2: Evaluation Results of the LSTM system

Evaluation Metric	Result	Comments
Prediction Accuracy	92.04%	Accuracy in predicting ‘NextLevel’ for the testing period
Prediction Usefulness	75.00%	Indicates the percentage of useful predictions

4.1.3 Visual Representation of Positive Results

Some of the positive outcomes of the model trained using the above parameters for the Nifty Index are given below.

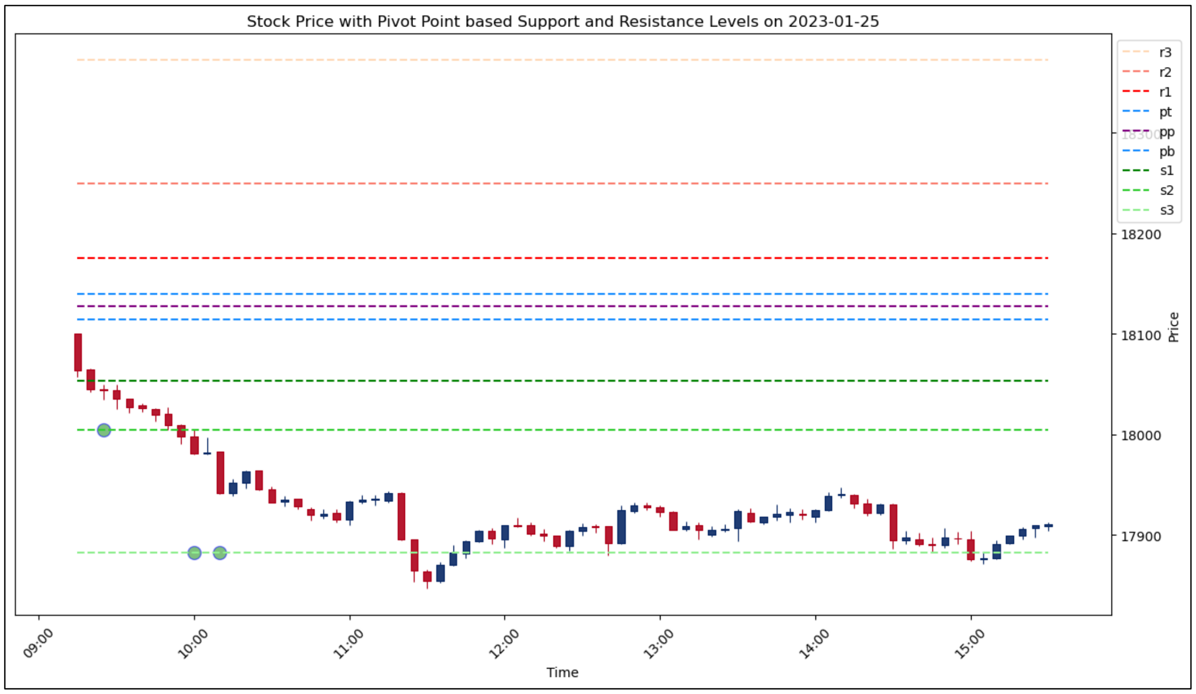


Figure 4.1.3 (a): Positive Prediction Results of Nifty Index – Result 1

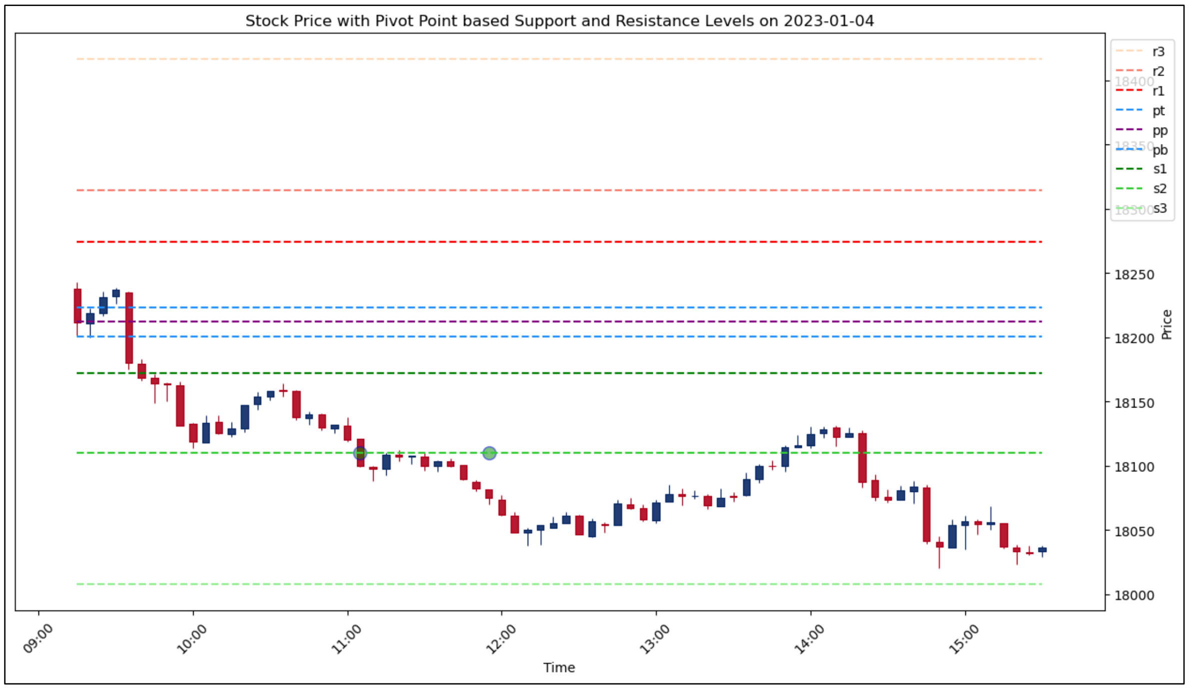


Figure 4.1.3 (b): Positive Prediction Results of Nifty Index – Result 2

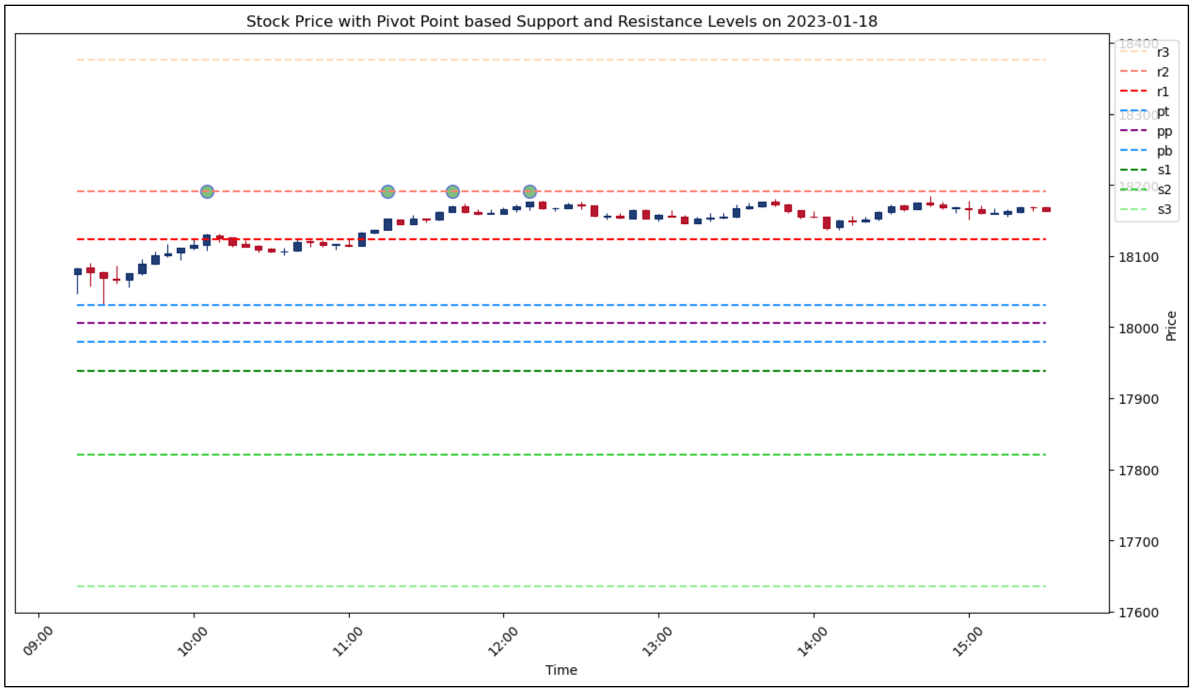


Figure 4.1.3 (c): Positive Prediction Results of Nifty Index – Result 3

4.1.4 Visual Representation of Negative Results

Some of the unsuccessful predictions of the model are given below. However, in the discussion section, we will see why these are not trade signals and how to filter out these to minimize losing trades.

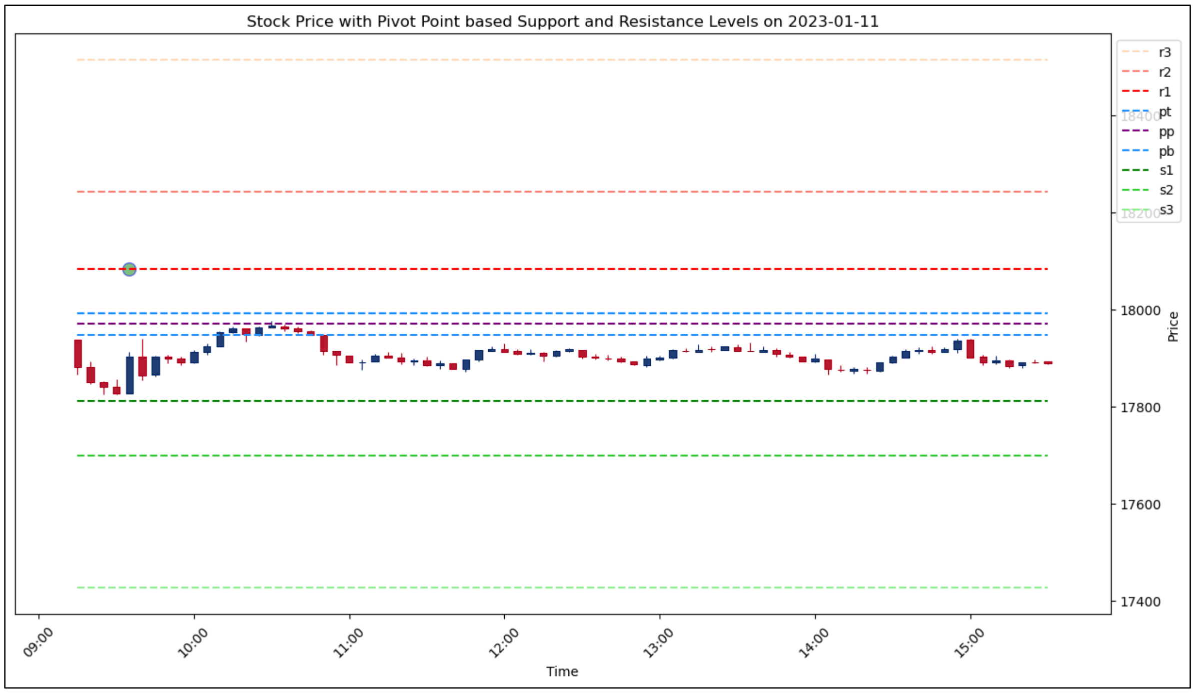


Figure 4.1.4 (a): Negative Prediction Results of Nifty Index – Result 1

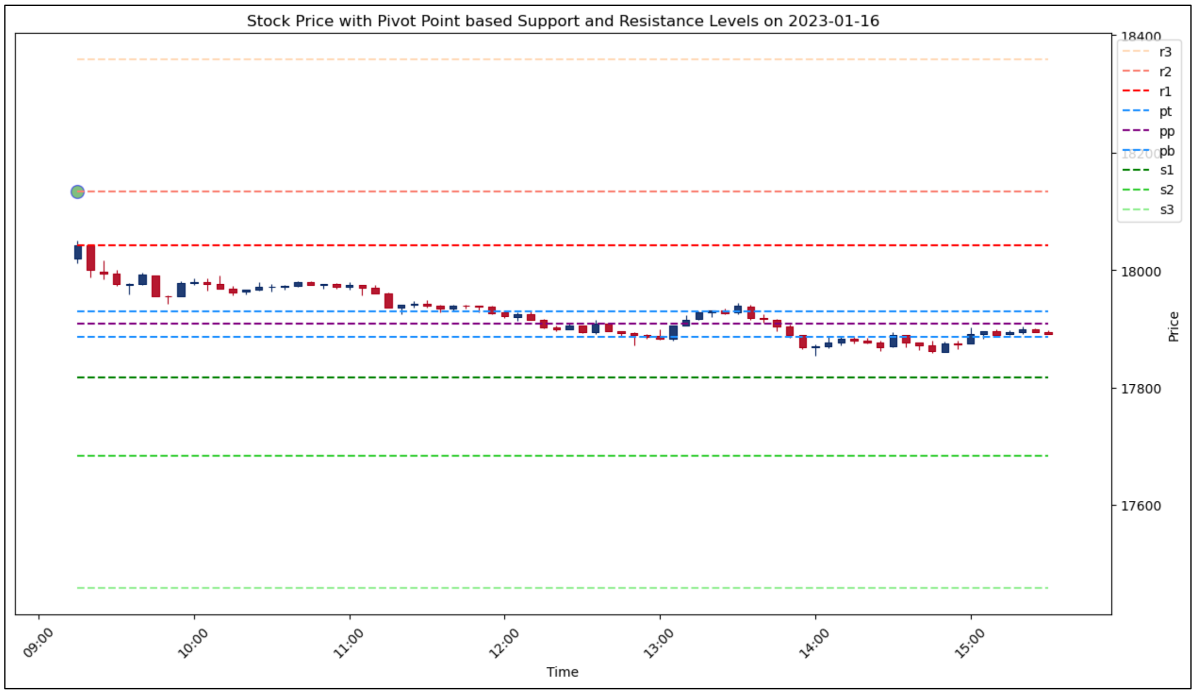


Figure 4.1.4 (b): Negative Prediction Results of Nifty Index – Result 2

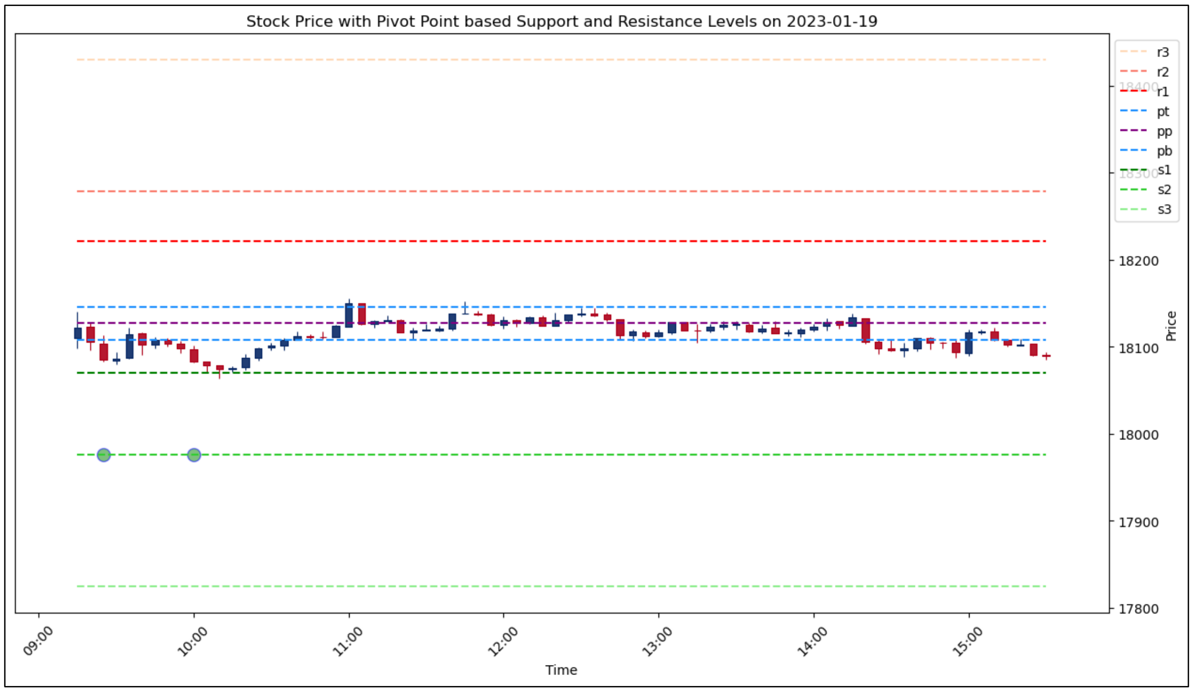


Figure 4.1.4 (c): Negative Prediction Results of Nifty Index – Result 3

4.2 Mapping Pivot Levels into a Dataset for training LSTM system

This section answers the research question "Since the pivot point-based support/resistance levels change every day, how do we map these ever-changing levels into a dataset that the LSTM can learn from?". In trading, pivot levels can vary based on the chosen time frame, whether it's for day trading, swing trading or positional trading, because it depends only on the previous timeframe's high, low and close values. In day trading, pivot levels change every day. So it's challenging to map these values to a target variable column in a dataset so that an artificial intelligence-based model can be trained using that dataset.

This study attempts to solve this challenge in the following way.

1. It calculates pivot points for every day in the dataset and adds those values to the dataset under 9 columns - 'pp', 'pb', 'pt', 'r1', 's1', 'r2', 's2', 'r3', 's3'
2. Uses the markNextLevel function specified in Appendix C to detect which pivot levels are going to be touched by the future candles for a specific look ahead window and returns the ‘name’ of the farthest level from the current candle
3. Creates a new column ‘NextLevel’ which will have the name of the above columns or an empty string when there are no future levels detected

This study demonstrates the process of mapping the 9 pivot levels to a single target column in a dataset, facilitating the training of an artificial intelligence model with this column as the target variable.

4.3 Optimal Technical Indicators and Candle Patterns for LSTM

This section answers the research question “What are the technical indicators and candlestick patterns that help in getting the LSTM-based system to learn better?”. To find the list of suitable technical indicators and candle patterns for training the LSTM-based model proposed in this study, many experiments with different configurations were carried out. The summary of experiments is outlined below:

4.3.1 Identifying Optimal Technical Indicators and Candle Patterns

In this set of experiments, we will individually try out different technical indicators along with candle values across a standard configuration.

4.3.1.1 Experiment 1 - RSI

Table 4.3.1.1 (a): Training Configuration – RSI

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.1 (b): Evaluation Results – RSI

Evaluation Metric	Result	Comments
Prediction Accuracy	92.70%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	72.73%	Indicates the % of useful predictions

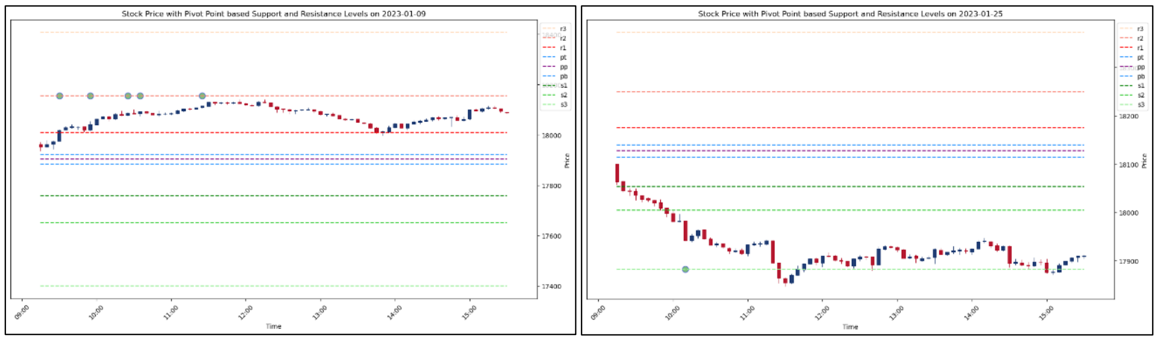


Figure 4.3.1.1: Sample predictions for visual analysis – RSI

4.3.1.2 Experiment 2 – EMA7

Table 4.3.1.2 (a): Training Configuration – EMA7

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close EMA7	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.2 (b): Evaluation Results – EMA7

Evaluation Metric	Result	Comments
Prediction Accuracy	93.22%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	00.00%	Indicates the % of useful predictions

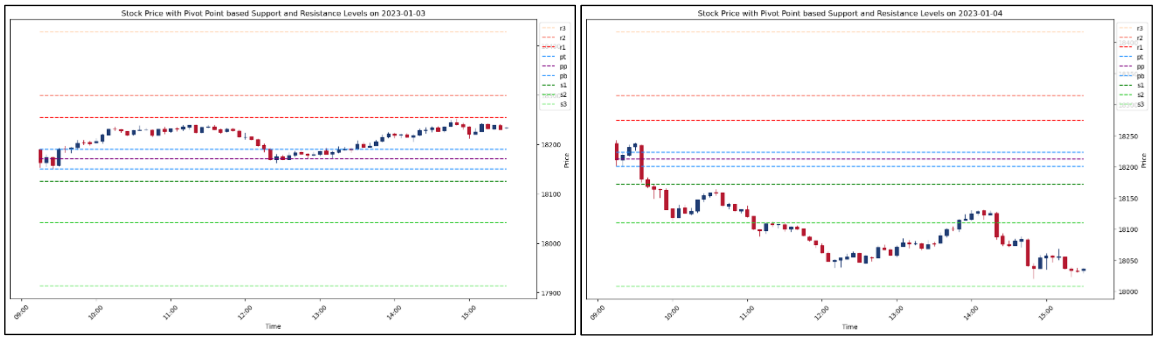


Figure 4.3.1.2: Sample predictions for visual analysis – EMA7

4.3.1.3 Experiment 3 – EMA14

Table 4.3.1.3 (a): Training Configuration – EMA14

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close EMA14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.3 (b): Evaluation Results – EMA14

Evaluation Metric	Result	Comments
Prediction Accuracy	93.09%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	66.67%	Indicates the % of useful predictions

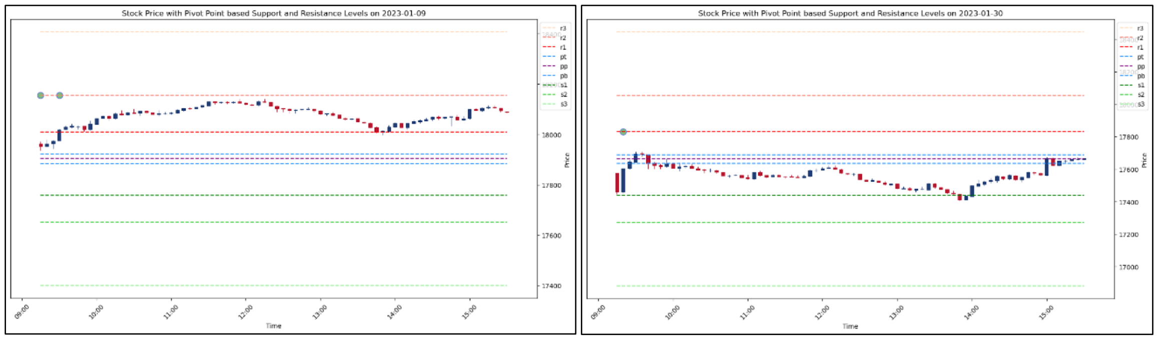


Figure 4.3.1.3: Sample predictions for visual analysis – EMA14

4.3.1.4 Experiment 4 - EMA_CROSSOVER_7AND14

Table 4.3.1.4 (a): Training Configuration – EMA_CROSSOVER_7AND14

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.4 (b): Evaluation Results – EMA_CROSSOVER_7AND14

Evaluation Metric	Result	Comments
Prediction Accuracy	93.09%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	50.00%	Indicates the % of useful predictions

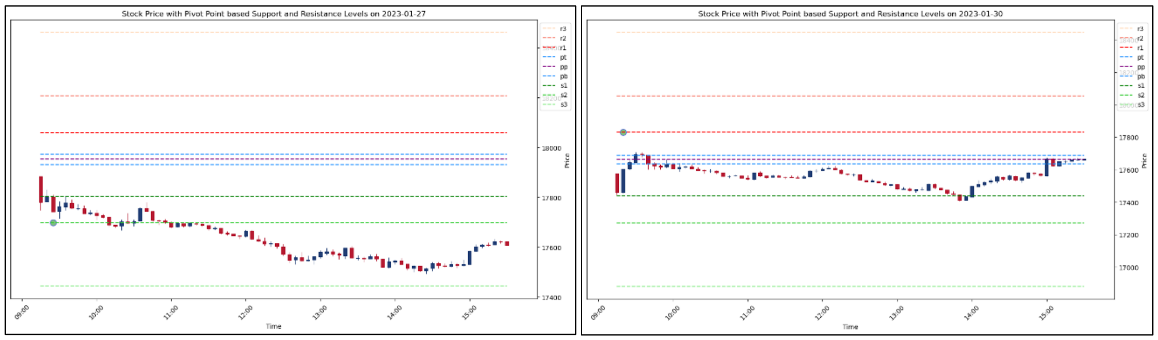


Figure 4.3.1.4: Sample predictions for visual analysis – EMA_CROSSOVER_7AND14

4.3.1.5 Experiment 5 - MOM

Table 4.3.1.5 (a): Training Configuration – MOM

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close MOM	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.5 (b): Evaluation Results – MOM

Evaluation Metric	Result	Comments
Prediction Accuracy	92.76%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	27.27%	Indicates the % of useful predictions

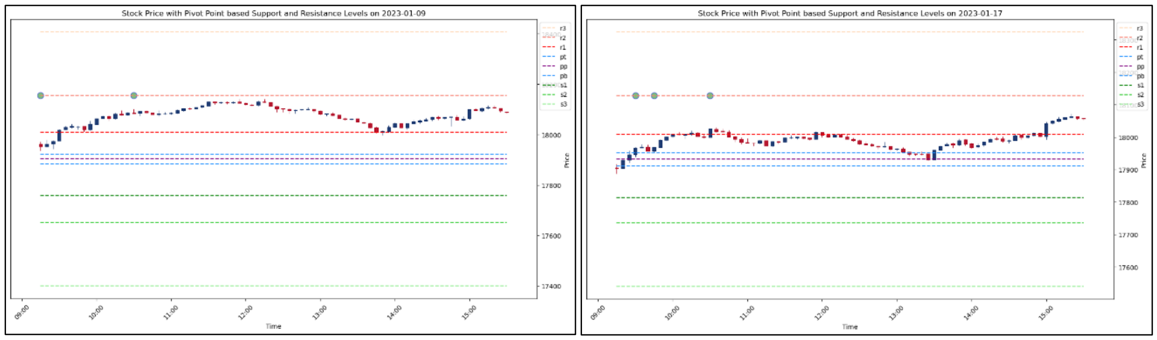


Figure 4.3.1.5: Sample predictions for visual analysis – MOM

4.3.1.6 Experiment 6 - ATR

Table 4.3.1.6 (a): Training Configuration – ATR

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close ATR	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.6 (b): Evaluation Results – ATR

Evaluation Metric	Result	Comments
Prediction Accuracy	93.16%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	50.00%	Indicates the % of useful predictions

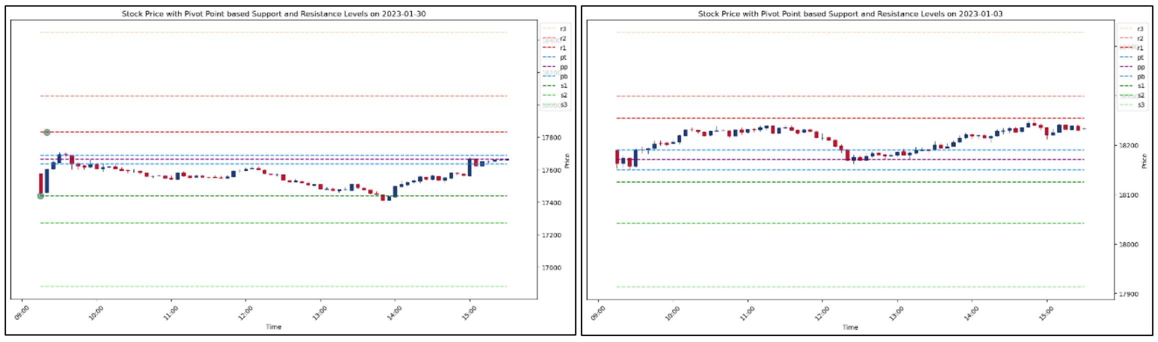


Figure 4.3.1.6: Sample predictions for visual analysis – ATR

4.3.1.7 Experiment 7 - ADX, MINUS_DI & PLUS_DI

Table 4.3.1.7 (a): Training Configuration – ADX, MINUS_DI & PLUS_DI

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open, High Low Close ADX, MINUS_DI, PLUS_DI	List of input Columns/Features taken for training the model. ADX is generally used in combinations with Plus and Minus DIs to get direction
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.7 (b): Evaluation Results – ADX, MINUS_DI & PLUS_DI

Evaluation Metric	Result	Comments
Prediction Accuracy	86.71%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	38.46%	Indicates the % of useful predictions

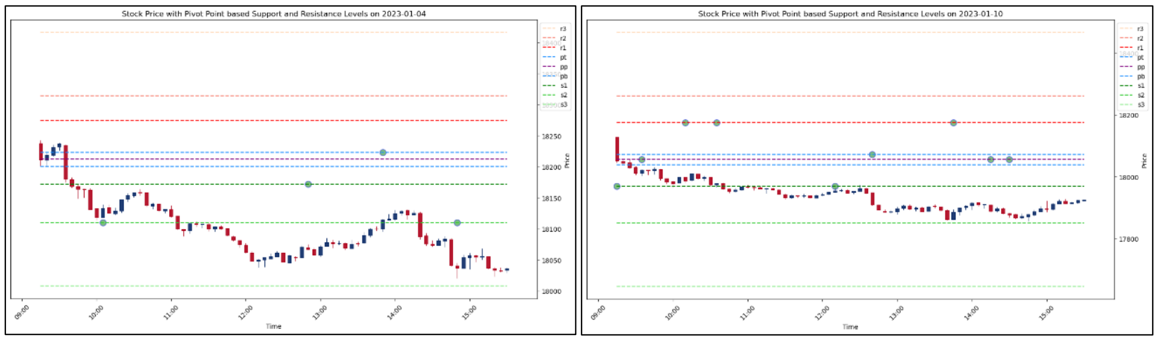


Figure 4.3.1.7: Sample predictions for visual analysis – ADX, MINUS_DI & PLUS_DI

4.3.1.8 Experiment 8 - WILLR

Table 4.3.1.8 (a): Training Configuration – WILLR

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close WILLR	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.8 (b): Evaluation Results – WILLR

Evaluation Metric	Result	Comments
Prediction Accuracy	93.16%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	0.00%	Indicates the % of useful predictions

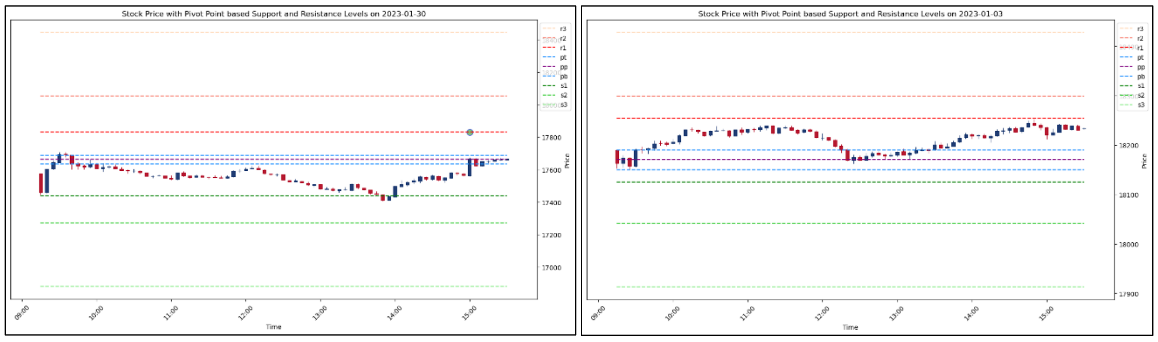


Figure 4.3.1.8: Sample predictions for visual analysis – WILLR

4.3.1.9 Experiment 9 - CDLMARUBOZU

Table 4.3.1.9 (a): Training Configuration – CDLMARUBOZU

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLMARUBOZU	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.9 (b): Evaluation Results – CDLMARUBOZU

Evaluation Metric	Result	Comments
Prediction Accuracy	93.22%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	00.00%	Indicates the % of useful predictions

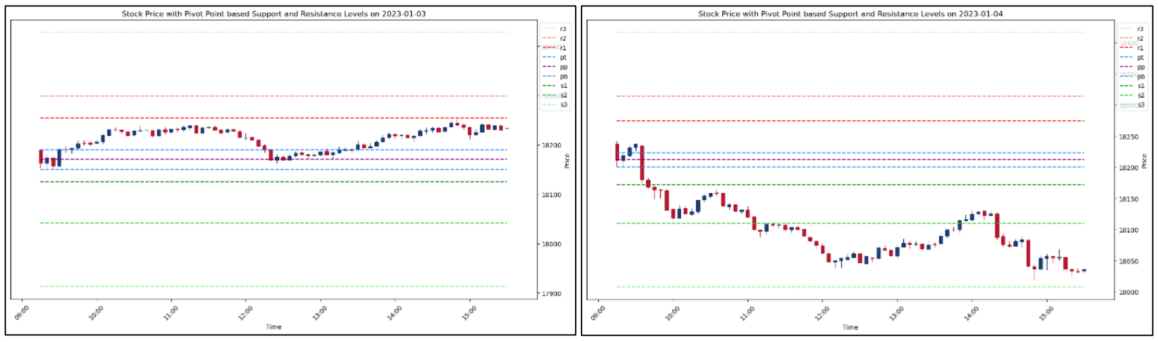


Figure 4.3.1.9: Sample predictions for visual analysis – CDLMARUBOZU

4.3.1.10 Experiment 10 - CDLDOJI

Table 4.3.1.10 (a): Training Configuration – CDLDOJI

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLDOJI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.10 (b): Evaluation Results – CDLDOJI

Evaluation Metric	Result	Comments
Prediction Accuracy	93.22%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	00.00%	Indicates the % of useful predictions

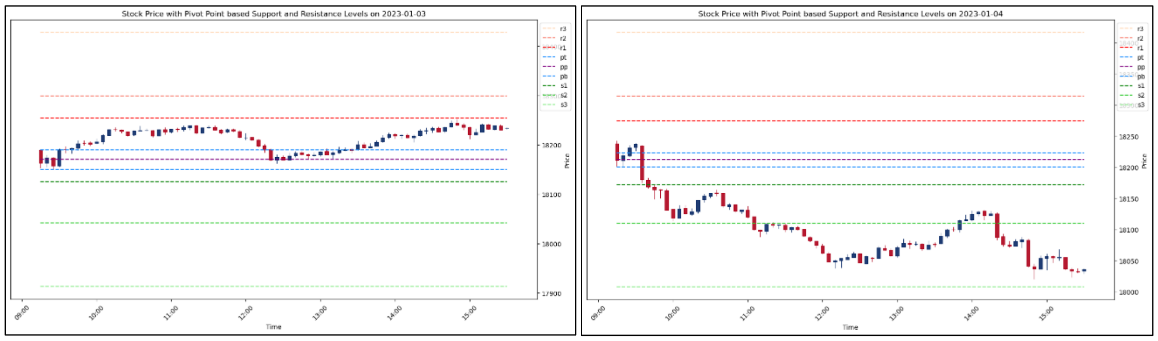


Figure 4.3.1.10: Sample predictions for visual analysis – CDLDOJI

4.3.1.11 Experiment 11 - CDLHAMMER

Table 4.3.1.11 (a): Training Configuration – CDLHAMMER

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLHAMMER	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.11 (b): Evaluation Results – CDLHAMMER

Evaluation Metric	Result	Comments
Prediction Accuracy	92.24%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	00.00%	Indicates the % of useful predictions

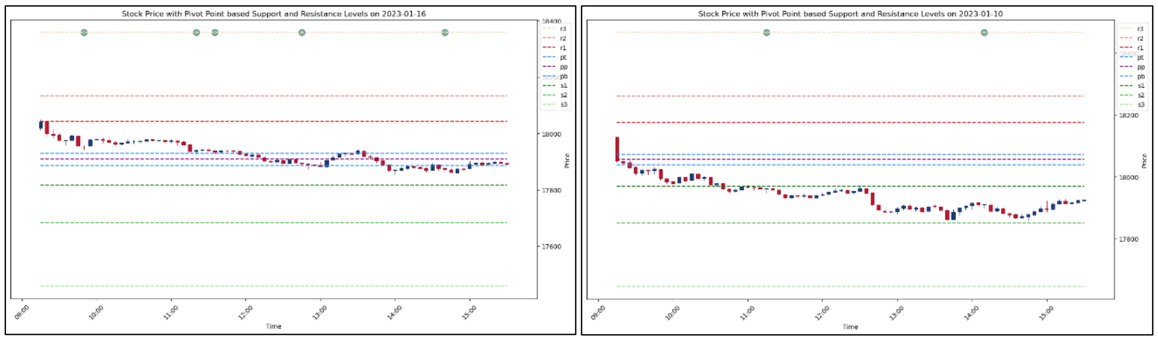


Figure 4.3.1.11: Sample predictions for visual analysis – CDLHAMMER

4.3.1.12 Experiment 12 - CDLHANGINGMAN

Table 4.3.1.12 (a): Training Configuration – CDLHANGINGMAN

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLHANGINGMAN	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.12 (b): Evaluation Results – CDLHANGINGMAN

Evaluation Metric	Result	Comments
Prediction Accuracy	93.22%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	00.00%	Indicates the % of useful predictions

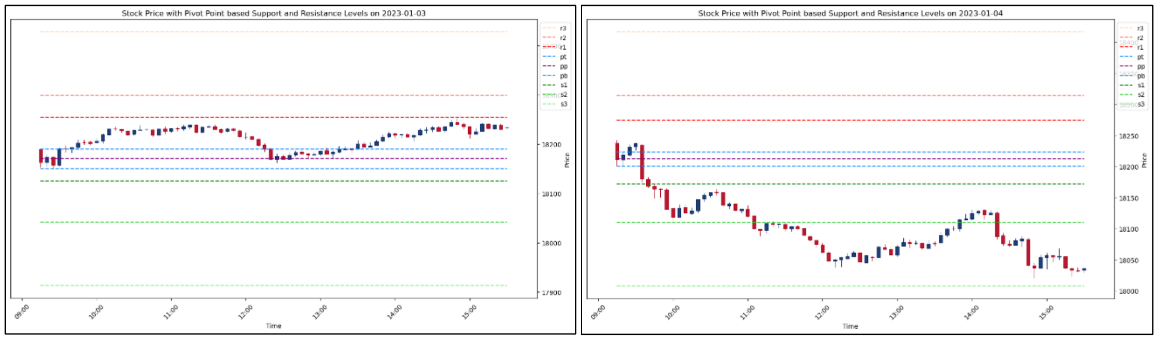


Figure 4.3.1.12: Sample predictions for visual analysis – CDLHANGINGMAN

4.3.1.13 Experiment 13 - CDLENGULFING

Table 4.3.1.13 (a): Training Configuration – CDLENGULFING

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLENGULFING	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.13 (b): Evaluation Results – CDLENGULFING

Evaluation Metric	Result	Comments
Prediction Accuracy	93.16%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	00.00%	Indicates the % of useful predictions

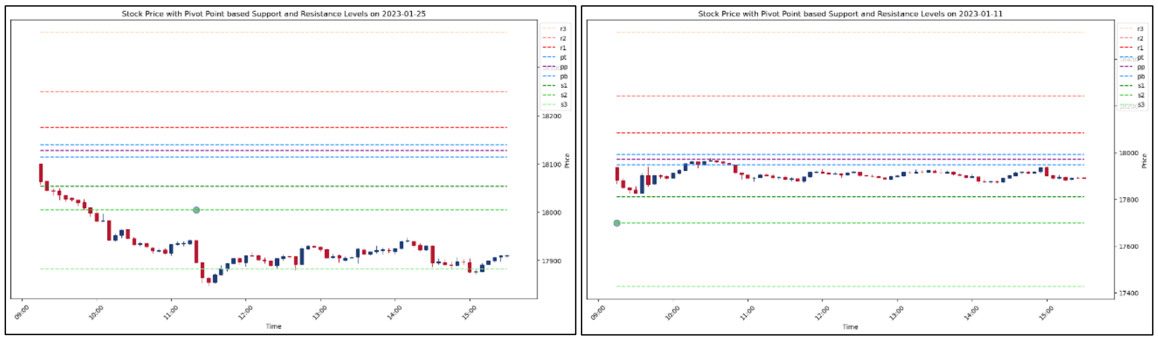


Figure 4.3.1.13: Sample predictions for visual analysis – CDLENGULFING

4.3.1.14 Experiment 14 - CDLHARAMI

Table 4.3.1.14 (a): Training Configuration – CDLHARAMI

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLHARAMI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.14 (b): Evaluation Results – CDLHARAMI

Evaluation Metric	Result	Comments
Prediction Accuracy	92.24%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	00.00%	Indicates the % of useful predictions

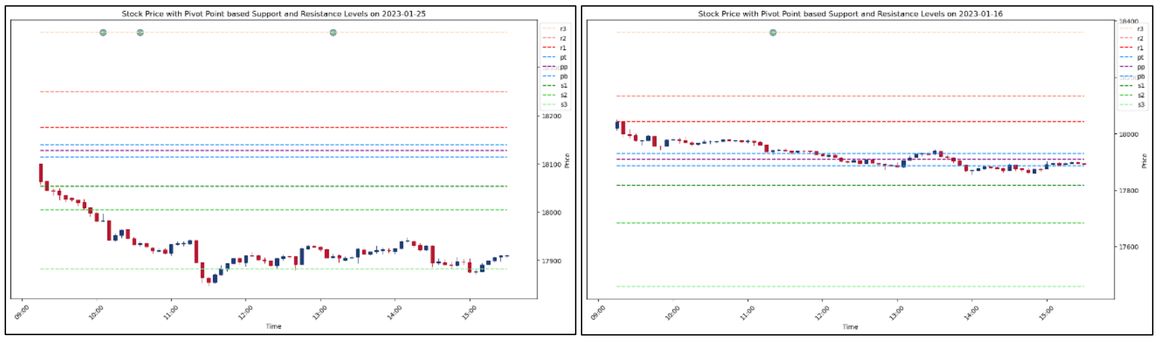


Figure 4.3.1.14: Sample predictions for visual analysis – CDLHARAMI

4.3.1.15 Experiment 15 - CDLMORNINGSTAR

Table 4.3.1.15 (a): Training Configuration – CDLMORNINGSTAR

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLMORNINGSTAR	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.15 (b): Evaluation Results – CDLMORNINGSTAR

Evaluation Metric	Result	Comments
Prediction Accuracy	93.09%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	50.00%	Indicates the % of useful predictions

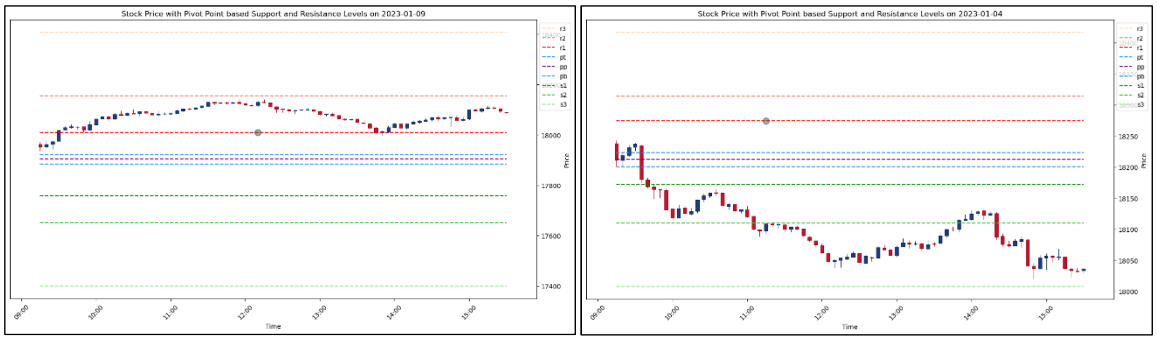


Figure 4.3.1.15: Sample predictions for visual analysis – CDLMORNINGSTAR

4.3.1.16 Experiment 16 - CDLEVENINGSTAR

Table 4.3.1.16 (a): Training Configuration – CDLEVENINGSTAR

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDLEVENINGSTAR	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.16 (b): Evaluation Results – CDLEVENINGSTAR

Evaluation Metric	Result	Comments
Prediction Accuracy	92.96%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	50.00%	Indicates the % of useful predictions

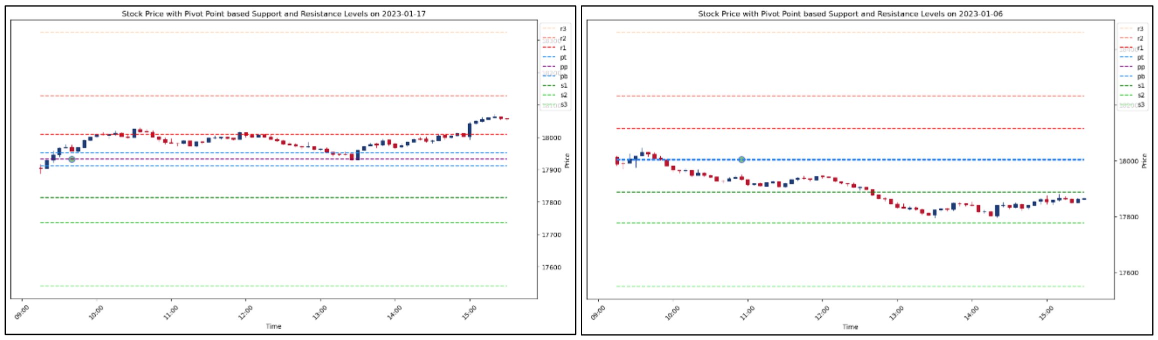


Figure 4.3.1.16: Sample predictions for visual analysis – CDLEVENINGSTAR

4.3.1.17 Experiment 17 - CDL3WHITESOLDIERS

Table 4.3.1.17 (a): Training Configuration – CDL3WHITESOLDIERS

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDL3WHITESOLDIERS	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.17 (b): Evaluation Results – CDL3WHITESOLDIERS

Evaluation Metric	Result	Comments
Prediction Accuracy	93.16%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	00.00%	Indicates the % of useful predictions

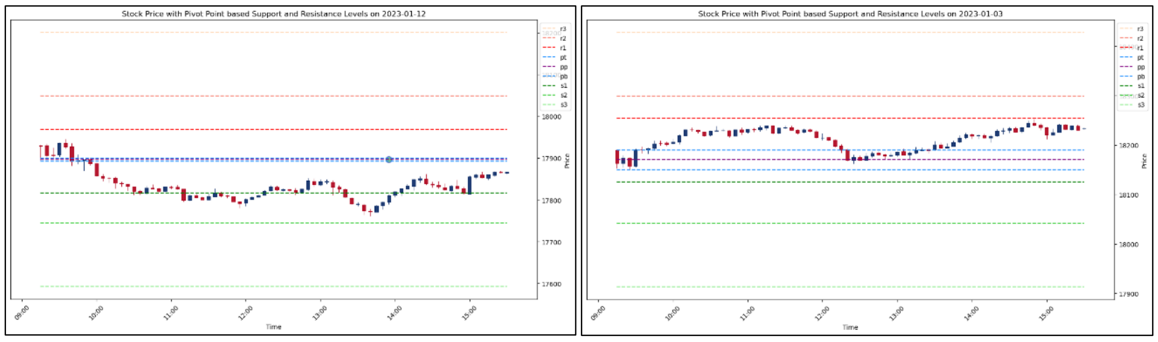


Figure 4.3.1.17: Sample predictions for visual analysis – CDL3WHITESOLDIERS

4.3.1.18 Experiment 18 - CDL3BLACKCROWS

Table 4.3.1.18 (a): Training Configuration – CDL3BLACKCROWS

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close CDL3BLACKCROWS	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.1.18 (b): Evaluation Results – CDL3BLACKCROWS

Evaluation Metric	Result	Comments
Prediction Accuracy	93.22%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	00.00%	Indicates the % of useful predictions

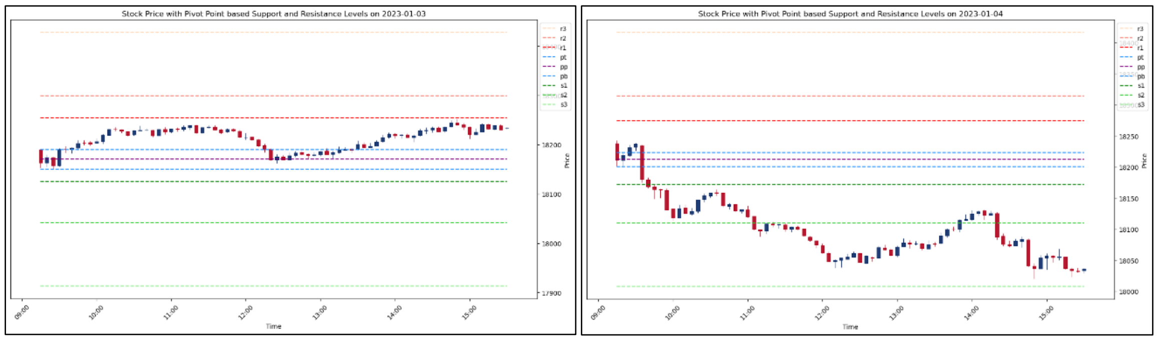


Figure 4.3.1.18: Sample predictions for visual analysis – CDL3BLACKCROWS

4.3.1.19 Summary

From the above set of experiments to pit all these 18 technical indicators and candle patterns against each other when taken individually, it is clear that RSI, by far, is the best one for training the proposed LSTM-based system along with the candle values of Open, High, Low, Close values. So RSI will be used in the subsequent experiments as a benchmark, along with candle values.

4.3.2 Experiments to find out Optimal Number of Epochs

In this set of experiments, we will keep the other configuration parameters constant and try different values for epochs such as 500, 1000, 2000 and 5000.

4.3.2.1 Experiment 1 – 500 Epochs

Table 4.3.2.1 (a): Training Configuration – 500 Epochs

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	500	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.2.1 (b): Evaluation Results - 500 Epochs

Evaluation Metric	Result	Comments
Prediction Accuracy	92.83%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	66.67%	Indicates the % of useful predictions

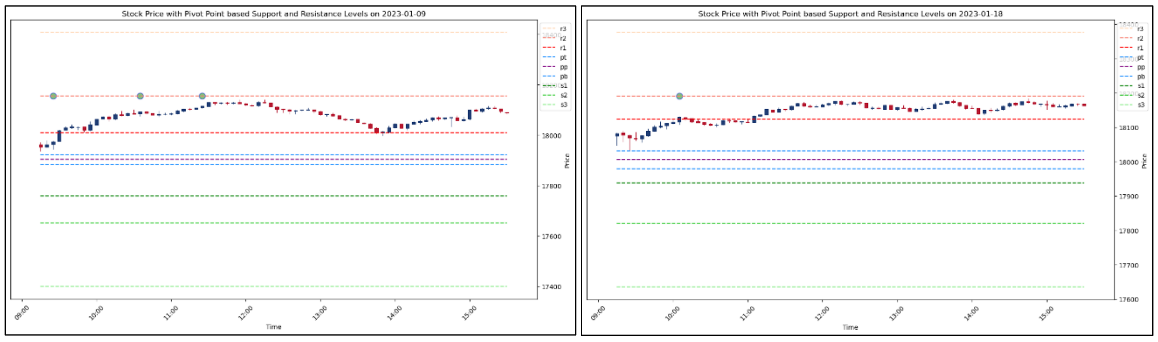


Figure 4.3.2.1: Sample predictions for visual analysis – 500 Epochs

4.3.2.2 Experiment 2 – 1000 Epochs

Table 4.3.2.2 (a): Training Configuration – 1000 Epochs

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.2.2 (b): Evaluation Results – 1000 Epochs

Evaluation Metric	Result	Comments
Prediction Accuracy	92.70%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	72.73%	Indicates the % of useful predictions

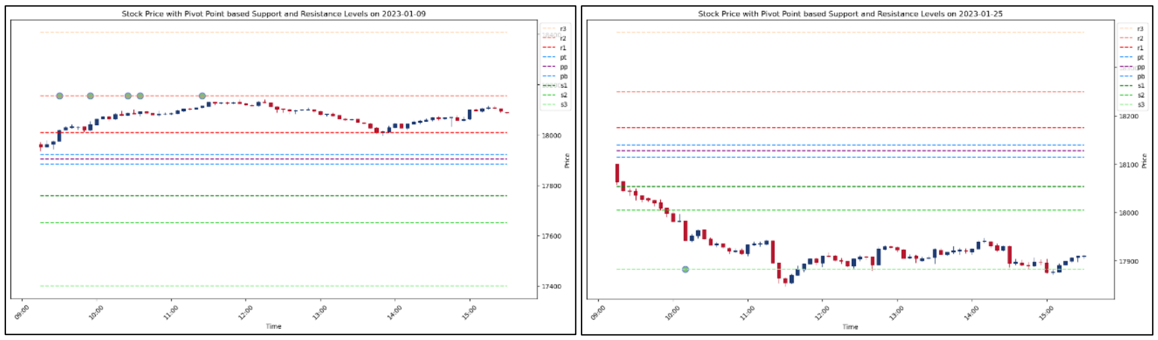


Figure 4.3.2.2: Sample predictions for visual analysis – 1000 Epochs

4.3.2.3 Experiment 3 – 2000 Epochs

Table 4.3.2.3 (a): Training Configuration – 2000 Epochs

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	2000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.2.3 (b): Evaluation Results – 2000 Epochs

Evaluation Metric	Result	Comments
Prediction Accuracy	91.78%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	29.63%	Indicates the % of useful predictions

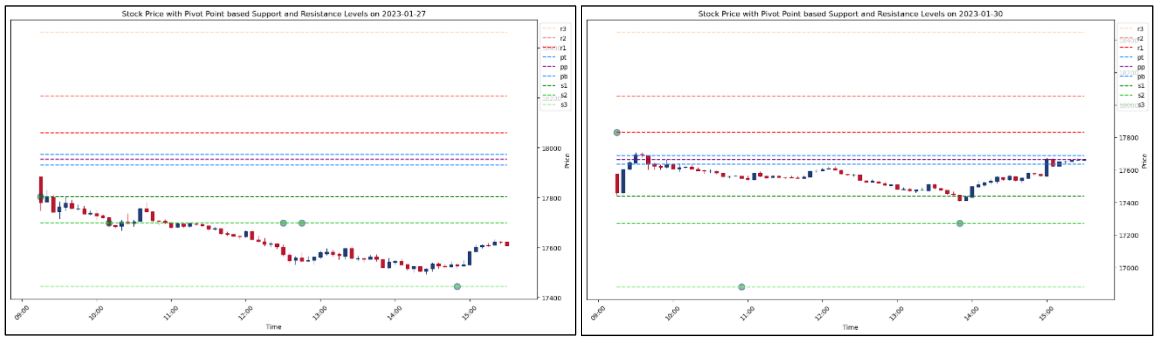


Figure 4.3.2.3: Sample predictions for visual analysis – 2000 Epochs

4.3.2.4 Experiment 4 – 5000 Epochs

Table 4.3.2.4 (a): Training Configuration – 5000 Epochs

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	5000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.2.4 (b): Evaluation Results – 5000 Epochs

Evaluation Metric	Result	Comments
Prediction Accuracy	86.45%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	41.38%	Indicates the % of useful predictions

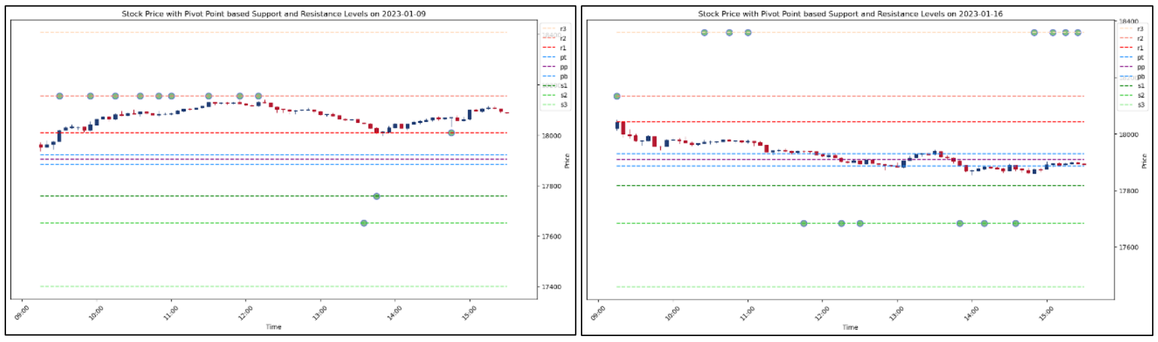


Figure 4.3.2.4: Sample predictions for visual analysis – 5000 Epochs

4.3.2.5 Summary

From the above set of experiments to try out different epochs, it can be observed that 1000 epochs work best for training the proposed LSTM-based system. So this will be used in the subsequent experiments as a benchmark.

4.3.3 Experiments to find out Optimal Model Architecture

In this set of experiments, we will experiment with configuration parameters associated with the model architecture while keeping the rest of the configuration parameters constant. We will try out 50, 100, 200, 500 for hidden dimensions with 1 layer. Then we will experiment with 1, 2, and 5 for the number of layers while keeping the hidden dimensions to 100.

4.3.3.1 Experiment 1 – 50 Hidden Dimensions

Table 4.3.3.1 (a): Training Configuration – 50 Hidden Dimensions

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	50	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.3.1 (b): Evaluation Results – 50 Hidden Dimensions

Evaluation Metric	Result	Comments
Prediction Accuracy	92.76%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	57.14%	Indicates the % of useful predictions

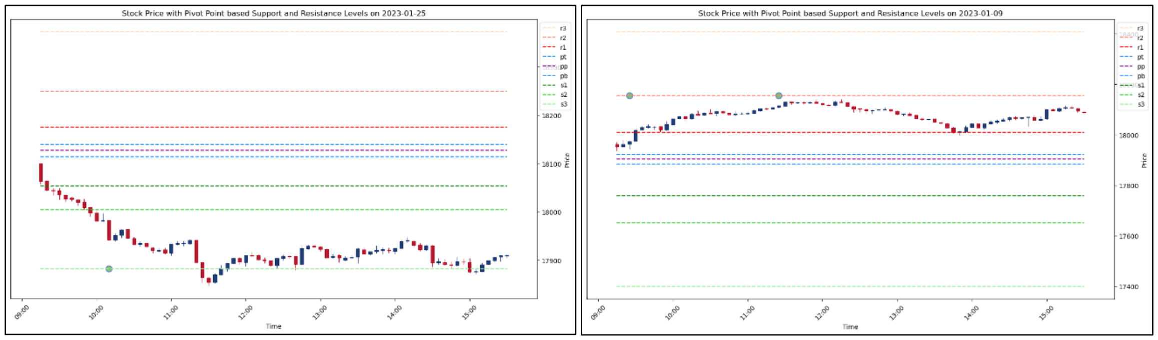


Figure 4.3.3.1: Sample predictions for visual analysis – 50 Hidden Dimensions

4.3.3.2 Experiment 2 – 100 Hidden Dimensions

Table 4.3.3.2 (a): Training Configuration – 100 Hidden Dimensions

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.3.2 (b): Evaluation Results – 100 Hidden Dimensions

Evaluation Metric	Result	Comments
Prediction Accuracy	92.70%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	72.73%	Indicates the % of useful predictions

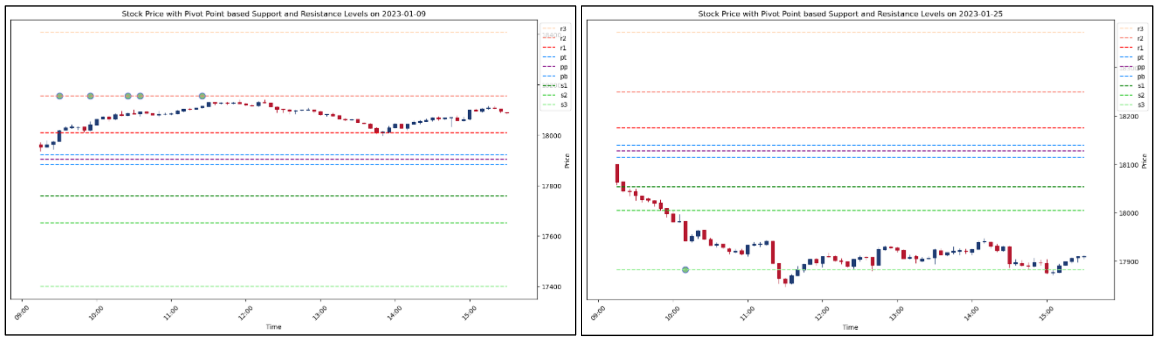


Figure 4.3.3.2: Sample predictions for visual analysis – 100 Hidden Dimensions

4.3.3.3 Experiment 3 – 200 Hidden Dimensions

Table 4.3.3.3 (a): Training Configuration – 200 Hidden Dimensions

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	200	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.3.3 (b): Evaluation Results – 200 Hidden Dimensions

Evaluation Metric	Result	Comments
Prediction Accuracy	92.57%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	50.00%	Indicates the % of useful predictions

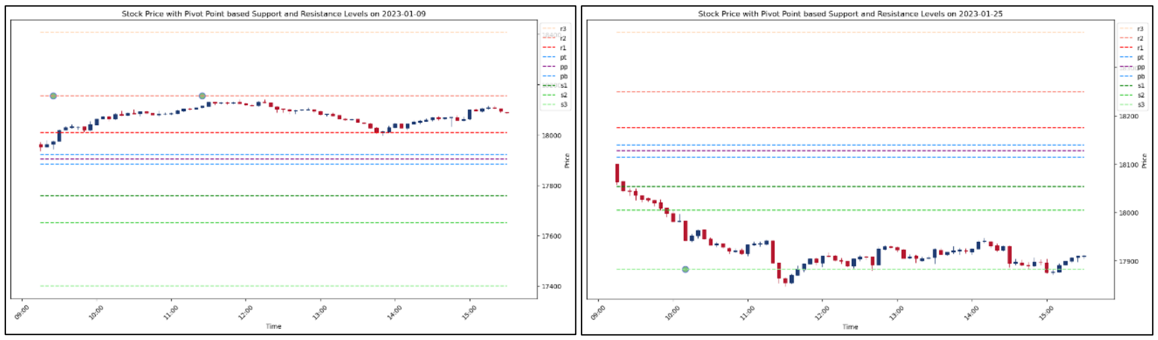


Figure 4.3.3.3: Sample predictions for visual analysis – 200 Hidden Dimensions

4.3.3.4 Experiment 4 – 500 Hidden Dimensions

Table 4.3.3.4 (a): Training Configuration – 500 Hidden Dimensions

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	500	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.3.4 (b): Evaluation Results – 500 Hidden Dimensions

Evaluation Metric	Result	Comments
Prediction Accuracy	92.76%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	60.00%	Indicates the % of useful predictions

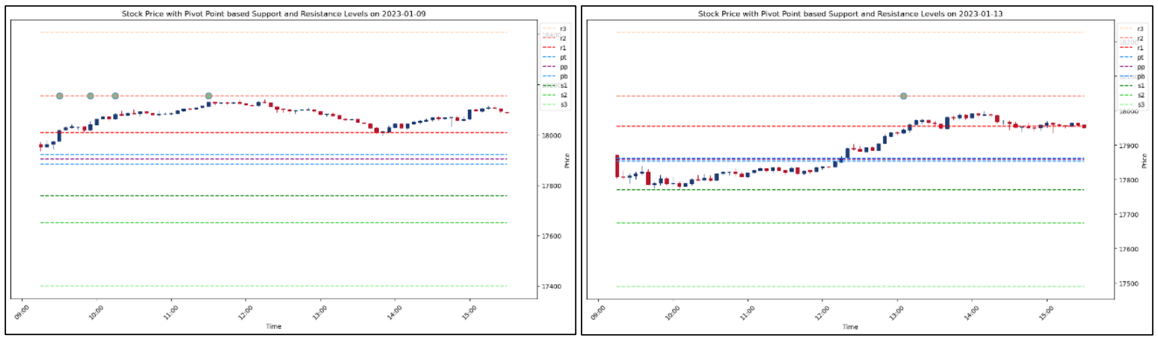


Figure 4.3.3.4: Sample predictions for visual analysis – 500 Hidden Dimensions

4.3.3.5 Experiment 5 – 2 Layers

Table 4.3.3.5 (a): Training Configuration – 2 Layers

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	2	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.3.5 (b): Evaluation Results – 2 Layers

Evaluation Metric	Result	Comments
Prediction Accuracy	79.34%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	47.35%	Indicates the % of useful predictions

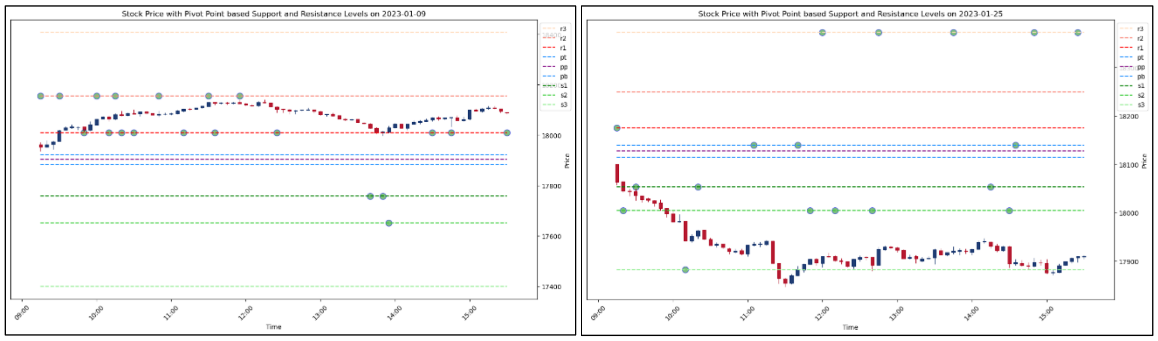


Figure 4.3.3.5: Sample predictions for visual analysis – 2 Layers

4.3.3.6 Experiment 6 – 5 Layers

Table 4.3.3.6 (a): Training Configuration – 5 Layers

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	5	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.3.6 (b): Evaluation Results – 5 Layers

Evaluation Metric	Result	Comments
Prediction Accuracy	81.91%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	46.15%	Indicates the % of useful predictions

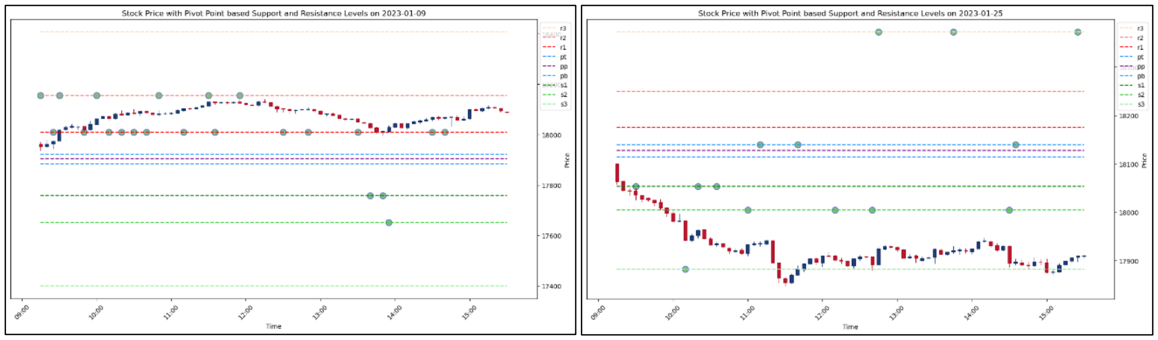


Figure 4.3.3.6: Sample predictions for visual analysis – 5 Layers

4.3.3.7 Summary

From the above set of experiments to try out different model architectures, it can be observed that 1 layer architecture with 100 hidden dimensions works optimally for training the proposed LSTM-based system. So this will be used in the subsequent experiments as a benchmark.

4.3.4 Experiments to find out Optimal Training Data Size

In this set of experiments, we will try different dataset sizes, such as 3 months, 6 months, 1 year and 3 years. Also for 3 months and 6 months, we will do additional experiments with more epochs to proportionally keep up with the ideal epoch value of 1000.

4.3.4.1 Experiment 1 – 3 Months Training Data

Table 4.3.4.1 (a): Training Configuration – 3 Months Training Data

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY_3Months.txt	3 months – from October to December - dataset of Nifty 2022
LSTM Hidden Dimensions	50	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.1 (b): Evaluation Results – 3 Months Training Data

Evaluation Metric	Result	Comments
Prediction Accuracy	91.51%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	21.43%	Indicates the % of useful predictions

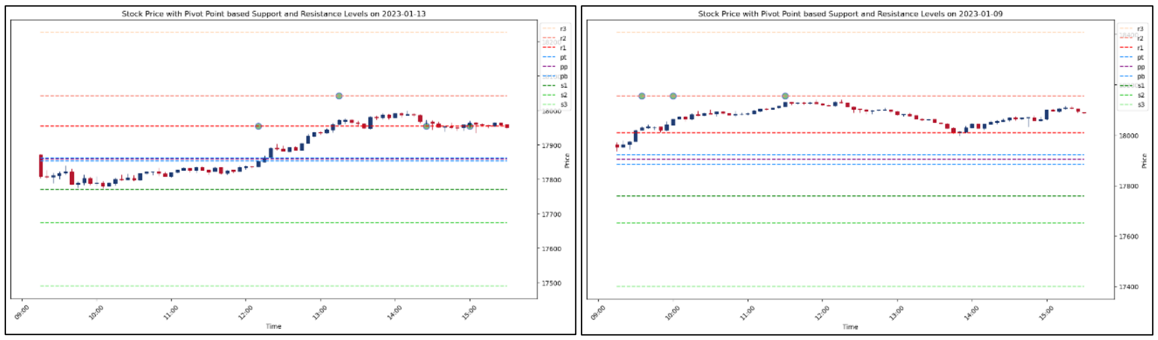


Figure 4.3.4.1: Sample predictions for visual analysis – 3 Months Training Data

4.3.4.2 Experiment 2 – 3 Months Training Data with 4000 Epochs

Table 4.3.4.2 (a): Training Configuration – 3 Months Training Data with 4000 Epochs

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY_3Months.txt	3 months – from October to December - dataset of Nifty 2022
LSTM Hidden Dimensions	50	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	4000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.2 (b): Evaluation Results – 3 Months Training Data with 4000 Epochs

Evaluation Metric	Result	Comments
Prediction Accuracy	86.05%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	45.52%	Indicates the % of useful predictions

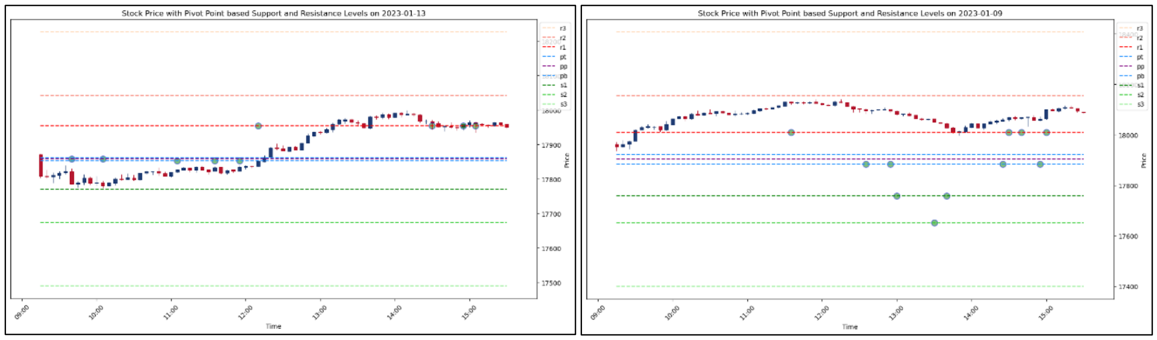


Figure 4.3.4.2: Sample predictions for visual analysis – 3 Months Training Data with 4000 Epochs

4.3.4.3 Experiment 3 – 6 Months Training Data

Table 4.3.4.3 (a): Training Configuration – 6 Months Training Data

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY_6Months.txt	6 months – from July to December - dataset of Nifty 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.3 (b): Evaluation Results – 6 Months Training Data

Evaluation Metric	Result	Comments
Prediction Accuracy	92.50%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	64.29%	Indicates the % of useful predictions

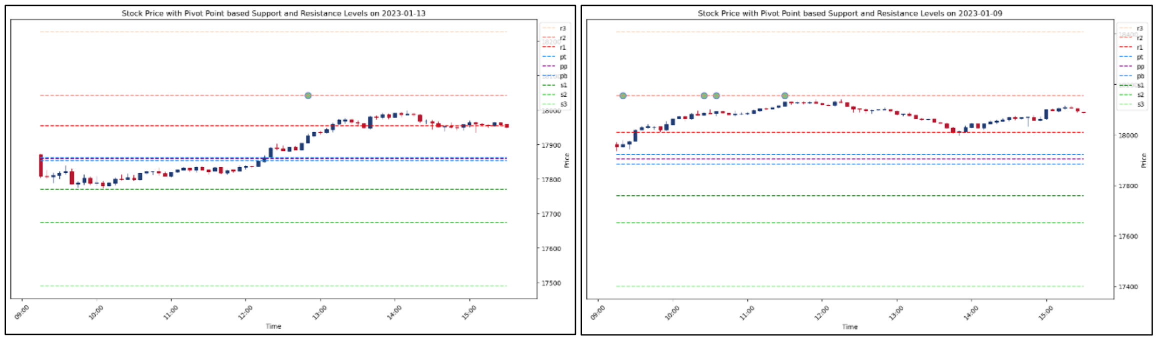


Figure 4.3.4.3: Sample predictions for visual analysis – 6 Months Training Data

4.3.4.4 Experiment 4 – 6 Months Training Data with 2000 Epochs

Table 4.3.4.4 (a): Training Configuration – 6 Months Training Data with 2000 Epochs

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY_6Months.txt	6 months – from July to December - dataset of Nifty 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	2000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.4 (b): Evaluation Results – 6 Months Training Data with 2000 Epochs

Evaluation Metric	Result	Comments
Prediction Accuracy	90.99%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	21.95%	Indicates the % of useful predictions

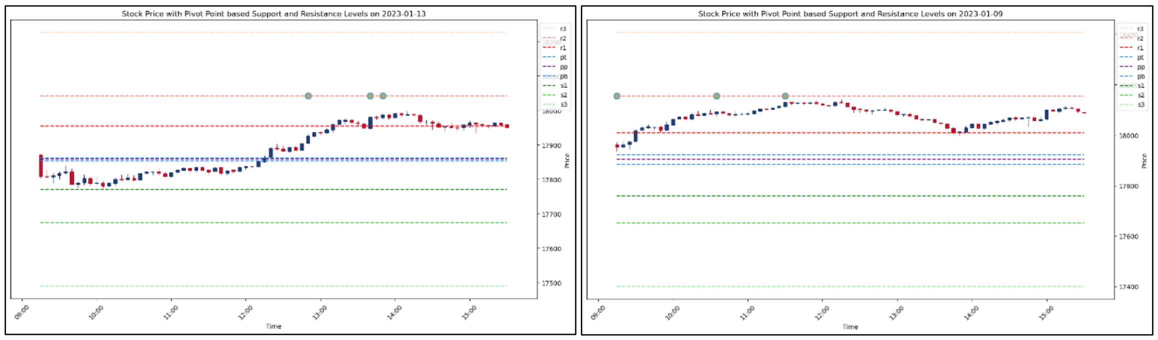


Figure 4.3.4.4: Sample predictions for visual analysis – 6 Months Training Data with 2000 Epochs

4.3.4.5 Experiment 5 – 1 Year Training Data

Table 4.3.4.5 (a): Training Configuration – 1 Year Training Data

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	200	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.5 (b): Evaluation Results – 1 Year Training Data

Evaluation Metric	Result	Comments
Prediction Accuracy	92.70%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	72.73%	Indicates the % of useful predictions

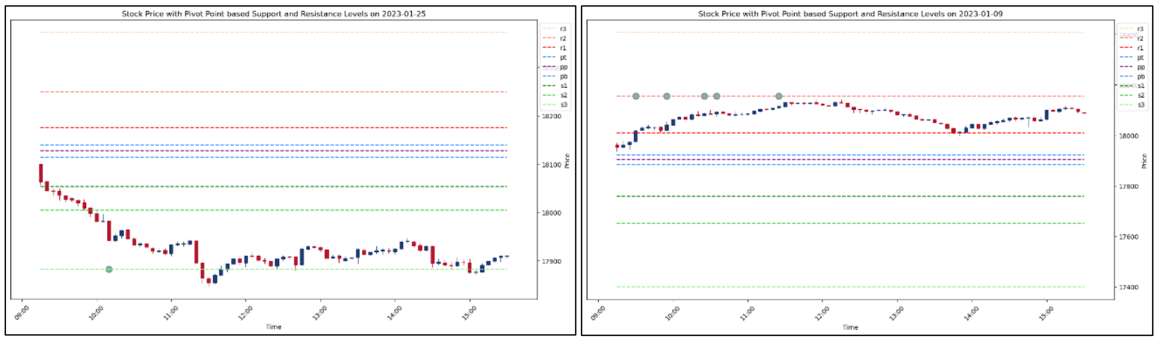


Figure 4.3.4.5: Sample predictions for visual analysis – 1 Year Training Data

4.3.4.6 Experiment 6 – 3 Years Training Data

Table 4.3.4.6 (a): Training Configuration – 3 Years Training Data

Configuration Parameter	Value	Comments
Training Dataset	2020-2022_NIFTY.txt	3 years dataset of Nifty Index from 2020 to 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.6 (b): Evaluation Results – 3 Years Training Data

Evaluation Metric	Result	Comments
Prediction Accuracy	93.22%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	00.00%	Indicates the % of useful predictions

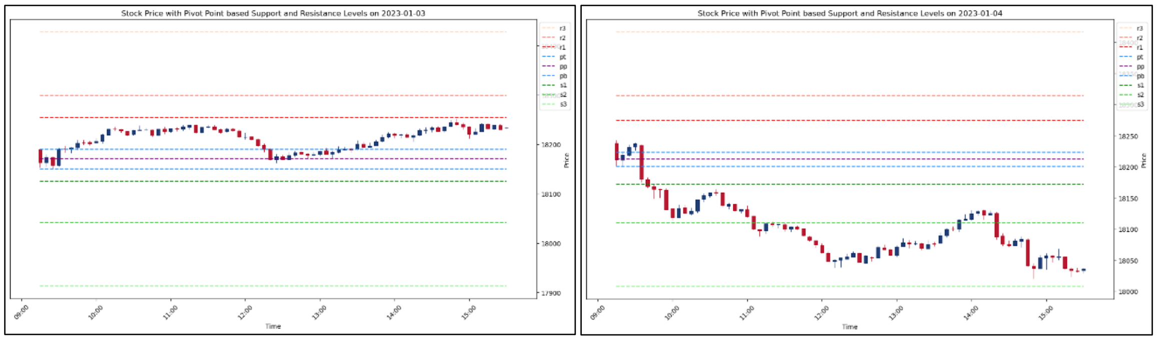


Figure 4.3.4.6: Sample predictions for visual analysis – 3 Years Training Data

4.3.4.7 Experiment 7 – 3 Years Training Data of 2017-2019

Table 4.3.4.7 (a): Training Configuration – 3 Years Training Data of 2017-2019

Configuration Parameter	Value	Comments
Training Dataset	2017-2019_NIFTY.txt	3 years dataset of Nifty Index from 2017 to 2019
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI	List of input Columns/Features taken for training the model
Test Dataset	2020_NIFTY.txt	First 30 trading days of the January 2020 Nifty Index dataset
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.4.7 (b): Evaluation Results – 3 Years Training Data of 2017-2019

Evaluation Metric	Result	Comments
Prediction Accuracy	95.13%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	50.00%	Indicates the % of useful predictions

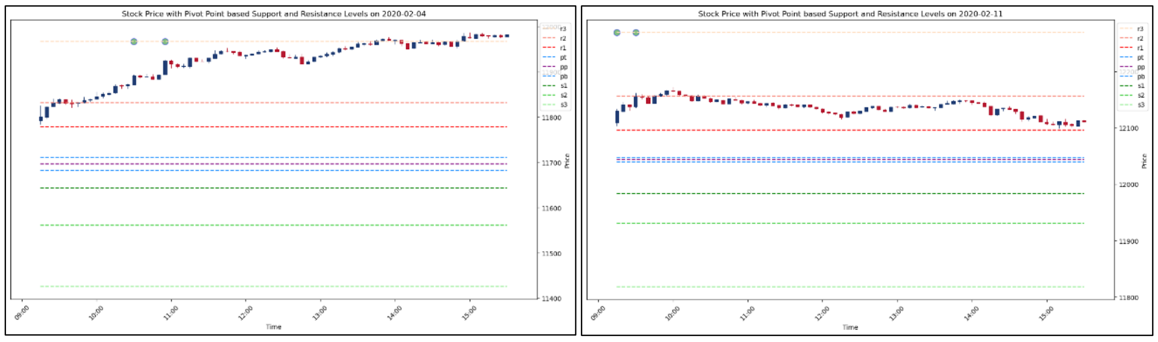


Figure 4.3.4.7: Sample predictions for visual analysis – 3 Years Training Data of 2017-2019

4.3.4.8 Summary

From the above set of experiments to figure out the optimal training data size, it can be observed that 1-year dataset works best for training the proposed LSTM-based system. So this will be used in the subsequent experiments as a benchmark. Also, it is noted that using 3-year datasets provided little to no predictions during testing.

4.3.5 Optimal Mix of Technical Indicators & Candlestick Patterns

In this set of experiments, we will try out different, but not all, combinations of technical indicators and candle patterns discussed in this study. Combinations are formed based on the results of experiments to find out the best individual technical indicators and candle patterns.

4.3.5.1 Experiment 1 – RSI & EMA14

Table 4.3.5.1 (a): Training Configuration – RSI & EMA14

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI, EMA14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.1 (b): Evaluation Results – RSI & EMA14

Evaluation Metric	Result	Comments
Prediction Accuracy	92.70%	Accuracy in predicting ‘NextLevel’
Prediction Usefulness	55.56%	Indicates the % of useful predictions

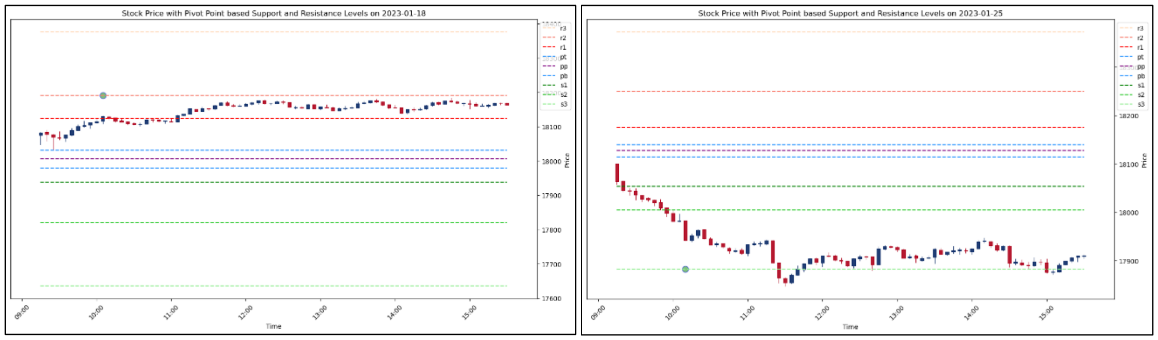


Figure 4.3.5.1: Sample predictions for visual analysis – RSI & EMA14

4.3.5.2 Experiment 2 – RSI & EMA CROSSOVER

Table 4.3.5.2 (a): Training Configuration – RSI & EMA CROSSOVER

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.2 (b): Evaluation Results – RSI & EMA CROSSOVER

Evaluation Metric	Result	Comments
Prediction Accuracy	92.04%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	75.00%	Indicates the % of useful predictions

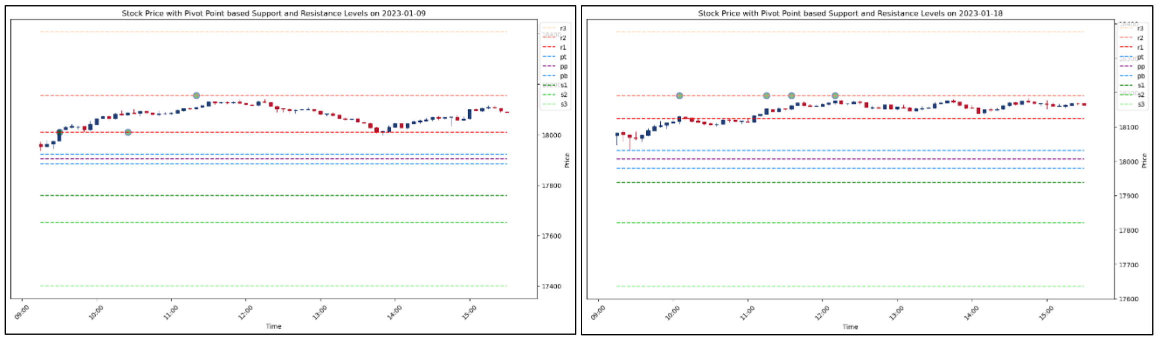


Figure 4.3.5.2: Sample predictions for visual analysis – RSI & EMA CROSSOVER

4.3.5.3 Experiment 3 – RSI, EMA14 & EMA CROSSOVER

Table 4.3.5.3 (a): Training Configuration – RSI, EMA14 & EMA CROSSOVER

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open, High Low, Close RSI, EMA14 EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.3 (b): Evaluation Results – RSI, EMA14 & EMA CROSSOVER

Evaluation Metric	Result	Comments
Prediction Accuracy	91.64%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	53.57%	Indicates the % of useful predictions

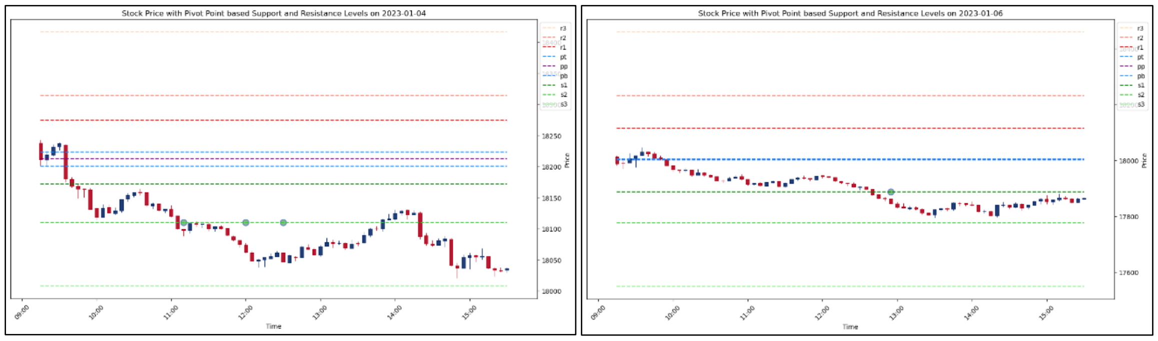


Figure 4.3.5.3: Sample predictions for visual analysis - RSI, EMA14 & EMA CROSSOVER

4.3.5.4 Experiment 4 – RSI & ATR

Table 4.3.5.4 (a): Training Configuration – RSI & ATR

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI, ATR	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.4 (b): Evaluation Results – RSI & ATR

Evaluation Metric	Result	Comments
Prediction Accuracy	90.86%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	51.06%	Indicates the % of useful predictions

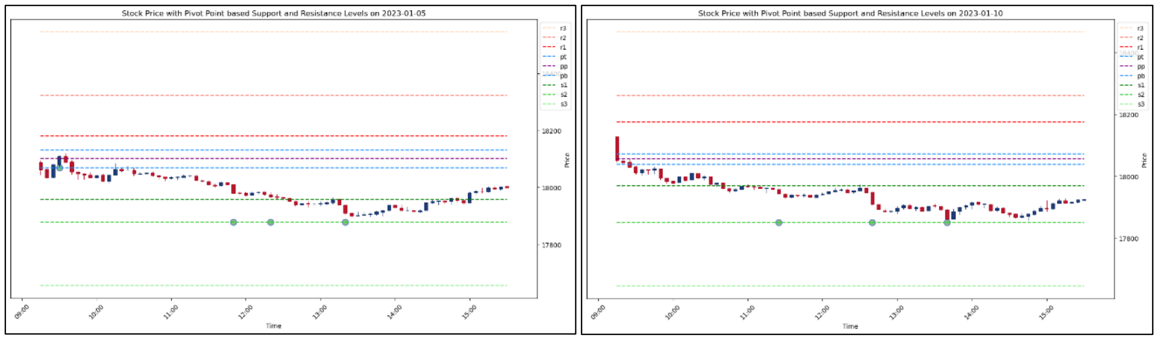


Figure 4.3.5.4: Sample predictions for visual analysis – RSI & ATR

4.3.5.5 Experiment 5 – RSI, EMA14 & ATR

Table 4.3.5.5 (a): Training Configuration – RSI, EMA14 & ATR

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close RSI, EMA14, ATR	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.5 (b): Evaluation Results – RSI, EMA14 & ATR

Evaluation Metric	Result	Comments
Prediction Accuracy	91.84%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	48.15%	Indicates the % of useful predictions

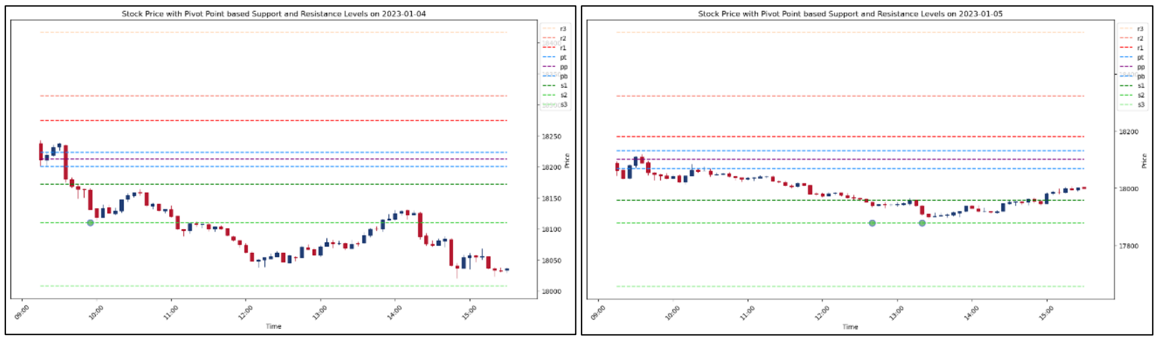


Figure 4.3.5.5: Sample predictions for visual analysis – RSI, EMA14 & ATR

4.3.5.6 Experiment 6 – RSI & Single Candle Patterns

Table 4.3.5.6 (a): Training Configuration – RSI & Single Candle Patterns

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open, High Low, Close, RSI CDLMARUBOZU, CDLDOJI, CDLHAMMER, CDLHANGINGMAN	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.6 (b): Evaluation Results – RSI & Single Candle Patterns

Evaluation Metric	Result	Comments
Prediction Accuracy	89.80%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	57.41%	Indicates the % of useful predictions

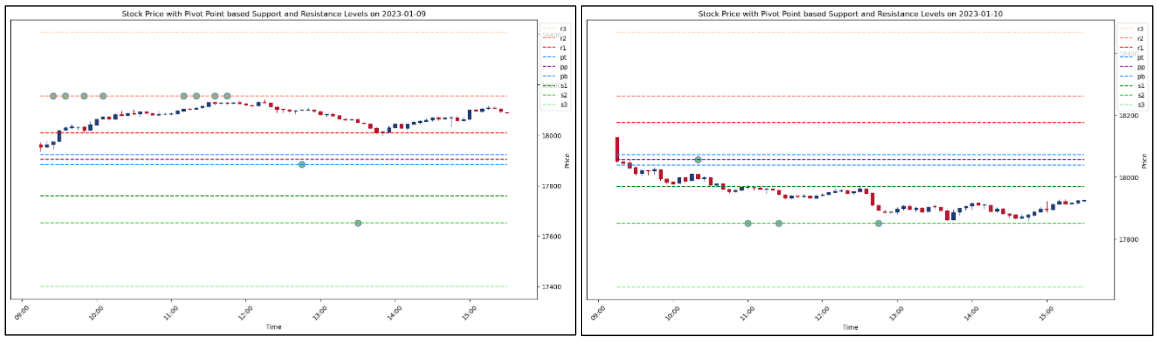


Figure 4.3.5.6: Sample predictions for visual analysis – RSI & Single Candle Patterns

4.3.5.7 Experiment 7 – RSI & Double Candle Patterns

Table 4.3.5.7 (a): Training Configuration – RSI & Double Candle Patterns

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open, High Low, Close, RSI CDLENGULFING CDLHARAMI	List of input Columns/Features taken for training the model. ADX is generally used in combinations with Plus and Minus DIs to get direction
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.7 (b): Evaluation Results – RSI & Double Candle Patterns

Evaluation Metric	Result	Comments
Prediction Accuracy	91.05%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	35.14%	Indicates the % of useful predictions

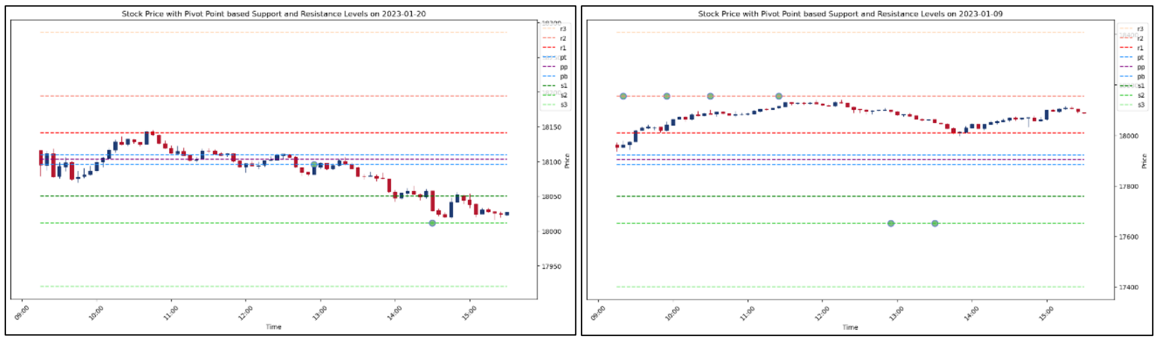


Figure 4.3.5.7: Sample predictions for visual analysis – RSI & Double Candle Patterns

4.3.5.8 Experiment 8 – RSI & Triple Candle Patterns

Table 4.3.5.8 (a): Training Configuration – RSI & Triple Candle Patterns

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of Nifty Index for the year 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open, High, Low, Close, RSI CDLMORNINGSTAR CDLEVENINGSTAR CDL3WHITESOLDIERS CDL3BLACKCROWS	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of Nifty Index
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.3.5.8 (b): Evaluation Results – RSI & Triple Candle Patterns

Evaluation Metric	Result	Comments
Prediction Accuracy	92.50%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	23.08%	Indicates the % of useful predictions

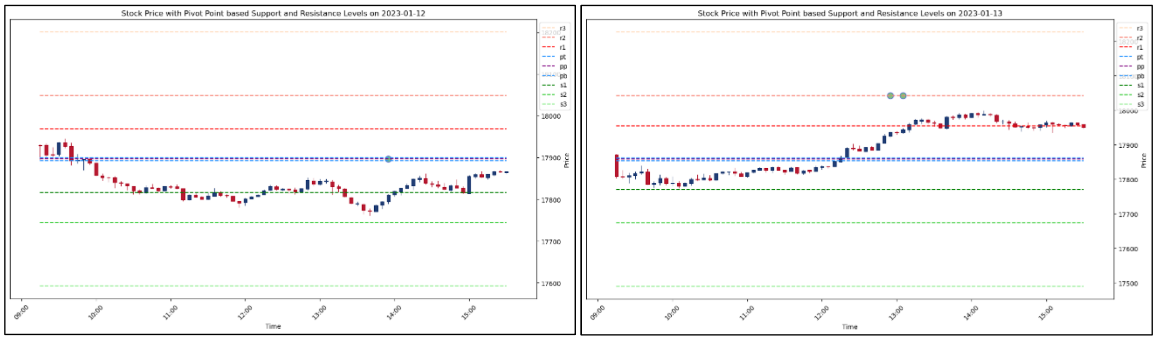


Figure 4.3.5.8: Sample predictions for visual analysis – RSI & Triple Candle Patterns

4.3.5.9 Experiment 9 – RSI & All Candle Patterns

Table 4.3.5.9 (a): Training Configuration – RSI & All Candle Patterns

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	Year 2022 dataset of Nifty
Hidden Dimensions	100	Number of LSTM units
LSTM Number of Layers	1	Number of LSTM layers
Training Epochs	1000	Number of epochs
Input Features	CDLMARUBOZU, CDLENGULFING Open, Low, High, Close, RSI, CDLDOJI, CDLHANGINGMAN, CDLHARAMI CDLMORNINGSTAR, CDLHAMMER CDLEVENINGSTAR, CDL3WHITESOLDIERS CDL3BLACKCROWS	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	Nifty January 2023 dataset
Probability Threshold	0.7	Threshold for predictions

Table 4.3.5.9 (b): Evaluation Results – RSI & All Candle Patterns

Evaluation Metric	Result	Comments
Prediction Accuracy	88.68%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	46.67%	Indicates the % of useful predictions

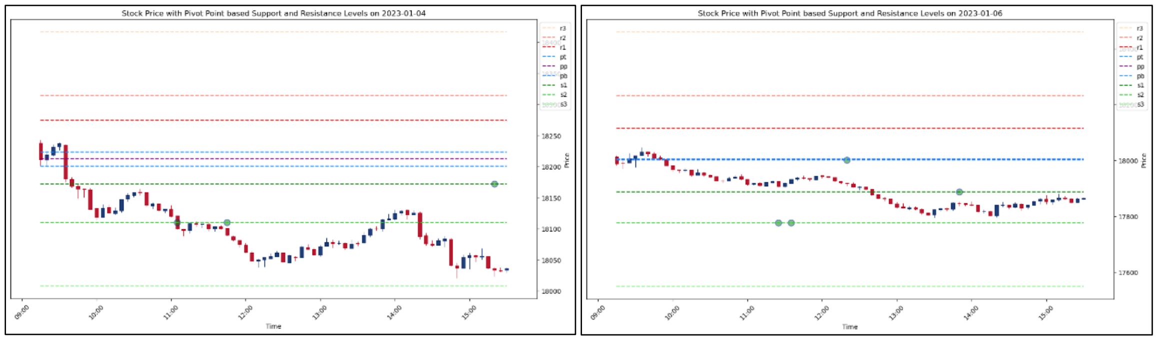


Figure 4.3.5.9: Sample predictions for visual analysis – RSI & All Candle Patterns

4.3.5.10 Summary

From the above set of experiments to figure out the optimal combination of technical indicators and patterns, it is observed that the combination of RSI and EMA_CROSSOVER_7AND14 along with the candle's Open, High, Low, and Close values outperforms everything else with an outstanding Prediction Usefulness score of 75% and Prediction Accuracy of 92%.

4.3.6 Conclusion

From the different set of experiments conducted above, we can outline the optimal set of parameters for training the proposed model as follows:

Table 4.3.6: Optimal Training Configuration

Configuration Parameter	Value
Training Dataset Size	1-year dataset
LSTM Hidden Dimensions	100
LSTM Number of Layers	1
Training Epochs	1000
Input Features	Open, High, Low, Close, RSI, EMA_CROSSOVER_7AND14

And we can conclude that the best set of technical indicators for training the LSTM-based model proposed in this study is the Open, High, Low, and Close prices of the candles, the Relative Strength Index and the Cross over of Exponential Moving Averages for the period of 7 and 14.

4.4 Proposed Model Proves Effective for Both Indices and Stocks

To answer our next research question “Can this learning be applied to different types of stock assets like indices and individual stocks?”, let’s run experiments on some of the stocks taken from the National Stock Exchange, India. We will use the optimal parameters outlined in the last chapter for the experiments on stocks too. The results of conducting experiments on some of the most popular stocks in the National Stock Exchange are listed below.

4.4.1 Experiment 1 – INFY

Table 4.4.1 (a): Training Configuration – INFY

Configuration Parameter	Value	Comments
Training Dataset	2022_INFY.txt	1-year dataset of INFY 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close, RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_INFY.txt	January 2023 dataset of INFY
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.4.1 (b): Evaluation Results – INFY

Evaluation Metric	Result	Comments
Prediction Accuracy	91.97%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	66.67%	Indicates the % of useful predictions

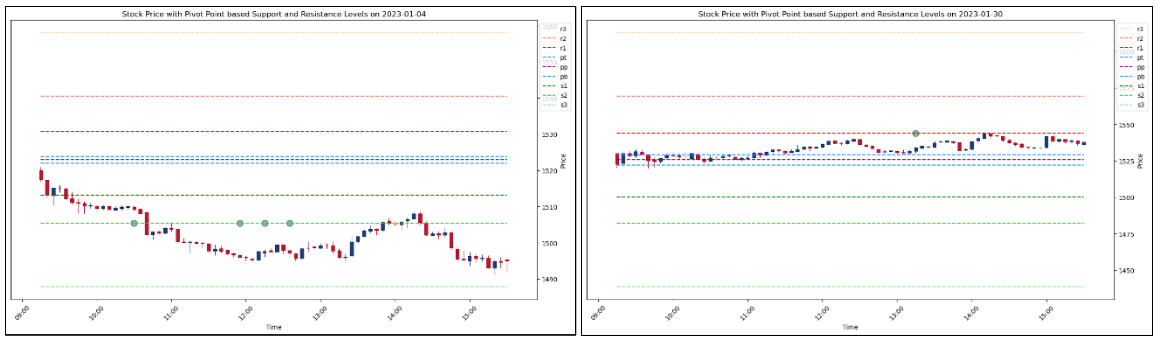


Figure 4.4.1: Sample predictions for visual analysis – INFY

4.4.2 Experiment 2 – HINDUNILVR

Table 4.4.2 (a): Training Configuration – HINDUNILVR

Configuration Parameter	Value	Comments
Training Dataset	2022_HINDUNILVR.txt	1-year dataset of HINDUNILVR 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low, Close, RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_HINDUNILVR.txt	January 2023 dataset of HINDUNILVR
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.4.2 (b): Evaluation Results – HINDUNILVR

Evaluation Metric	Result	Comments
Prediction Accuracy	94.47%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	63.64%	Indicates the % of useful predictions

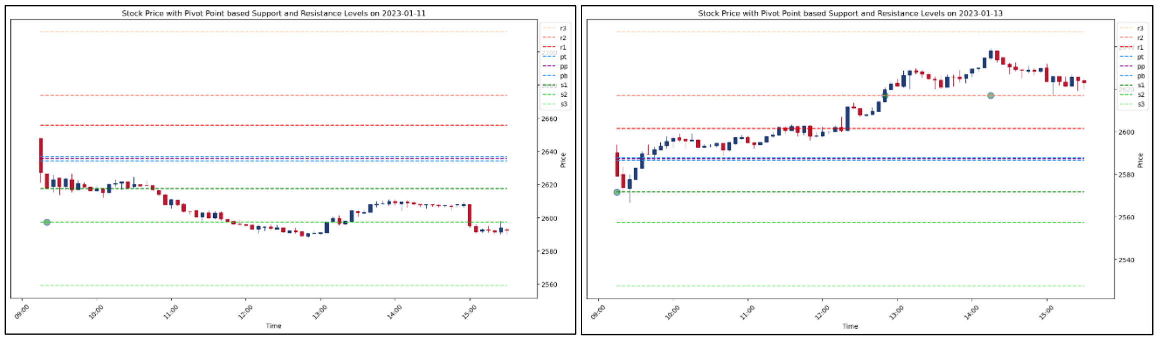


Figure 4.4.2: Sample predictions for visual analysis – HINDUNILVR

4.4.3 Experiment 3 – HEROMOTOCO

Table 4.4.3 (a): Training Configuration – HEROMOTOCO

Configuration Parameter	Value	Comments
Training Dataset	2022_HEROMOTOCO.txt	1-year dataset of HEROMOTOCO 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low, Close, RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_HEROMOTOCO.txt	January 2023 dataset of HEROMOTOCO
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.4.3 (b): Evaluation Results – HEROMOTOCO

Evaluation Metric	Result	Comments
Prediction Accuracy	91.51%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	58.33%	Indicates the % of useful predictions

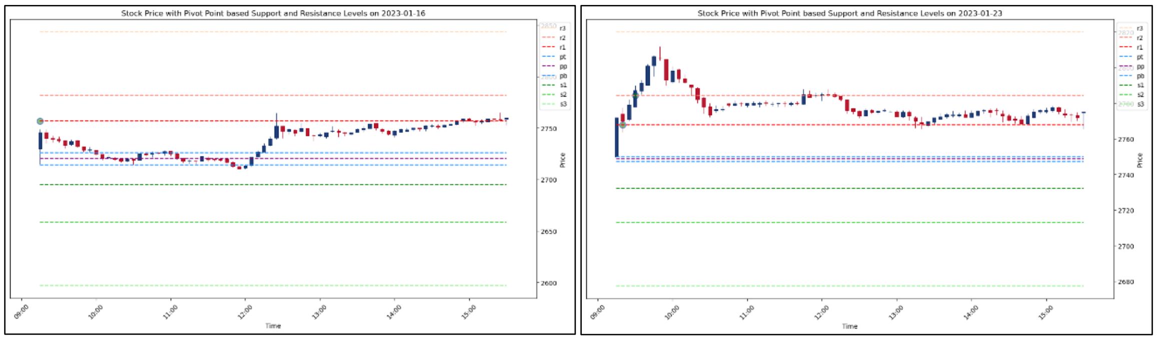


Figure 4.4.3: Sample predictions for visual analysis – HEROMOTOCO

4.4.4 Experiment 4 – ITC

Table 4.4.4 (a): Training Configuration – ITC

Configuration Parameter	Value	Comments
Training Dataset	2022_ITC.txt	1-year dataset of ITC 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close, RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_ITC.txt	January 2023 dataset of ITC
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.4.4 (b): Evaluation Results – ITC

Evaluation Metric	Result	Comments
Prediction Accuracy	86.97%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	57.14%	Indicates the % of useful predictions

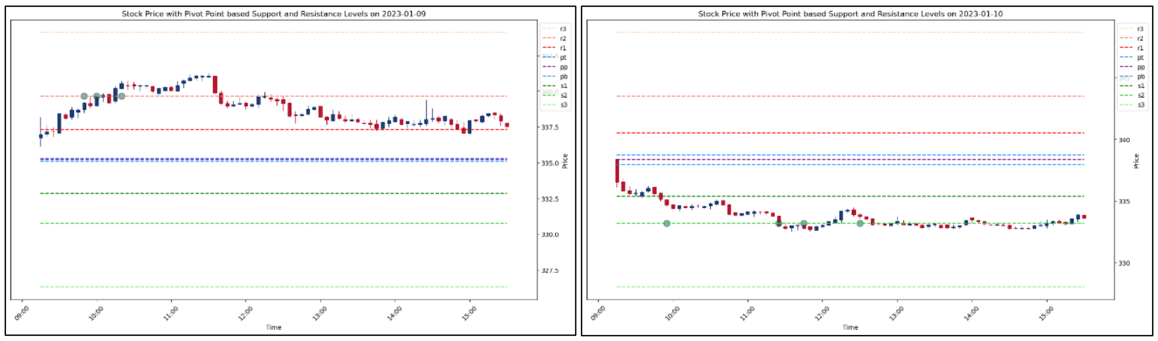


Figure 4.4.4: Sample predictions for visual analysis – ITC

4.4.5 Conclusion

From the results of different sets of experiments conducted above, enough pieces of evidence are found to prove that the proposed model architecture works perfectly fine with different types of stock assets like individual stocks and indices.

4.5 Proposed Model Proves Effective for Different Timeframes

To answer the final research question “Can this learning be applied to different timeframes?”, we will experiment with different timeframes to see if the proposed model architecture works well regardless of timeframes. To minimize code changes, we will run experiments with 2 other timeframes – 15 minutes and 1 hour. We will stick to the optimal configurations found in the earlier chapters of this study, except for the 1-hour time frame, we will use 3 3-year dataset to have enough candlesticks needed for efficient training.

4.5.1 Experiment 1 – 15-minute Timeframe

Table 4.5.1 (a): Training Configuration – 15-minute Timeframe

Configuration Parameter	Value	Comments
Training Dataset	2022_NIFTY.txt	1-year dataset of NIFTY 2022
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low Close, RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of NIFTY
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.5.1 (b): Evaluation Results – 15-minute Timeframe

Evaluation Metric	Result	Comments
Prediction Accuracy	85.77%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	56.25%	Indicates the % of useful predictions

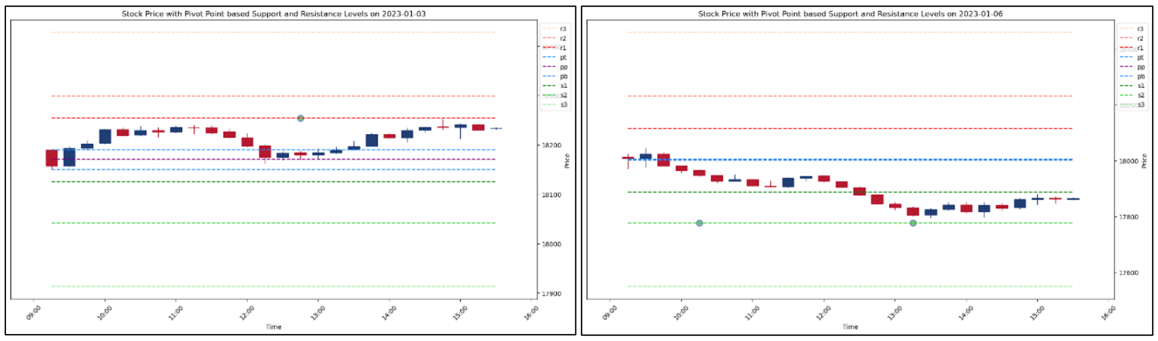


Figure 4.5.1: Sample predictions for visual analysis – 15-minute Timeframe

4.5.2 Experiment 2 – 1 Hour Timeframe

Table 4.5.2 (a): Training Configuration – 1 Hour Timeframe

Configuration Parameter	Value	Comments
Training Dataset	2020-2022_NIFTY.txt	3 years dataset of NIFTY 2020-22
LSTM Hidden Dimensions	100	Number of LSTM units per layer
LSTM Number of Layers	1	Number of LSTM layers in the model
Training Epochs	1000	Number of epochs in the training loop
Input Features	Open High Low, Close, RSI EMA_CROSSOVER_7AND14	List of input Columns/Features taken for training the model
Test Dataset	2023JAN_NIFTY.txt	January 2023 dataset of NIFTY
Probability Threshold	0.7	Predictions above this value will only be considered

Table 4.5.2 (b): Evaluation Results – 1 Hour Timeframe

Evaluation Metric	Result	Comments
Prediction Accuracy	72.86%	Accuracy in predicting 'NextLevel'
Prediction Usefulness	50.64%	Indicates the % of useful predictions

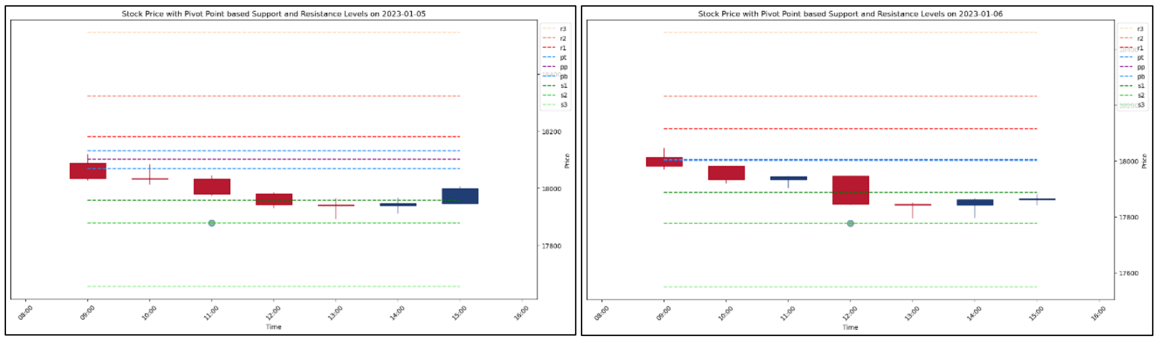


Figure 4.5.2: Sample predictions for visual analysis – 1 Hour Timeframe

4.5.3 Summary

There is enough evidence to prove that the proposed model architecture works for different timeframes with good results. Remember that the pivot points are not limited to any timeframes. It only depends on the previous high, previous low and previous close values to calculate pivot levels for the current trade session – it doesn’t matter whether it is day, week or month.

For swing trading with 4 Hours candles for a week, calculate the weekly pivot lines using last week’s high, last week’s low and last week’s close prices.

For positional trading with day candles for a month, calculate the monthly pivot lines using last month’s high, last month’s low and last month’s close prices.

The logic to train and test the proposed model is not dependent on the time frame, so any timeframe can be accommodated. We only need to change the code in the data preprocessing stage to correctly calculate the pivot lines according to the timeframe that we select, and also incorporate similar changes in the marking on ‘NextLevel’. Once the dataset has this target column ‘NextLevel’, then model training can be carried out without any changes. During the testing phase, we need to make similar timeframe-related changes in the report generation functions to check if the price will touch the predicted levels or not.

4.6 Optimizing Model Retraining Schedules

To examine the duration of model effectiveness before we need to retrain the model with an updated set of datasets to include recent data, an experiment was conducted to train a model with optimal configuration on the 2017 dataset. Then this model was evaluated with the proposed metrics for every month in 2018. Summarized below are the experiment's key findings.

Table 4.6: Experiment Summary for Optimizing Model Retraining Schedules

Test Month	Prediction Accuracy	Prediction Usefulness
January	88.53%	60.71%
February	89.40%	46.43%
March	92.54%	47.62%
April	89.80%	64.58%
May	88.28%	57.95%
June	88.49%	60.44%
July	87.47%	71.11%
August	88.55%	65.91%
September	86.69%	37.50%
October	89.41%	54.05%
November	89.55%	43.08%
December	88.78%	40.66%

From the results, it can be interpreted that the model's Prediction Accuracy and Prediction Usefulness stayed good for the next 8 months, once trained with a 1-year dataset. After that, there is a decline in the Prediction Usefulness score. So the retraining schedule for a trained model can be assumed as 8 months, provided the price range of the stock asset isn't changing much.

4.7 Summary of Findings

In this chapter, we concluded that it is possible to train an LSTM-based model to predict the pivot levels that the future candles are going to touch.

This study also demonstrated the process of mapping the 9 pivot levels to a single target column in a dataset, enabling the training of an artificial intelligence model with this column as the target variable.

Moreover, through systematic selection of configuration parameters over a variety of experiments, it is possible to deduce that the optimal configuration for training the proposed model will be as follows:

Table 4.7: Optimal Training Configuration

Configuration Parameter	Value
Training Dataset Size	1-year dataset
LSTM Hidden Dimensions	100
LSTM Number of Layers	1
Training Epochs	1000
Input Features	Open, High, Low, Close, RSI, EMA_CROSSOVER_7AND14

Using the above optimal configurations, our best model could achieve a Prediction Accuracy of 92.04% and a Prediction Usefulness score of 75%.

Furthermore, it is noted that the suggested model has the potential to perform effectively for different types of stock assets like indices and individual stocks.

In addition, it is anticipated that the model proposed could function efficiently for all timeframes.

Once trained with a 1-year dataset, it is observed that the retraining schedule for a trained model can be concluded as 8 months.

4.8 Conclusion

The findings from this chapter answered all the research questions formulated to guide this study. The insights gained from various experiments completed as part of this chapter will be discussed in the next chapter and will also pave the way for recommendations for future research.

CHAPTER V:
DISCUSSION

5.1 How to Make Profits Without Knowing Price Trends?

In theory, it is quite possible to make profits without knowing the direction of the price movements of the stock assets and just by random selection of direction. These are the fundamental principles of trading if we want to do it successfully for a longer period. Before commencing, it is imperative to bear in mind that when we are randomly selecting the direction of the price movement, the probability of an outcome in our favour is 50%. Now, let’s examine these fundamental principles in trading.

5.1.1 The Law of Large Numbers

The law of large numbers (Anon, n.d.h) states that the average of the results independent and identical random samples converge to the true value if it exists. Furthermore, it is asserted that when a coin is flipped a large number of times, the probability of predicting one side would converge close to 50%.

In trading, if we arbitrarily select a direction as possible price movement, and if we do the same a large number of times, then according to the law of large numbers, the probability of making winning trades and losing trades equally stands at 50%.

How do we ensure that we get a chance of making a large number of trades, without depleting our entire trading capital in the first few losing trades itself? The second principle offers a solution to this challenge.

5.1.2 The One Percent Rule

“Never risk more than 1% of your capital on any single trade” (Burns & Burns, 2021), suggested that the traders should carry out appropriate position sizing so that not more than 1% of the trading capital is lost on any single trade. This fundamental principle enables the trader to carry out a relatively large number of trades, compared to risking a huge chunk of the trading capital in one or two losing trades. (Burns & Burns, 2021) also indicated that the trader should first place the stop-loss at a safe enough position on the chart and then adjust the position size accordingly – it should not be the other way around.

And how can we guarantee that with this large volume of trades with a success rate of 50% - meaning half of the trades turn out to be profitable and the other half are still losing trades -, the overall trading outcome remains profitable?

5.1.3 Higher Risk-to-Reward Ratio

(Katz & McCormick, 2000) outlined that when a trade is profitable, secure greater profits, and when the trade is unfavourable, minimize losses. In addition, it is asserted that this objective is accomplished by implementing stop-loss orders to limit losses and setting target orders to realize profits. Additionally, it is mentioned that the ideal placement for the stop-loss order is 1.5 Average True Range (ATR) points from the entry point, and the target is set at 4.5 ATR points from the entry, establishing a risk-to-reward ratio of 1:3.

According to (Burns & Burns, 2021), "
1:1 risk/reward ratio requires greater than 50% win rate for profitability.
1:2 risk/reward ratio requires greater than 33% win rate for profitability.
1:3 risk/reward ratio requires greater than 25% win rate for profitability.
"

This implies that adhering to the first two principles and maintaining a 1:3 risk-to-reward ratio in accordance with the third principle enables a trader to execute around 100 trades. Should the trader achieve 26 or more successful trades, profitability is assured. Furthermore, under the law of large numbers, there is a likelihood of approximately 50 trades being profitable, even with a random selection of the direction of the price movement.

It also implies that any trading system which can provide a winning rate of more than 50% will contribute to more profit margins.

However, in practice, the profit margins are expected to be lower than those projected in the ideal scenario. Factors such as brokerage fees, taxes, and price slippages will inevitably consume a portion of the profits.

5.1.4 This Study Aims to Supplement the Fundamental Principles

This research paper intends to supplement, not overhaul, the fundamental principles of trading. The proposed model is designed to help the trader have more probability trades, but the trader is expected not to discard the fundamental principles at any point in time

5.2 Formulating an Illustrative Trading Strategy Using the Model

According to (Anon, n.d.n), a trading strategy is a fixed plan that is designed to achieve a profitable return in the stock market. Typically, this aspect is highly subjective and varies according to the preferences and trading styles of individual traders. In this section, we will formulate an example trading strategy for the model proposed in this study, using the fundamental trading principles and the general nature of pivot levels outlined in the following sub-section.

5.2.1 Pivot Points as Resilient Barriers

(Person, 2004) asserted that whenever the price is near pivot levels, it is likely to bounce back or stall before breaking through eventually. This suggests that the pivot levels are not easy to pass through. If the proposed model predicts a pivot level which is so far away from the current price candles, and if there are many pivot levels to pass through, it’s not advisable to take those trades.

According to (Person, 2004), it is a good practice to place the stop-loss and target orders close to the pivot levels, but at a safe distance. So we will see how this can be applied to our result analysis.

5.2.2 Illustrative Trading Strategy Using the Proposed Model

With the knowledge of fundamental trading principles and the knowledge about pivot lines, an illustrative trading strategy can be formulated as:

1. When the proposed model predicts a pivot level that the prices are expected to touch in future, we will not straightaway take a trade, but we will wait until 2 conditions are met:
    a. There are no other pivot lines between the current candle and the predicted pivot level
    b. 1:3 Risk-to-Reward ratio can be established, if we place a target order close enough to the predicted level (but in between the current candle and the predicted level) and a stop-loss order after the next pivot line in the opposite direction, at a safe distance.
2. Depending on the distance between the entry point and the calculated stop-loss, and the trading capital available, we will do position sizing, in such a way that not more than 1% of the capital is risked with the stop-loss order.

3. Exit from the trade is when either the target order or stop-loss order is hit. Or when squaring off at the end of the trading session, if applicable
4. When a prediction is made, if there are one or more pivot levels in between the current candle and predicted level, we may try to take it as multiple individual trades considering each pivot level in between, if every such trade conforms to rules #1 and #2.

Let’s take an example to understand this trading strategy better.

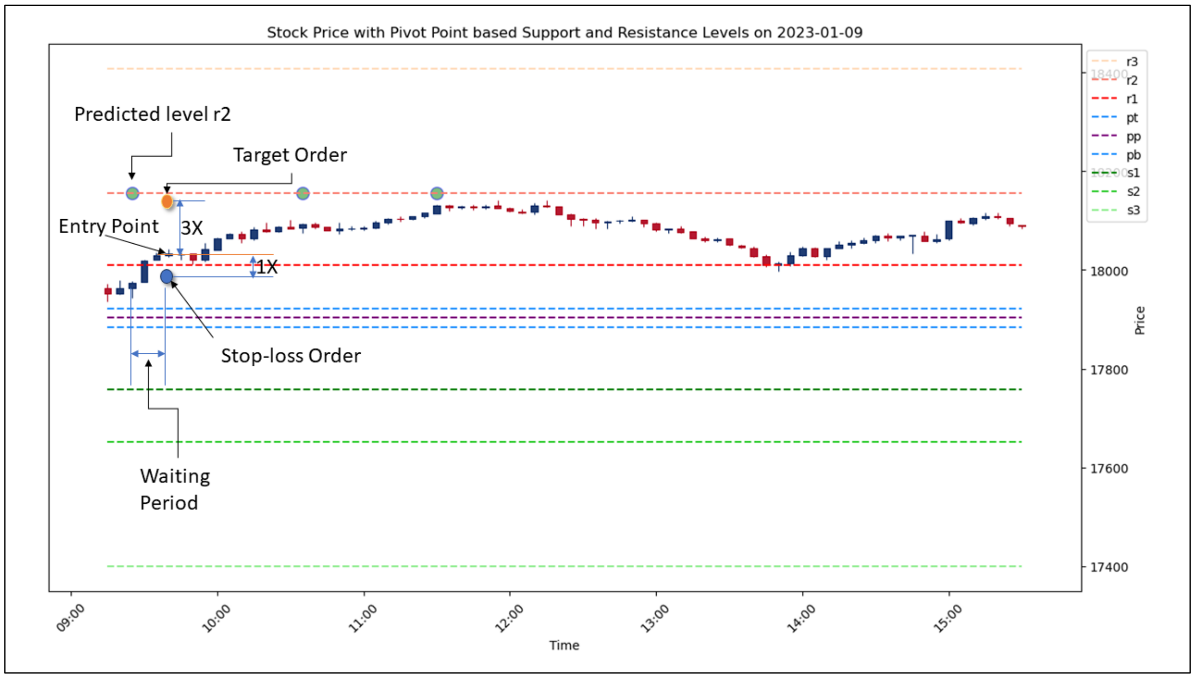


Figure 5.2.2 (a): Illustrative Trading Strategy

As is observed from Figure 5.2.2 (a), the proposed model predicted the level as ‘r2’ at 9:25. But when we check the candle at that time, it is still below the resistance level ‘r1’, so we should not take a trade according to our trading strategy. Instead, we will wait to see if the prices are going to break the level r1. The next candle at 9:30 broke the level ‘r1’ and closed above it. But the pivot line ‘r1’ is still going through the candle, so we will wait for more. The subsequent candle at 9:35 met this condition, but it failed to establish a 1:3 Risk-to-Reward ratio if we placed the stop-loss order below the pivot level ‘r1’.

All rules in the example trading strategy met at only 9:40. There was an entry point where a 1:3 risk-to-reward ratio was established by placing the stop-loss order well below the pivot level ‘r1’ and placing the target order just below the predicted level ‘r2’. So this is identified as the correct point to enter into the trade.

Now let’s take an example of the prediction level being too far away – meaning there are one or more pivot lines in between the predicted level and the current candle. Then we will try to split the trade into multiple smaller trades, each adhering to the rules of our trading strategy.

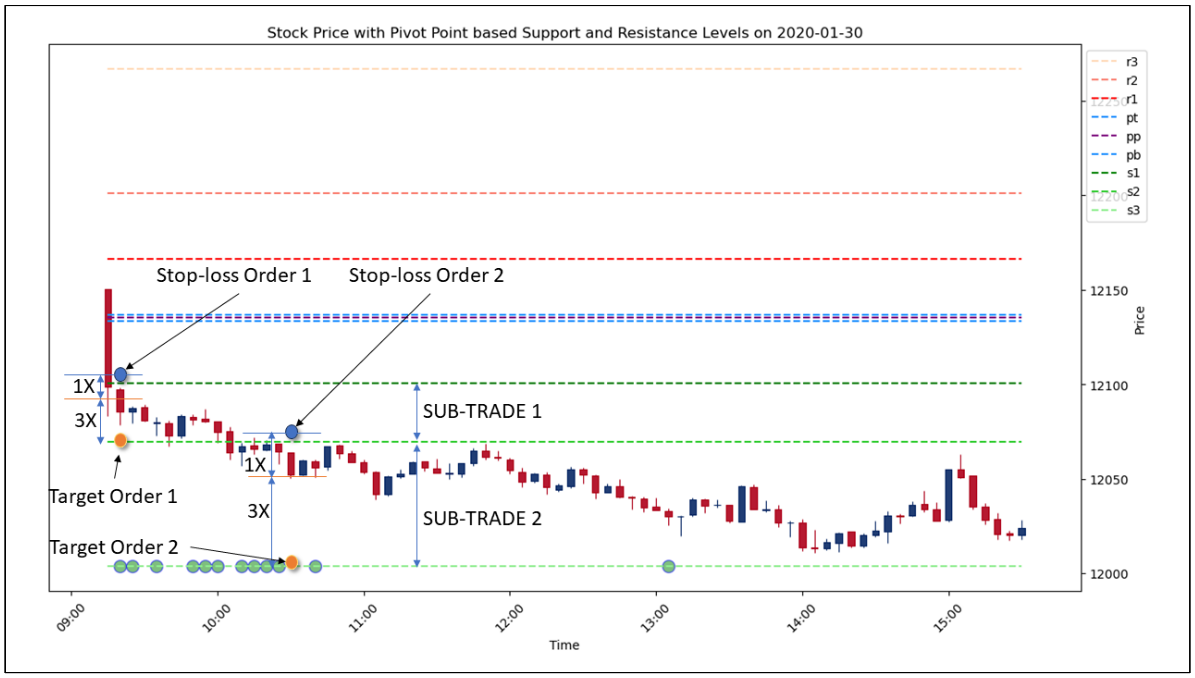


Figure 5.2.2 (b): Illustrative Trading Strategy – Splitting trades into multiple smaller trades

In Figure 5.2.3 (b), the proposed model made its first prediction of pivot level ‘s3’ at 9:20. But as we can see, there is another level ‘s2’ between the candle at that time and the predicted level ‘s3’. So we will split these into 2 sub-trades and apply the rules of the trading strategy to each of them.

In sub-trade 1, we will try to keep the target close to the in-between level ‘s2’ and the stop-loss above ‘s1’ which is the pivot level in the opposite direction. Only if we achieve positive results with this trade, will we attempt the sub-trade 2. In sub-trade 2, we will keep the target as ‘s3’ and stop-loss above ‘s2’. Each of these sub-trades has to adhere to every rule of the trading strategy.

5.3 Discussing Model Outcomes Based on Trading Strategy

In this section, we will analyze the outcomes of the proposed model from Chapter IV: RESULTS, in accordance with the trading strategy formulated.

5.3.1 Analysis of Positive Result 1

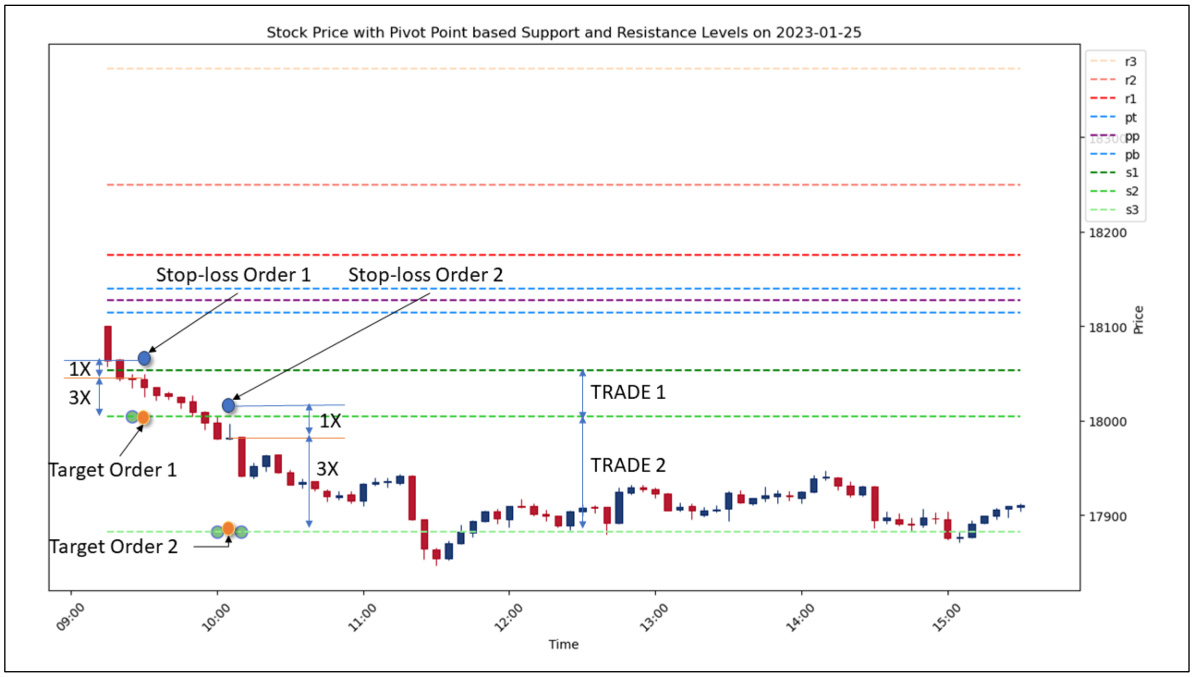


Figure 5.3.1: Analysis of Positive Result 1

As we can see, there are 2 separate trades possible on this day, as the proposed model predicted 2 levels – first on level ‘s2’ at 9:25, and then another 2 on level ‘s3’ at 10:00 and 10:10 respectively.

For the first trade, our entry is at 9:30, when the price reaches a point where our trading strategy rules are met. At 9:50, our target order would be hit and we could exit with a profit.

For the second trade, our entry is at 10:05 as the price conformed to our trading strategy rules. Again we could make a profit when the candles at 11:30 hit the target order.

5.3.2 Analysis of Positive Result 2

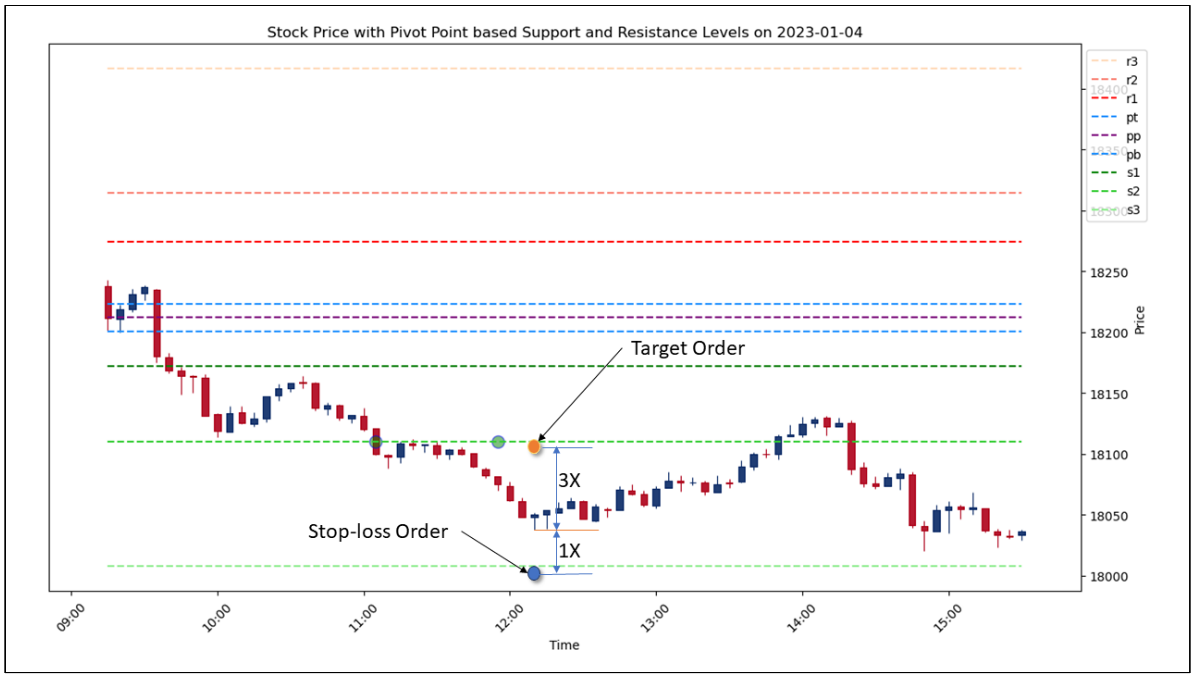


Figure 5.3.2: Analysis of Positive Result 2

The model predicted the same level ‘s2’ two times – at 11:05 and 11:55. As soon as the prediction happened at 11:05, we started looking for a correct entry point which is as per our trading strategy, but we had to wait till 12:10. We made the entry at 12:10, keeping the stop-loss under ‘s3’ and the target just below ‘s2’. We could take a profit when the target order is hit at 13:55.

5.3.3 Analysis of Positive Result 3

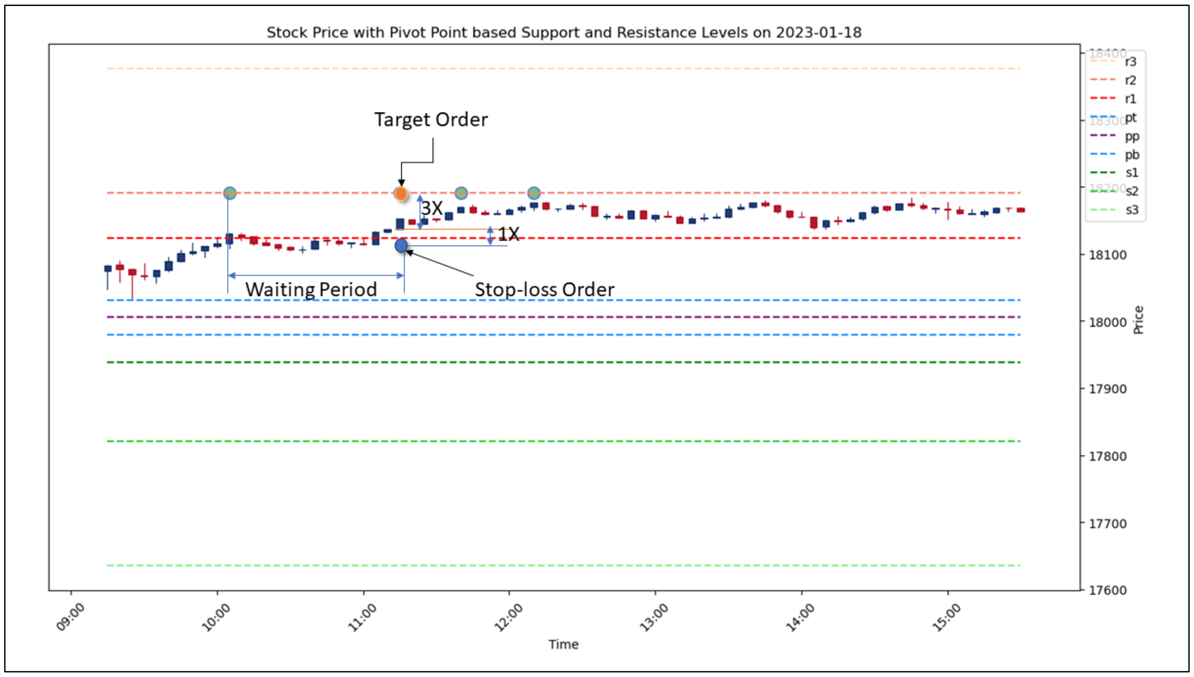


Figure 5.3.3: Analysis of Positive Result 3

In this result as well, there was a long waiting period from the time of the first prediction at 10:05 and the trading strategy conditions being met at 11:15. Once the conditions were met, we took the trade, kept the target order just below the predicted level ‘r2’ and placed the stop-loss order at a safe distance below the level in the opposite direction ‘r1’. And it is observed that the target order was hit at 14:45 and we could exit with a 3X profit than the risk taken.

5.3.4 Analysis of Negative Result 1

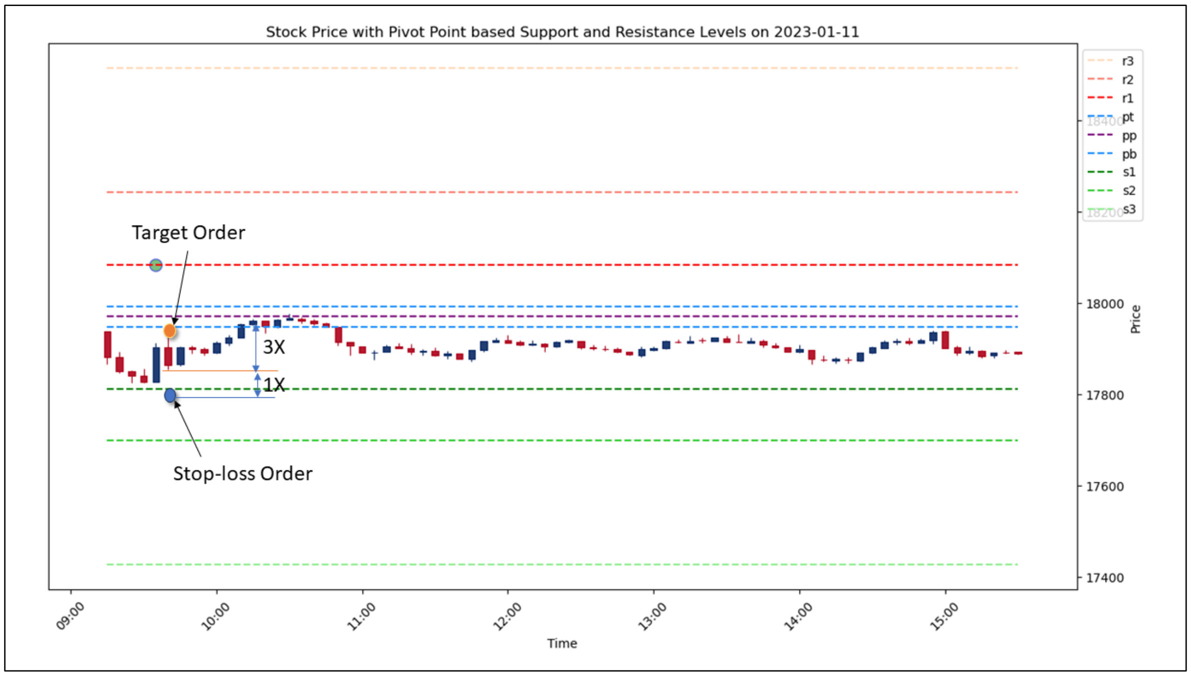


Figure 5.3.4 Analysis of Negative Result 1

Upon initial examination, this trade appears to be a loss, given the model's incorrect prediction. Yet, it serves as a quintessential case of the benefits derived from adhering strictly to our trading strategy, demonstrating the importance of following each guideline meticulously.

As per the trading strategy, if there are one or more pivot levels between the predicted level and the current candle, it is asserted to look for multiple sub-trades. As soon as the candle value drops to a suitable level that conforms to the 1:3 Risk-to-reward rule at 9:40, we took the sub-trade. This sub-trade should keep the target just below the first intermediate level starting from the candle towards the predicted level, here in this case ‘pb’, and place the stop-loss order just below ‘s1’. As we can see, this sub-trade yielded positive results when the price reached ‘pb’ at 10:10.

As per the trading strategy, we should look for the next sub-trades, provided that they all conform to all of the trading strategy rules. But in this case, there were no sub-trades which could correctly match the trading strategy. So we will exit the trade with profit from the first sub-trade.

5.3.5 Analysis of Negative Result 2

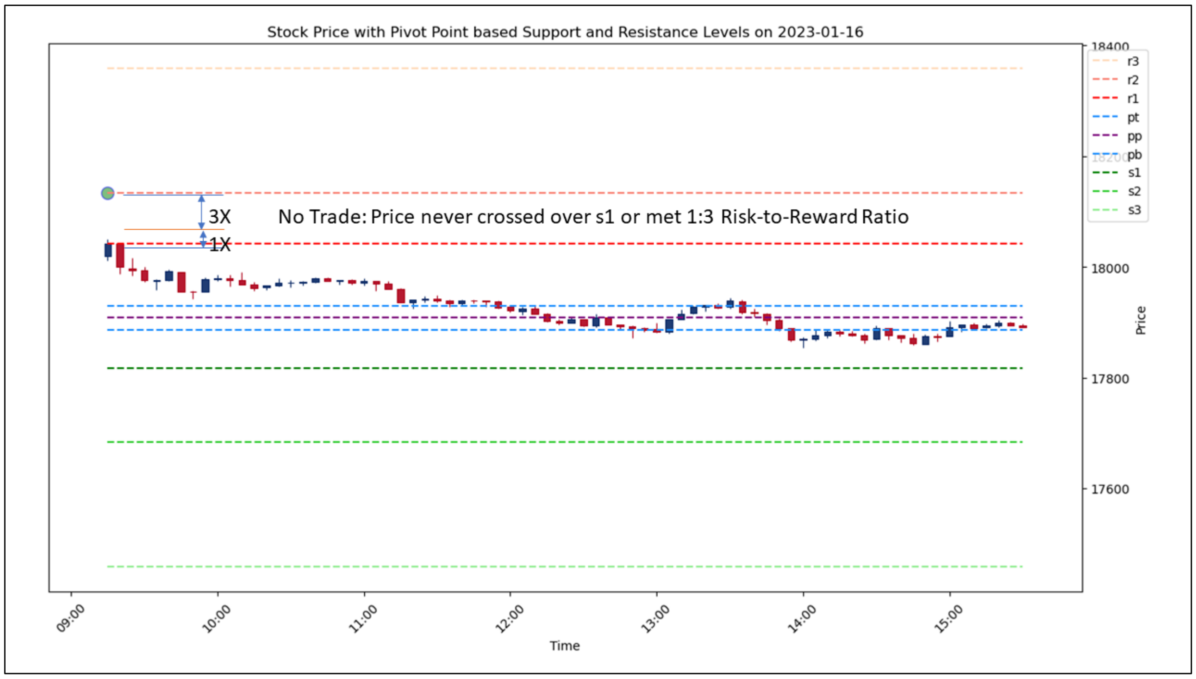


Figure 5.3.5 Analysis of Negative Result 2

In this scenario, the model predicted the wrong level, and at no point in time, the price could go above the ‘r1’ pivot level. So essentially, we cannot take any trades because they won’t match our trading strategy. But it is to be noted that there is no loss either, just because the trading strategy was followed meticulously.

5.3.6 Analysis of Negative Result 3

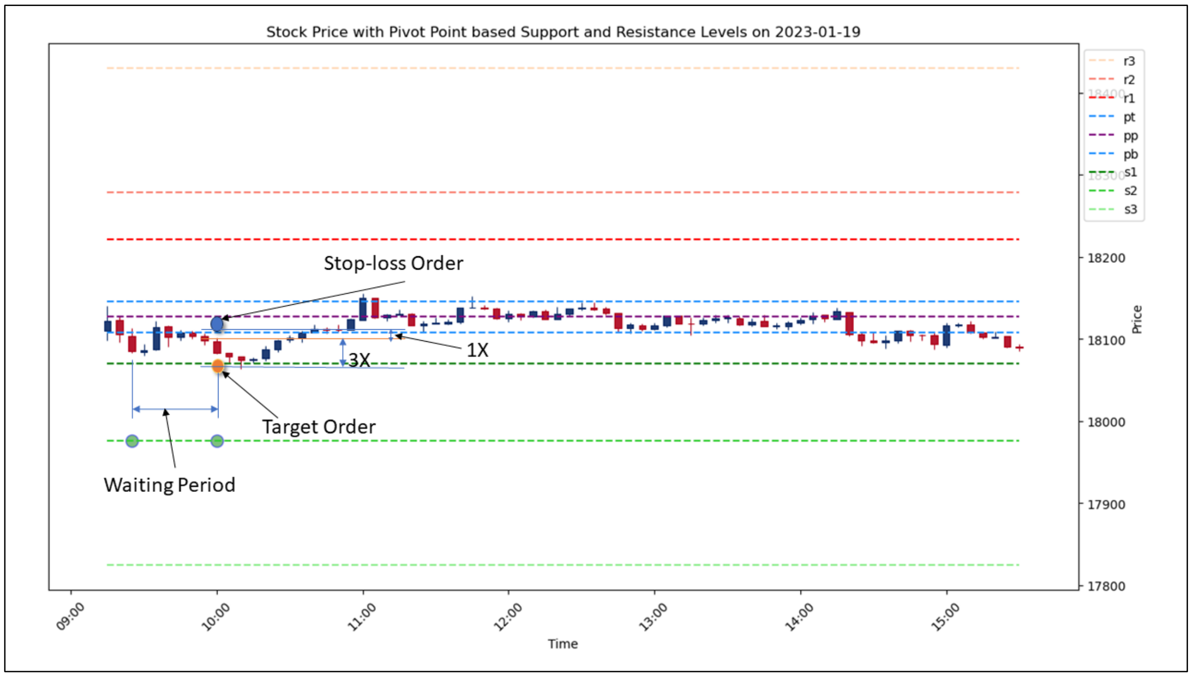


Figure 5.3.6: Analysis of Negative Result 3

In this example also, it can be observed how adhering strictly to the trading strategy at all times can result in profit even with a wrong prediction.

The model predicted ‘s2’ at 9:25, and since there is ‘s1’ in between the current candle and the predicted level, we can not take a trade on the predicted level, instead, we have to look for sub-trades.

It can be mistaken as we will get into a losing trade at 9:35 since it matches the 1:3 Risk-to-Reward ratio condition, but if we check our trading strategy guidelines correctly, we cannot take a trade if there is any level in between the candle and the sub-trade’s target level. Here at 9:35, the candles are piercing the level ‘pb’, essentially making it come between that particular candle and ‘s1’. We have to wait until the candles close above ‘pb’ and meet the 1:3 Risk-to-Reward ratio. That opportunity came only at

9:55, and we took the trade placing the stop-loss order above ‘pb’ and target order just below ‘s1’. This sub-trade yielded positive results at 10:10.

As per the trading strategy, the next sub-trade is not possible as the price never closed below ‘s1’. But still, we could successfully trade with a net profit by meticulously following the trading strategy.

CHAPTER VI:
SUMMARY, IMPLICATIONS, AND RECOMMENDATIONS

6.1 Summary

In this dissertation, the application of artificial intelligence in financial trading is analyzed in detail, emphasizing its potential to predict pivot levels in trading sessions. The summary of the findings is listed below:

6.1.1 Data Preprocessing Steps

1. Combined ‘Date’ and ‘Time’ fields to form a Pandas DateTime Object and saved it as the column ‘DateTime’. This is set as the index of the data frame
2. Dropped the ‘Date’, ‘Time’ and ‘Unknown’ columns
3. Filtered out rows which are not within the trading window of 9:15 AM IST and 3:30 PM IST
4. Resampled and aggregated the rows to 5-minute candle data
5. Dropped all the rows with NaN values
6. Pivot point-based support and resistance levels are calculated and placed under 9 new columns - 'pp', 'pb', 'pt', 'r1', 's1', 'r2', 's2', 'r3', 's3'
7. The following different technical indicators and candle patterns are added as columns of this Pandas data frame. Please refer to Appendix A for their detailed descriptions.
    a. RSI
    b. EMA7
    c. EMA14
    d. EMA_CROSSOVER_7AND14
    e. Momentum

f. ATR
g. ADX, Plus DI and Minus DI
h. Williams' %R
i. Doji
j. Marubozu
k. Hammer
l. Hanging Man
m. Engulfing
n. Harami
o. Morning Star
p. Evening Star
q. 3 White Soldiers
r. 3 Black Crows

8. Marked the next level, among the 9 pivot levels, the future candles are going to touch under the column ‘NextLevel’. This will be used as the output variable while training the AI model. This is one of the most important pieces of this research study. Please refer to the Python code in Appendix C to understand how this is calculated.

6.1.2 Model Architecture

Proposed the following Model Architecture to create a system to predict pivot levels as output.

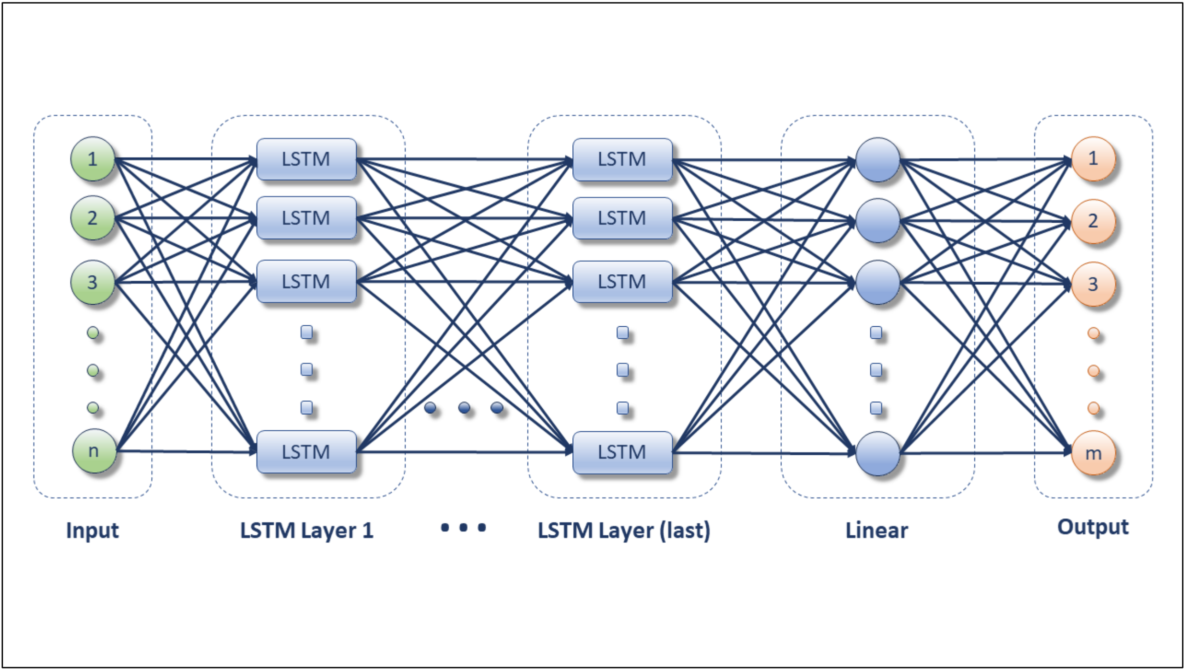


Figure 6.1: Model Architecture

6.1.3 Model Evaluation Metrics

This study recommended the following metrics to evaluate the models trained using this architecture:

1. Prediction Accuracy
2. Prediction Usefulness
3. Visual Evaluation and the need for the same

6.1.4 Optimal Configuration for the Proposed Model

After conducting a huge number of rigorous experiments, this study outlines the optimal configuration for training such systems, which is given below.

Table 6.1: Optimal Training Configuration

Configuration Parameter	Value
Training Dataset Size	1-year dataset
LSTM Hidden Dimensions	100
LSTM Number of Layers	1
Training Epochs	1000
Input Features	Open, High, Low, Close, RSI, EMA_CROSSOVER_7AND14

6.1.5 Versatility of the Proposed Model

This research asserted that the proposed model works well across different types of stock assets like indices and individual stocks, and also that the proposed model is effective for different timeframes for day, swing or positional trading.

6.1.6 Illustrative Trading Strategy Using the Proposed Model

Finally, this research also formulated an illustrative trading strategy using the proposed model and fundamental principles of trading.

1. When the proposed model predicts a pivot level that the prices are expected to touch in future, we will not straightaway take a trade, but we will wait until 2 conditions are met:
    a. There are no other pivot lines between the current candle and the predicted pivot level
    b. 1:3 Risk-to-Reward ratio can be established, if we place a target order close enough to the predicted level (but in between the current candle and the predicted level) and a stop-loss order after the next pivot line in the opposite direction, at a safe distance.

2. Depending on the distance between the entry point and the calculated stop-loss, and the trading capital available, we will do position sizing, in such a way that not more than 1% of the capital is risked with the stop-loss order.
3. Exit from the trade is when either the target order or stop-loss order is hit. Or when squaring off at the end of the trading session, if applicable
4. When a prediction is made, if there are one or more pivot levels in between the current candle and predicted level, we may try to take it as multiple individual trades considering each pivot level in between, if every such trade conforms to rules #1 and #2.

6.2 Implications

This research study outlined how to come up with a Long Short-Term Memory-based system to forecast the pivot levels that the future candles are going to touch. As we saw during the literature review, how important pivot point-based levels are for day trading and trading in general, an Artificial Intelligence-based system predicting these levels can be of great help to the traders. This enables individual day traders to get into more profitable trades and also enables them to have efficient risk management techniques. It could significantly benefit the algorithmic traders, to easily build a pipeline to continuously build such AI models without human intervention and to automate their trading with the help of the proposed model and trading strategies. This study could also aid big financial institutions engaged in having millions of financial trading transactions per month, by taking the proposed model as another indicator for automatically getting trading signals, and also to take a large number of automated trades conforming to the trading strategy and fundamental principles. Finally, this is helpful for researchers trying to bring state-of-the-art artificial intelligence-based models into financial trading.

6.3 Recommendations for Future Research

This research considered a total of 18 features including the major technical indicators and candlestick patterns, apart from the price information to train the proposed model. To select the optimal mix of these, a step-by-step process of taking the best configuration from a set of experiments was followed but tried out all the possible combinations. So future studies could try out other different combinations of these same technical indicators and candlestick patterns or even try out new ones following the same methods that were proposed in this thesis.

This study developed individual models for different types of stock assets and individual models. An area of future research could be combining multiple models into one generic model.

Another area that could be explored by future researchers is finding different ways to mark the target variable column ‘NextLevel’.

Trying out new metrics to evaluate the models or developing new ones could also possibly be another area of future research.

6.4 Conclusion

This thesis presents an in-depth examination of applying artificial intelligence into the domain of financial trading, focusing on its ability to forecast the pivot levels of the trading sessions. This study started by reviewing the literature to understand the previous studies in the field and to pinpoint the research gaps in this domain.

A dataset with one-minute stock market data in the National Stock Exchange is collected and preprocessed to mark the pivot levels along with different technical indicators and candle patterns. A novel way of mapping these pivot levels into a single column was devised so that an artificial intelligence-based model could be trained on this dataset with this new column as the output.

Based on the literature review findings that the Long Short-Term Memory networks perform strongly on financial trading data, this study proposed a Model Architecture to create a system to predict pivot levels as output.

This research also outlined the Metrics to evaluate the models trained using this architecture, including a custom Metric ‘Prediction Usefulness’ score which was introduced to examine the real-world practicality of the generated model.

Through 44 rigorous experiments examining different aspects of the proposed model, this study outlines the optimal configuration for training such systems.

By conducting another 6 experiments, this study gives insights that the proposed model works well for both indices and individual stocks, and also that the proposed model is effective for different timeframes and types of trading.

Finally, this research also formulated an illustrative trading strategy which ensures that the majority of the trades taken using this proposed model can be profitable.

APPENDIX A

INPUT FEATURES/COLUMNS AND THEIR DESCRIPTIONS

As explained by (Foot, n.d.), a candlestick is a type of price chart that shows the opening, high, low and closing prices for the given period. In addition, it is reported that candlesticks are the most popular way to quickly analyse price action and they offer much more information visually than the line charts. Furthermore, it is noted that the candlesticks can be used to analyse the price action over any time frame.

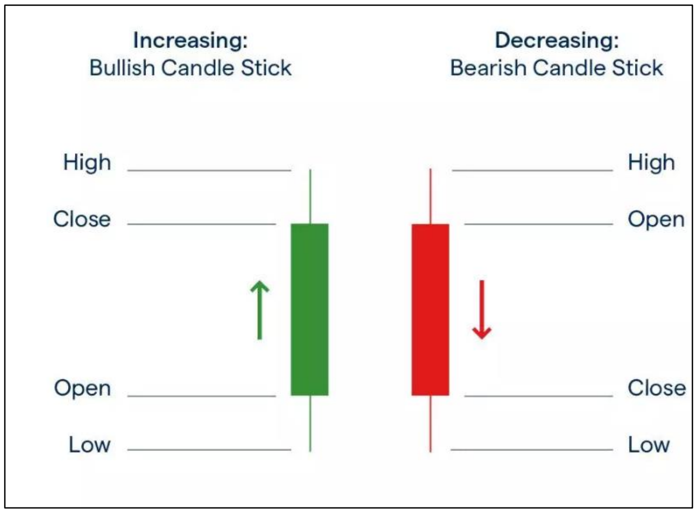


Figure Appendix A (a): Candle's Open, High, Low and Close values. Source: (Foot, n.d.)

The following are the input features used in this study along with their descriptions.

1. Open

   This is the opening price of the candle.

   As shown in Figure Appendix A (a), this is the bottom of the candle body, if it is's bullish candle. And it will be the top of the candle body if it's a bearish candle.

2. High

   This is the highest price of the candle. For both bullish and bearish candles, this will be the tip of the top wick.

3. Low

   This is the lowest price of the candle. It is the tip of the bottom wick for both bullish and bearish candles.

4. Close

   This is the closing price of the candle. It will be the top of the candle body for bullish candles and the bottom of the candle body for bearish candles.

5. Volume

   The volume associated with a candle is the total number of traded shares or contracts during the period of that candle. Volume is not available for the index funds. So in this study, it is used only for experiments related to the Stocks.

6. RSI

   This represents the Relative Strength Index, which compares the asset’s gains over its losses over a specific duration (Anon, n.d.f). It is also stated there that if the gains significantly exceeded losses over that period, then the asset is considered overbought, and if the losses significantly exceeded the gains, then it’s considered oversold. The equations to calculate RSI are also given in the same article:

Relative Strength (RS) = (Sums of gains on up days for 14 days/14) / (Sums of losses on down days for 14 days/14)

RSI = 100 – (100/(1+RS))

RSI for this study is calculated using the closing price and a period of 14.

RSI is implemented in Python using TA-Lib (Anon, n.d.i), as follows:

```
df['RSI'] = ta.RSI(df['Close'], 14)
```

7. EMA7

   As explained by (Anon, n.d.d), the Exponential Moving Average is a type of technical indicator which highlights the trend shown by the asset. It’s a type of moving average in which the weight on recent data is more, compared to older data points. It’s a lagging indicator, which means it changes after some time the actual price has been changed.

   The equation to calculate EMA is also given by the same author (Anon, n.d.d), as follows:

   EMA = K x (Current Price – Previous EMA) + Previous EMA

   where,

   K is the weighting factor for the EMA and is calculated as K = 2/(n+1) where n is the selected time period.

   EMA7 in this study is the Exponential Moving Average calculated using the closing price and a period of 7.

This is calculated with the help of the TA-Lib library (Anon, n.d.i) in Python as follows,

```
df['EMA7'] = ta.EMA(df['Close'], 7)
```

8. EMA14

   This is the Exponential Moving Average calculated using the closing price and a period of 14

   This is calculated with the help of the TA-Lib library (Anon, n.d.i) in Python as follows,

   ```
   df['EMA14'] = ta.EMA(df['Close'], 14)
   ```

9. EMA_CROSSOVER_7AND14

   This is the cross-over of the EMA7 line compared to the EMA14 line. This value will be positive when the EMA7 line is above the EMA14 line which indicates a bullish trend. When it is negative, it indicates a bearish trend.

   'EMA_CROSSOVER_7AND14' is calculated as,

   ```
   df['EMA_CROSSOVER_7AND14'] = df['EMA7'] - df['EMA14']
   ```

   where,
   df is the pandas dataframe,
   df['EMA7'] is the column in the data frame with the values of EMA7, and

df['EMA14'] is the column in the data frame with the values of EMA14

10. Momentum

    Momentum in technical analysis is the rate of change of an asset (Anon, n.d.f). The equation to calculate momentum is also stated in the same article and is given below:

    Momentum = Current Price – Earlier Price

    where,

    Earlier price is typically the price before 14 days, though any earlier price can be chosen.

    In this study, Momentum is calculated using the Close price with a period of 14 days.

    It is implemented in Python using TA-Lib (Anon, n.d.i), as follows:

    ```
    df['MOM'] = ta.MOM(df['Close'], 14)
    ```

11. ATR

    Average True Range is a technical indicator that measures the volatility, it doesn't indicate the price direction (Anon, n.d.c). (Anon, n.d.c) also outlines the method to calculate the ATR, as follows,

    Current ATR = [(Prior ATR x 13) + Current TR] / 14

    where True Range (TR) is defined as the greatest value among the following 3:

a. Current High less the current Low
b. Current High less the previous Close (absolute value)
c. Current Low less the previous Close (absolute value)

12. ADX, Plus DI and Minus DI

Average Directional Index, Plus Directional Indicator and Minus Directional Indicator represent a group of technical indicators used to assess the strength of the trend and its directions respectively (Anon, n.d.b).

Step-by-step ADX calculation is outlined by (Anon, n.d.b), as given below:

a. Calculate the True Range (TR), Plus Directional Movement (+DM) and Minus Directional Movement (-DM) for each period.
b. Smooth these periodic values using Wilder's smoothing techniques. These are explained in detail in the next section.
c. Divide the 14-day smoothed Plus Directional Movement (+DM) by the 14-day smoothed True Range to find the 14-day Plus Directional Indicator (+DI14). Multiply by 100 to move the decimal point to two places. This +DI14 is the green Plus Directional Indicator line (+DI) that is plotted along with the ADX line.
d. Divide the 14-day smoothed Minus Directional Movement (-DM) by the 14-day smoothed True Range to find the 14-day Minus Directional Indicator (-DI14). Multiply by 100 to move the decimal point to two places. This -DI14 is the red Minus Directional Indicator line (-DI) that is plotted along with the ADX line.

e. The Directional Movement Index (DX) equals the absolute value of +DI14 less -DI14 divided by the sum of +DI14 and -DI14. Multiply the result by 100 to move the decimal point over two places.

f. After all these steps, it is time to calculate the Average Directional Index (ADX) line. The first ADX value is simply a 14-day average of DX. Subsequent ADX values are smoothed by multiplying the previous 14-day ADX value by 13, adding the most recent DX value and dividing this total by 14.

13. Williams' %R

Williams’ %R is a type of oscillator which compares the most recent close to the high of the window period and is used in a range market as a confirmation signal, with values exceeding -20 indicating an overbought condition while values less than -80 denotes an oversold situation (Anon, n.d.f). The same article gives the equation to find the Williams’ %R as follows:

Williams’ %R = ((Highest High – Close Price) / (Highest High – Lowest Low)) x 100

It is implemented in this study for a period of 14 as follows, with the help of TA-Lib (Anon, n.d.i):

```
df['WILLR'] = ta.WILLR(df['High'], df['Low'], df['Close'], 14)
```

14. Doji

Doji is a single candlestick pattern in which the open and close prices are almost or exactly equal so that the body appears to be a thin line – typically less than 5% of the total range for that period (Foot, n.d.). The author also notes that Doji could be an indication of the upcoming reversal since the bulls and bears have cancelled each other out during this candle period.

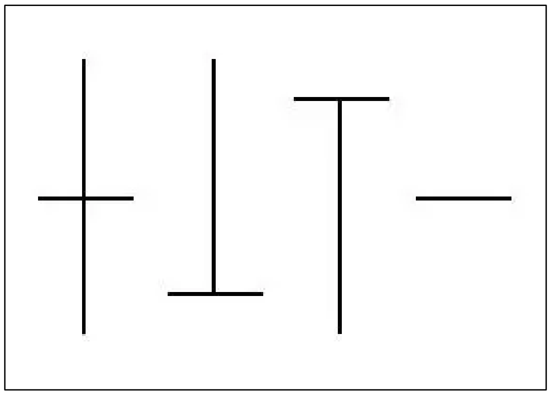


Figure Appendix A (b): Different types of Doji candlestick patterns.
Source: (Foot, n.d.)

In this study, the detection of the Doji pattern is implemented using TA-Lib (Anon, n.d.i), as follows:

df['CDLDOJI'] = ta.CDLDOJI(df['Open'], df['High'], df['Low'], df['Close'])

15. Marubozu

Marubozu is another single candlestick pattern in which the candle has no wicks, which is a clear indication of a strong bullish or bearish trend, depending on the colour of the candle (Foot, n.d.).

Figure Appendix A (c): Marubozu candlestick pattern with bullish and bearish candles. Source: (Foot, n.d.)

Detection of Marubozu candles is implemented using TA-Lib (Anon, n.d.i), as follows:

df['CDLMARUBOZU'] = ta.CDLMARUBOZU(df['Open'], df['High'], df['Low'], df['Close'])

16. Hammer

Hammer pattern is a candle which has a long wick below a short body with small or no wick above, and is typically formed after a downtrend (Foot, n.d.). It is also stated that while the bullish reversal might be on the card, it is advisable to wait for confirmation in the next candles.

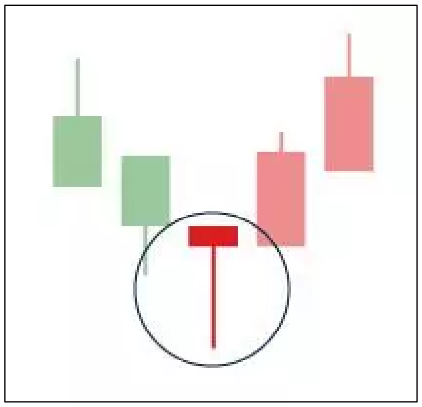


Figure Appendix A (d): Hammer candlestick pattern. Source: (Foot, n.d.)

Hammer pattern is implemented using TA-Lib (Anon, n.d.i), as follows:

df['CDLHAMMER'] = ta.CDLHAMMER(df['Open'], df['High'], df['Low'], df['Close'])

17. Hanging Man

A hanging man pattern looks identical to the hammer, but the difference is that it is coming up after a bullish trend, possibly as a trend reversal (Foot, n.d.). It is also noted that, just like the hammer pattern, it's advisable to wait for confirmation in the next candles.

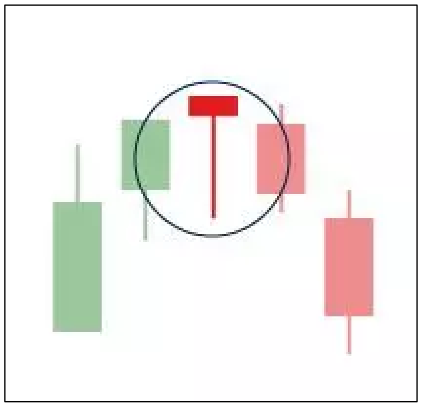


Figure Appendix A (e): Hanging Man candlestick pattern. Source: (Foot, n.d.)

Detecting the Hanging Man pattern in Python is done using the TA-Lib (Anon, n.d.i), in the following manner.

df['CDLHANGINGMAN'] = ta.CDLHANGINGMAN(df['Open'], df['High'], df['Low'], df['Close'])

18. Engulfing

An engulfing pattern is a double candlestick pattern, in which a candlestick is immediately followed by another larger one in the opposite direction (Foot, n.d.). Moreover, it is asserted that the direction of the future movement is expected to be the same as the direction of the second candle which is the larger of the 2 candles.

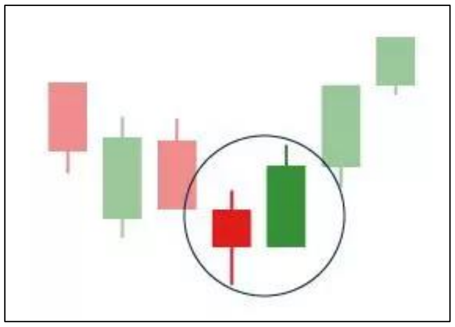


Figure Appendix A (f): Engulfing candlestick pattern. Source: (Foot, n.d.)

Using TA-Lib (Anon, n.d.i), the engulfing pattern is detected in the code as detailed below:

df['CDLENGULFING'] = ta.CDLENGULFING(df['Open'], df['High'], df['Low'], df['Close'])

19. Harami

Harami pattern is another double candlestick pattern in which the first candlestick is immediately followed by a much smaller one in the opposite direction (Foot, n.d.). Additionally, it is mentioned that the direction of the expected future movement is similar to the direction of the second candle, and also the smaller the second candle’s body is, the stronger the signal is.

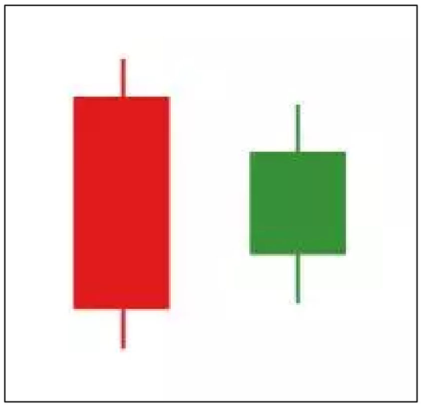


Figure Appendix A (g): Harami candlestick pattern. Source: (Foot, n.d.)

TA-Lib (Anon, n.d.i) is used to look for Harami patterns in the following way:
df['CDLHARAMI'] = ta.CDLHARAMI(df['Open'], df['High'], df['Low'], df['Close'])

20. Morning Star

Morning Star pattern is a triple candlestick pattern that predicts a bullish recovery after a downtrend, and it is formed with a large red candle, followed by a small candle and then a third candle with a large green body (Foot, n.d.).

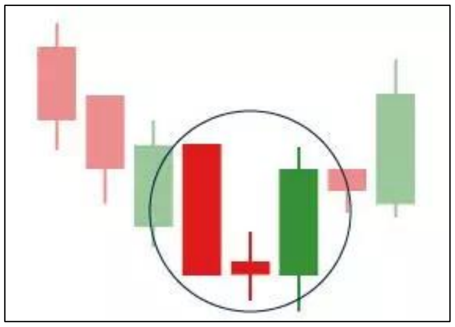


Figure Appendix A (h): Morning Star candlestick pattern. Source: (Foot, n.d.)

Detection of the Morning Star pattern in Python is done with the help of TA-Lib (Anon, n.d.i), as mentioned below:

df['CDLMORNINGSTAR'] = ta.CDLMORNINGSTAR(df['Open'], df['High'], df['Low'], df['Close'])

21. Evening Star

    This is another triple candlestick pattern which indicates a possible bearish downtrend after a bullish uptrend, and it consists of a large green candle followed by a small candle and then a large red candle (Foot, n.d.).

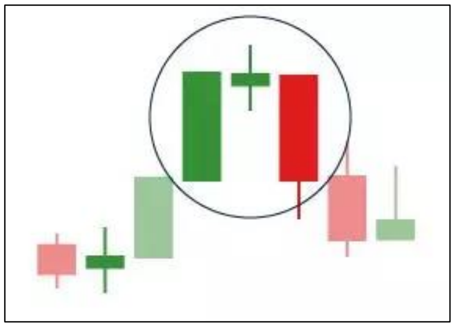


Figure Appendix A (i): Evening Star candlestick pattern. Source: (Foot, n.d.)

Evening Star pattern is implemented using TA-Lib (Anon, n.d.i) as follows:

df['CDLEVENINGSTAR'] = ta.CDLEVENINGSTAR(df['Open'], df['High'], df['Low'], df['Close'])

22. 3 White Soldiers

As the name indicates, the 3 white soldiers pattern is a triple candlestick pattern in which 3 consecutive green candles are progressively bigger whereas the last one has little to no wick (Foot, n.d.). Additionally, it is mentioned that this pattern predicts a bullish uptrend after a downtrend.

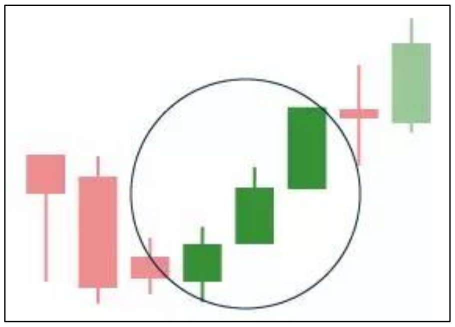


Figure Appendix A (j): 3 White Soldiers candlestick pattern. Source: (Foot, n.d.)

Implementation of 3 White Soldiers pattern with the help of TA-Lib (Anon, n.d.i) is given below:

df['CDL3WHITESOLDIERS'] = ta.CDL3WHITESOLDIERS(df['Open'], df['High'], df['Low'], df['Close'])

23. 3 Black Crows

This triple candlestick pattern is exactly opposite to 3 White Soldiers, consisting of 3 consecutive red candles that are progressively larger in body size and the third one should have a small or no wick (Foot, n.d.). Furthermore, it is noted that this pattern appears after an uptrend and is considered a strong indicator of a bearish downtrend.

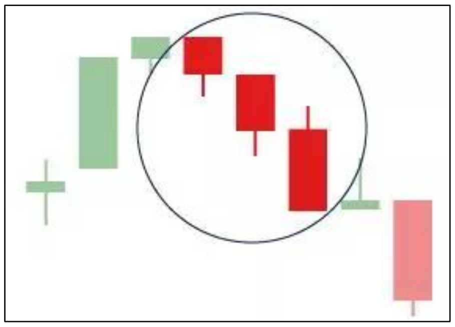


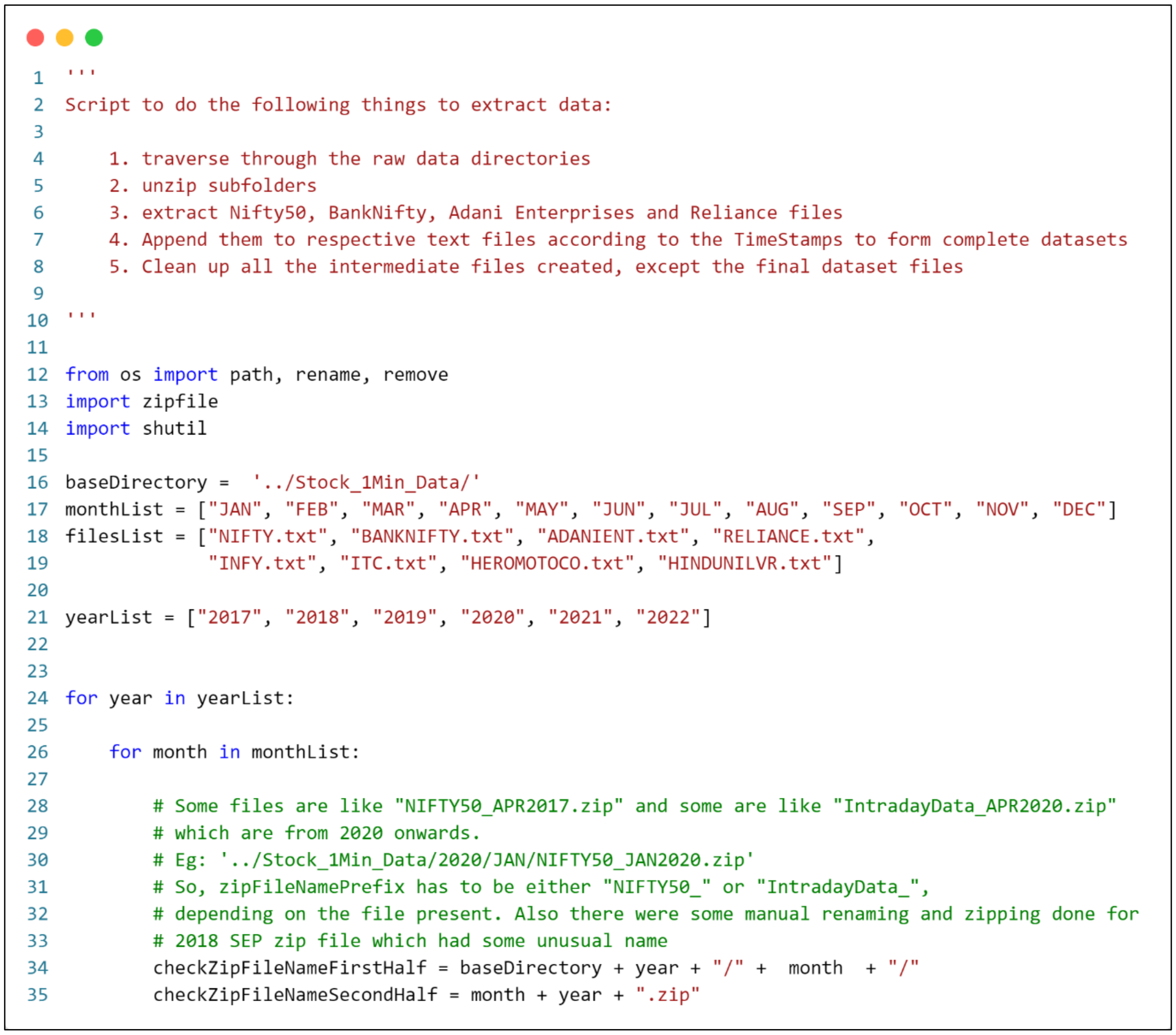
Figure Appendix A (k): 3 Black Crows candlestick pattern. Source: (Foot, n.d.)

3 Black Crows pattern is detected using TA-Lib (Anon, n.d.i) in the following manner:

df['CDL3BLACKCROWS'] = ta.CDL3BLACKCROWS(df['Open'], df['High'], df['Low'], df['Close'])

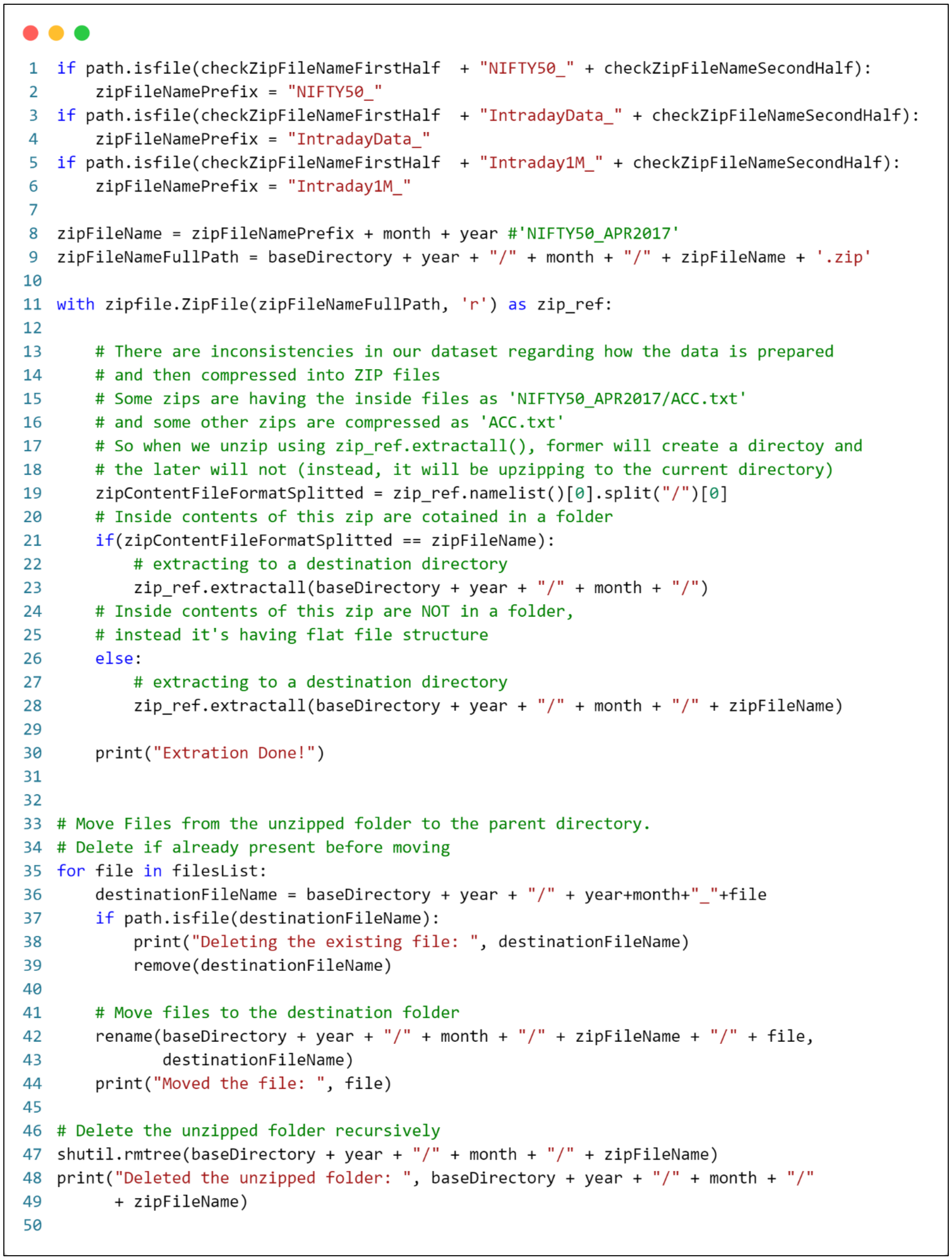
APPENDIX B

PYTHON SCRIPT: EXTRACT DATA FROM ONE-MINUTE DATASET



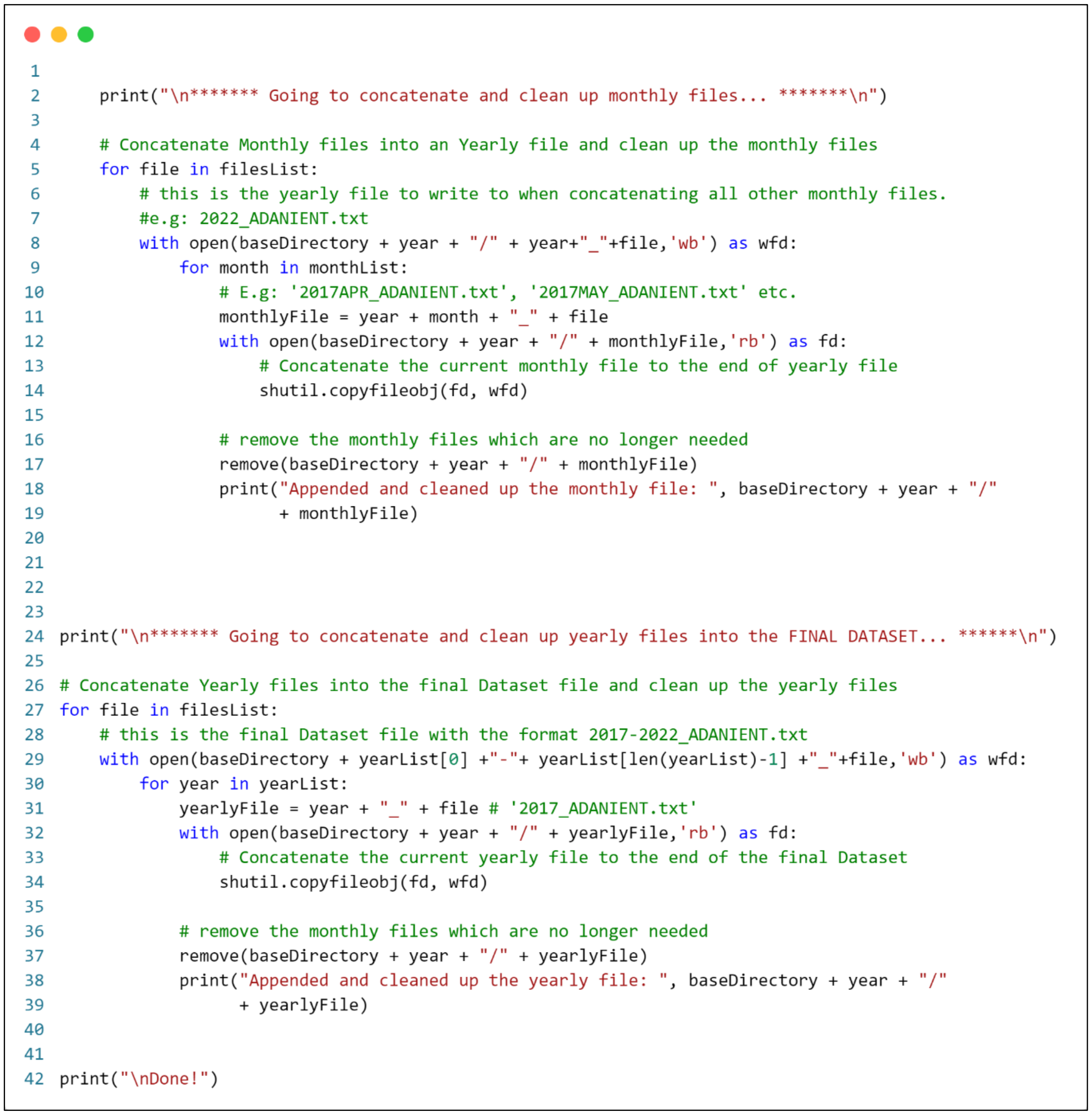
```python
'''
Script to do the following things to extract data:

    1. traverse through the raw data directories
    2. unzip subfolders
    3. extract Nifty50, BankNifty, Adani Enterprises and Reliance files
    4. Append them to respective text files according to the TimeStamps to form complete datasets
    5. Clean up all the intermediate files created, except the final dataset files

'''

from os import path, rename, remove
import zipfile
import shutil

baseDirectory =  '../Stock_1Min_Data/'
monthList = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"]
filesList = ["NIFTY.txt", "BANKNIFTY.txt", "ADANIENT.txt", "RELIANCE.txt",
             "INFY.txt", "ITC.txt", "HEROMOTOCO.txt", "HINDUNILVR.txt"]

yearList = ["2017", "2018", "2019", "2020", "2021", "2022"]


for year in yearList:

    for month in monthList:

        # Some files are like "NIFTY50_APR2017.zip" and some are like "IntradayData_APR2020.zip"
        # which are from 2020 onwards.
        # Eg: '../Stock_1Min_Data/2020/JAN/NIFTY50_JAN2020.zip'
        # So, zipFileNamePrefix has to be either "NIFTY50_" or "IntradayData_",
        # depending on the file present. Also there were some manual renaming and zipping done for
        # 2018 SEP zip file which had some unusual name
        checkZipFileNameFirstHalf = baseDirectory + year + "/" +  month  + "/"
        checkZipFileNameSecondHalf = month + year + ".zip"
```

Figure Appendix B (a): extractDataset.py – Part 1 of 3



```python
if path.isfile(checkZipFileNameFirstHalf  + "NIFTY50_" + checkZipFileNameSecondHalf):
    zipFileNamePrefix = "NIFTY50_"
if path.isfile(checkZipFileNameFirstHalf  + "IntradayData_" + checkZipFileNameSecondHalf):
    zipFileNamePrefix = "IntradayData_"
if path.isfile(checkZipFileNameFirstHalf  + "Intraday1M_" + checkZipFileNameSecondHalf):
    zipFileNamePrefix = "Intraday1M_"

zipFileName = zipFileNamePrefix + month + year #'NIFTY50_APR2017'
zipFileNameFullPath = baseDirectory + year + "/" + month + "/" + zipFileName + '.zip'

with zipfile.ZipFile(zipFileNameFullPath, 'r') as zip_ref:

    # There are inconsistencies in our dataset regarding how the data is prepared
    # and then compressed into ZIP files
    # Some zips are having the inside files as 'NIFTY50_APR2017/ACC.txt'
    # and some other zips are compressed as 'ACC.txt'
    # So when we unzip using zip_ref.extractall(), former will create a directoy and
    # the later will not (instead, it will be upzipping to the current directory)
    zipContentFileFormatSplitted = zip_ref.namelist()[0].split("/")[0]
    # Inside contents of this zip are cotained in a folder
    if(zipContentFileFormatSplitted == zipFileName):
        # extracting to a destination directory
        zip_ref.extractall(baseDirectory + year + "/" + month + "/")
    # Inside contents of this zip are NOT in a folder,
    # instead it's having flat file structure
    else:
        # extracting to a destination directory
        zip_ref.extractall(baseDirectory + year + "/" + month + "/" + zipFileName)

    print("Extration Done!")


# Move Files from the unzipped folder to the parent directory.
# Delete if already present before moving
for file in filesList:
    destinationFileName = baseDirectory + year + "/" + year+month+"_"+file
    if path.isfile(destinationFileName):
        print("Deleting the existing file: ", destinationFileName)
        remove(destinationFileName)

    # Move files to the destination folder
    rename(baseDirectory + year + "/" + month + "/" + zipFileName + "/" + file,
           destinationFileName)
    print("Moved the file: ", file)

# Delete the unzipped folder recursively
shutil.rmtree(baseDirectory + year + "/" + month + "/" + zipFileName)
print("Deleted the unzipped folder: ", baseDirectory + year + "/" + month + "/"
      + zipFileName)

```

Figure Appendix B (b): extractDataset.py – Part 2 of 3



```python

    print("\n******* Going to concatenate and clean up monthly files... *******\n")

    # Concatenate Monthly files into an Yearly file and clean up the monthly files
    for file in filesList:
        # this is the yearly file to write to when concatenating all other monthly files.
        #e.g: 2022_ADANIENT.txt
        with open(baseDirectory + year + "/" + year+"_"+file,'wb') as wfd:
            for month in monthList:
                # E.g: '2017APR_ADANIENT.txt', '2017MAY_ADANIENT.txt' etc.
                monthlyFile = year + month + "_" + file
                with open(baseDirectory + year + "/" + monthlyFile,'rb') as fd:
                    # Concatenate the current monthly file to the end of yearly file
                    shutil.copyfileobj(fd, wfd)

                # remove the monthly files which are no longer needed
                remove(baseDirectory + year + "/" + monthlyFile)
                print("Appended and cleaned up the monthly file: ", baseDirectory + year + "/"
                      + monthlyFile)




print("\n******* Going to concatenate and clean up yearly files into the FINAL DATASET... ******\n")

# Concatenate Yearly files into the final Dataset file and clean up the yearly files
for file in filesList:
    # this is the final Dataset file with the format 2017-2022_ADANIENT.txt
    with open(baseDirectory + yearList[0] +"-"+ yearList[len(yearList)-1] +"_"+file,'wb') as wfd:
        for year in yearList:
            yearlyFile = year + "_" + file # '2017_ADANIENT.txt'
            with open(baseDirectory + year + "/" + yearlyFile,'rb') as fd:
                # Concatenate the current yearly file to the end of the final Dataset
                shutil.copyfileobj(fd, wfd)

            # remove the monthly files which are no longer needed
            remove(baseDirectory + year + "/" + yearlyFile)
            print("Appended and cleaned up the yearly file: ", baseDirectory + year + "/"
                  + yearlyFile)


print("\nDone!")
```

Figure Appendix B (c): extractDataset.py – Part 3 of 3

APPENDIX C

DATA PREPROCESSING STEP: MARK NEXT LEVEL

Finding and marking the ‘NextLevel’ is one of the most important steps of this research study. NextLevel will be one among the 9 pivot point levels - pp, pb, pt, s1, s2, s3, r1, r2, r3. The core idea is summarised as follows:

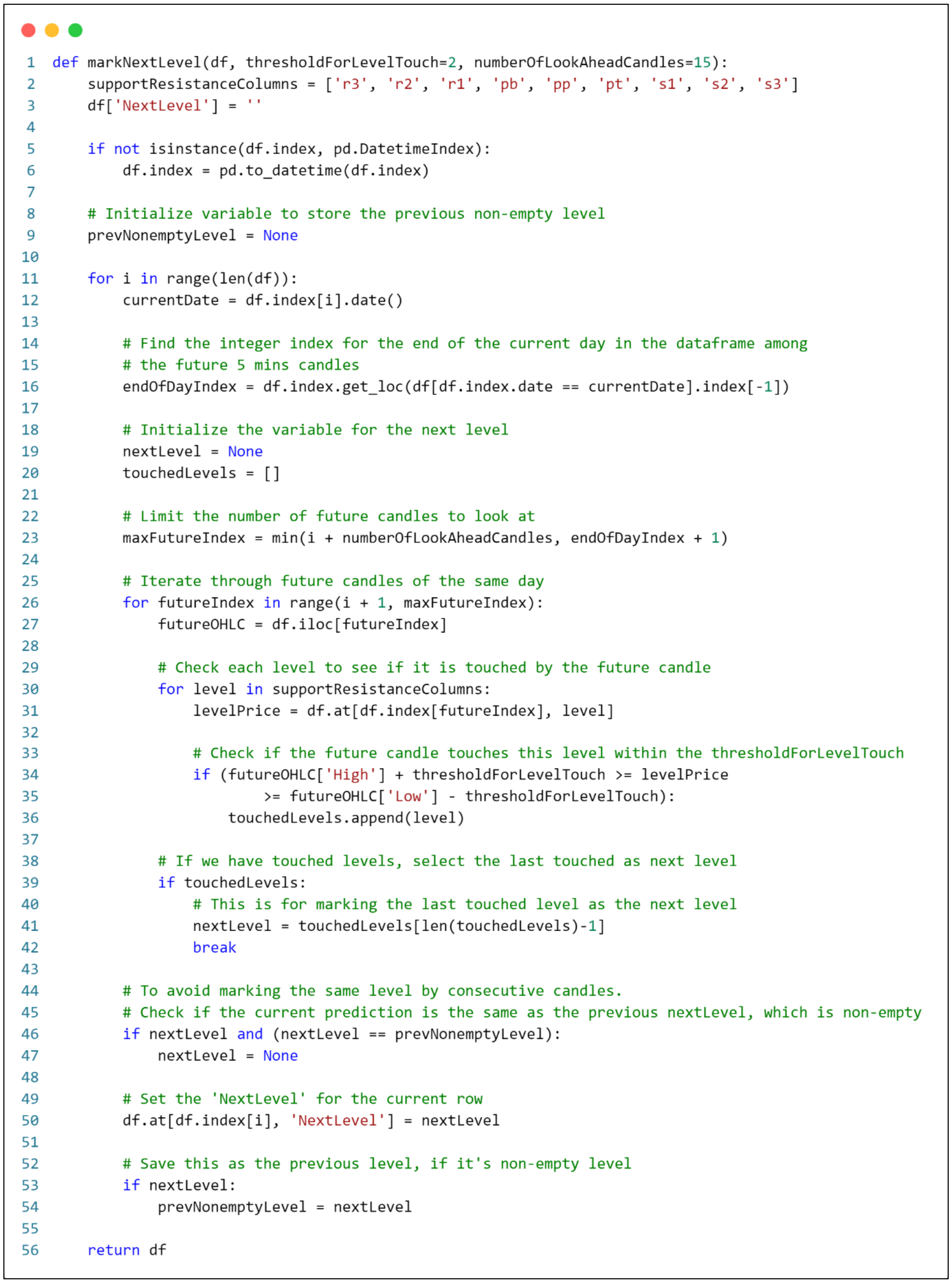
1. Take 5-minute candles per trading day
2. For each candle of a dataset, look for a certain number of future candles ahead, or until the end candle of that day is reached, and find out the levels the future candles are touching

   e.g.: s1, s2, s3
3. If there is more than one level, take the level which is farthest from the current candle and mark it as the NextLevel.

   e.g.: if the current candle is between pt and s1, then out of the 3 levels that the future candles touched, s3 would be the farthest one. So,

   NextLevel = s3

   On a side note, we could choose to take the nearest level instead of the farthest, but then the prediction could be late in the timeline and will be limited to the immediate level above or below the current candle.
4. If consecutive candles are going to mark the same level as the previously marked one, then keep only the first one as the NextLevel and the following consecutive ones are reset to ‘’.
5. A new column ‘NextLevel’ is appended to the pandas data frame and returned the data frame. This column ‘NextLevel’ will be the target variable while training the model.



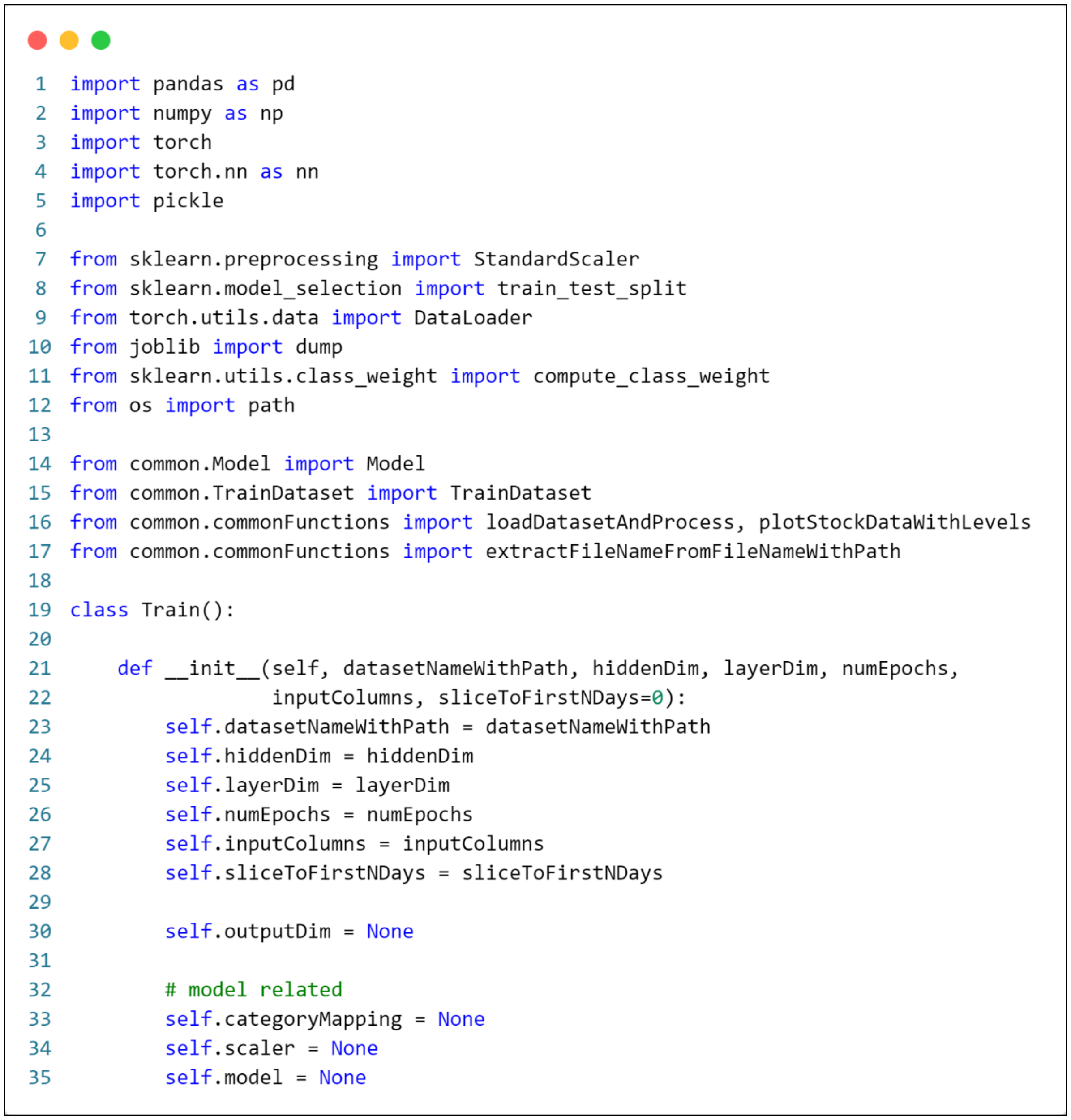
```python
def markNextLevel(df, thresholdForLevelTouch=2, numberOfLookAheadCandles=15):
    supportResistanceColumns = ['r3', 'r2', 'r1', 'pb', 'pp', 'pt', 's1', 's2', 's3']
    df['NextLevel'] = ''

    if not isinstance(df.index, pd.DatetimeIndex):
        df.index = pd.to_datetime(df.index)

    # Initialize variable to store the previous non-empty level
    prevNonemptyLevel = None

    for i in range(len(df)):
        currentDate = df.index[i].date()

        # Find the integer index for the end of the current day in the dataframe among
        # the future 5 mins candles
        endOfDayIndex = df.index.get_loc(df[df.index.date == currentDate].index[-1])

        # Initialize the variable for the next level
        nextLevel = None
        touchedLevels = []

        # Limit the number of future candles to look at
        maxFutureIndex = min(i + numberOfLookAheadCandles, endOfDayIndex + 1)

        # Iterate through future candles of the same day
        for futureIndex in range(i + 1, maxFutureIndex):
            futureOHLC = df.iloc[futureIndex]

            # Check each level to see if it is touched by the future candle
            for level in supportResistanceColumns:
                levelPrice = df.at[df.index[futureIndex], level]

                # Check if the future candle touches this level within the thresholdForLevelTouch
                if (futureOHLC['High'] + thresholdForLevelTouch >= levelPrice
                        >= futureOHLC['Low'] - thresholdForLevelTouch):
                    touchedLevels.append(level)

            # If we have touched levels, select the last touched as next level
            if touchedLevels:
                # This is for marking the last touched level as the next level
                nextLevel = touchedLevels[len(touchedLevels)-1]
                break

        # To avoid marking the same level by consecutive candles.
        # Check if the current prediction is the same as the previous nextLevel, which is non-empty
        if nextLevel and (nextLevel == prevNonemptyLevel):
            nextLevel = None

        # Set the 'NextLevel' for the current row
        df.at[df.index[i], 'NextLevel'] = nextLevel

        # Save this as the previous level, if it's non-empty level
        if nextLevel:
            prevNonemptyLevel = nextLevel

    return df
```

Figure Appendix C: Function to mark "Next Level"

This function takes in 3 parameters - the data frame, the threshold to define touching of a level and the number of candles to look ahead in future to detect this touching. And returns the data frame by adding a new column ‘NextLevel’

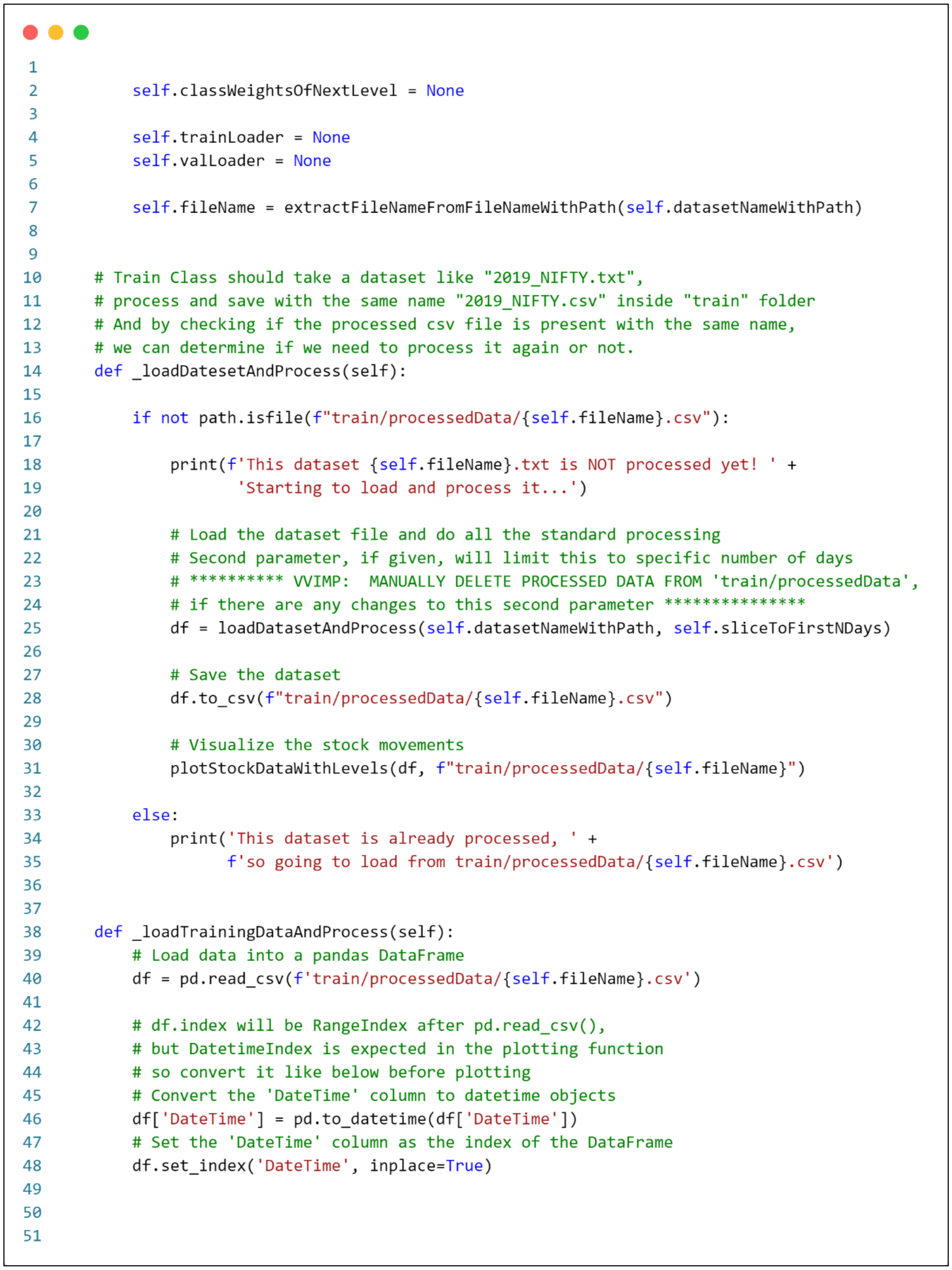
APPENDIX D

PYTHON SCRIPT: TRAIN MODEL



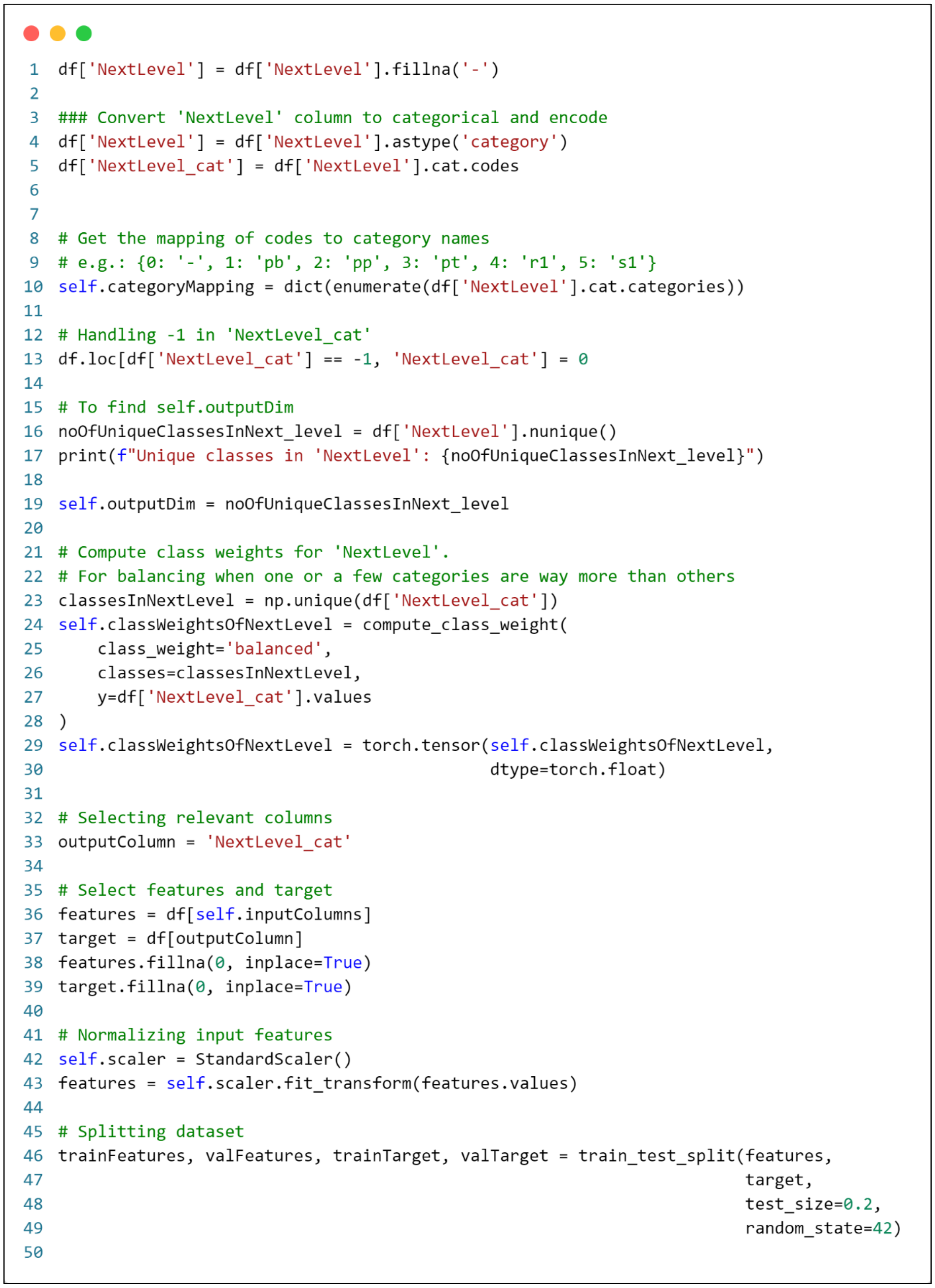
```python
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import pickle

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from torch.utils.data import DataLoader
from joblib import dump
from sklearn.utils.class_weight import compute_class_weight
from os import path

from common.Model import Model
from common.TrainDataset import TrainDataset
from common.commonFunctions import loadDatasetAndProcess, plotStockDataWithLevels
from common.commonFunctions import extractFileNameFromFileNameWithPath

class Train():

    def __init__(self, datasetNameWithPath, hiddenDim, layerDim, numEpochs,
                 inputColumns, sliceToFirstNDays=0):
        self.datasetNameWithPath = datasetNameWithPath
        self.hiddenDim = hiddenDim
        self.layerDim = layerDim
        self.numEpochs = numEpochs
        self.inputColumns = inputColumns
        self.sliceToFirstNDays = sliceToFirstNDays

        self.outputDim = None

        # model related
        self.categoryMapping = None
        self.scaler = None
        self.model = None
```

Figure Appendix D (a): Train.py – Part 1 of 4



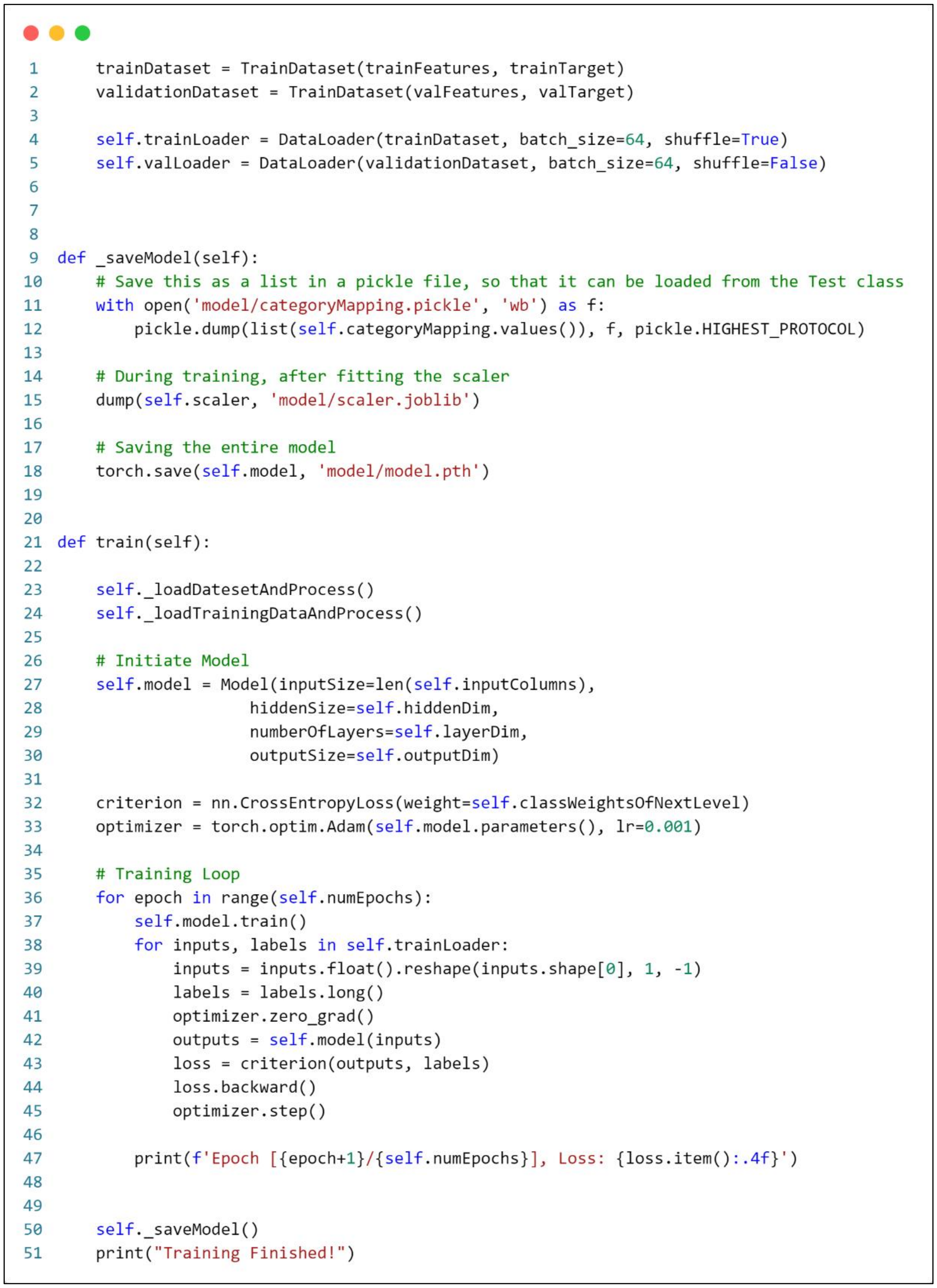
```python

        self.classWeightsOfNextLevel = None

        self.trainLoader = None
        self.valLoader = None

        self.fileName = extractFileNameFromFileNameWithPath(self.datasetNameWithPath)


    # Train Class should take a dataset like "2019_NIFTY.txt",
    # process and save with the same name "2019_NIFTY.csv" inside "train" folder
    # And by checking if the processed csv file is present with the same name,
    # we can determine if we need to process it again or not.
    def _loadDatesetAndProcess(self):

        if not path.isfile(f"train/processedData/{self.fileName}.csv"):

            print(f'This dataset {self.fileName}.txt is NOT processed yet! ' +
                   'Starting to load and process it...')

            # Load the dataset file and do all the standard processing
            # Second parameter, if given, will limit this to specific number of days
            # ********** VVIMP:  MANUALLY DELETE PROCESSED DATA FROM 'train/processedData',
            # if there are any changes to this second parameter ***************
            df = loadDatasetAndProcess(self.datasetNameWithPath, self.sliceToFirstNDays)

            # Save the dataset
            df.to_csv(f"train/processedData/{self.fileName}.csv")

            # Visualize the stock movements
            plotStockDataWithLevels(df, f"train/processedData/{self.fileName}")

        else:
            print('This dataset is already processed, ' +
                  f'so going to load from train/processedData/{self.fileName}.csv')


    def _loadTrainingDataAndProcess(self):
        # Load data into a pandas DataFrame
        df = pd.read_csv(f'train/processedData/{self.fileName}.csv')

        # df.index will be RangeIndex after pd.read_csv(),
        # but DatetimeIndex is expected in the plotting function
        # so convert it like below before plotting
        # Convert the 'DateTime' column to datetime objects
        df['DateTime'] = pd.to_datetime(df['DateTime'])
        # Set the 'DateTime' column as the index of the DataFrame
        df.set_index('DateTime', inplace=True)



```

Figure Appendix D (b): Train.py – Part 2 of 4



```python
df['NextLevel'] = df['NextLevel'].fillna('-')

### Convert 'NextLevel' column to categorical and encode
df['NextLevel'] = df['NextLevel'].astype('category')
df['NextLevel_cat'] = df['NextLevel'].cat.codes


# Get the mapping of codes to category names
# e.g.: {0: '-', 1: 'pb', 2: 'pp', 3: 'pt', 4: 'r1', 5: 's1'}
self.categoryMapping = dict(enumerate(df['NextLevel'].cat.categories))

# Handling -1 in 'NextLevel_cat'
df.loc[df['NextLevel_cat'] == -1, 'NextLevel_cat'] = 0

# To find self.outputDim
noOfUniqueClassesInNext_level = df['NextLevel'].nunique()
print(f"Unique classes in 'NextLevel': {noOfUniqueClassesInNext_level}")

self.outputDim = noOfUniqueClassesInNext_level

# Compute class weights for 'NextLevel'.
# For balancing when one or a few categories are way more than others
classesInNextLevel = np.unique(df['NextLevel_cat'])
self.classWeightsOfNextLevel = compute_class_weight(
    class_weight='balanced',
    classes=classesInNextLevel,
    y=df['NextLevel_cat'].values
)
self.classWeightsOfNextLevel = torch.tensor(self.classWeightsOfNextLevel,
                                            dtype=torch.float)

# Selecting relevant columns
outputColumn = 'NextLevel_cat'

# Select features and target
features = df[self.inputColumns]
target = df[outputColumn]
features.fillna(0, inplace=True)
target.fillna(0, inplace=True)

# Normalizing input features
self.scaler = StandardScaler()
features = self.scaler.fit_transform(features.values)

# Splitting dataset
trainFeatures, valFeatures, trainTarget, valTarget = train_test_split(features,
                                                                      target,
                                                                      test_size=0.2,
                                                                      random_state=42)

```

Figure Appendix D (c): Train.py – Part 3 of 4

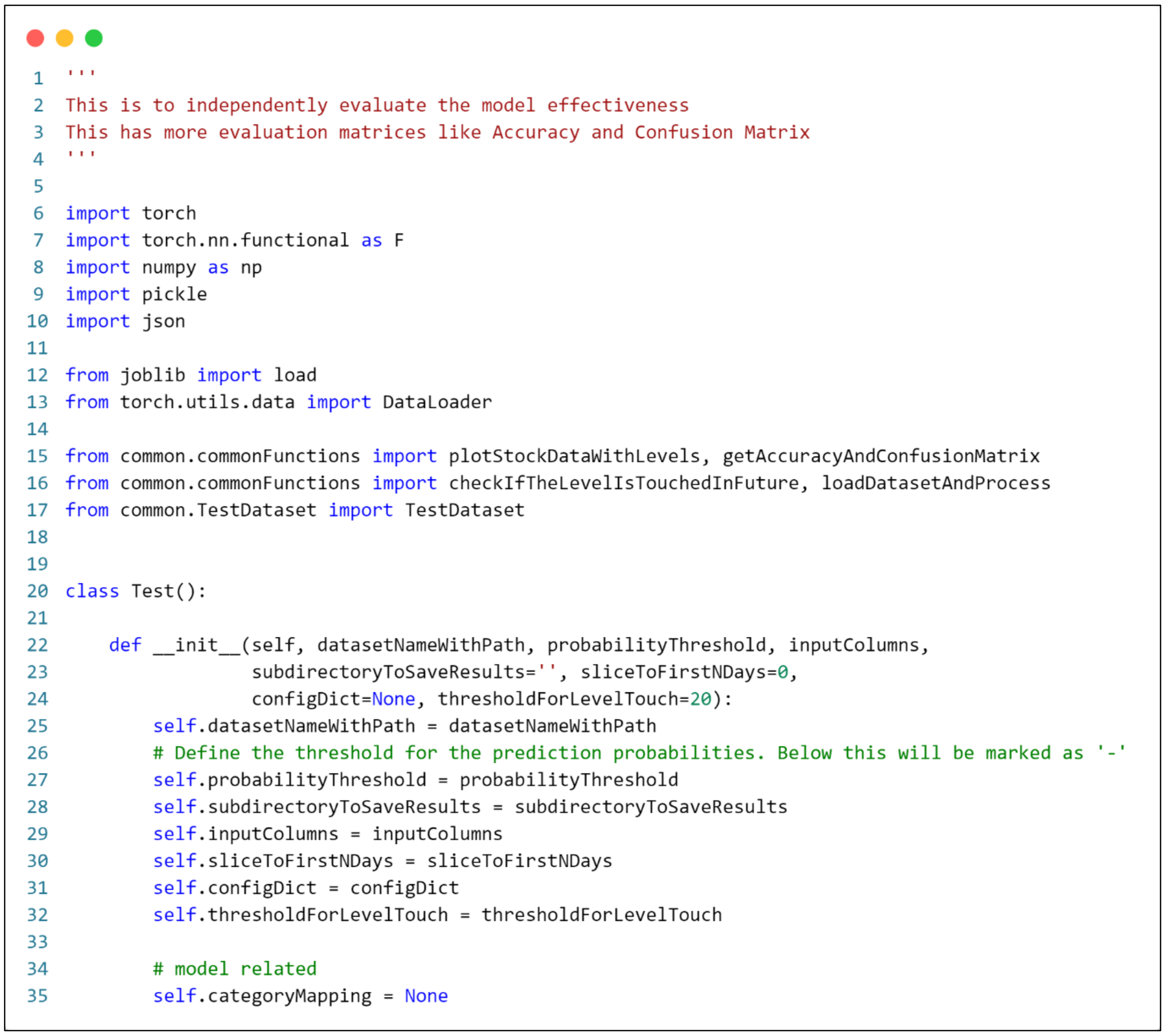


```python
    trainDataset = TrainDataset(trainFeatures, trainTarget)
    validationDataset = TrainDataset(valFeatures, valTarget)

    self.trainLoader = DataLoader(trainDataset, batch_size=64, shuffle=True)
    self.valLoader = DataLoader(validationDataset, batch_size=64, shuffle=False)



def _saveModel(self):
    # Save this as a list in a pickle file, so that it can be loaded from the Test class
    with open('model/categoryMapping.pickle', 'wb') as f:
        pickle.dump(list(self.categoryMapping.values()), f, pickle.HIGHEST_PROTOCOL)

    # During training, after fitting the scaler
    dump(self.scaler, 'model/scaler.joblib')

    # Saving the entire model
    torch.save(self.model, 'model/model.pth')


def train(self):

    self._loadDatesetAndProcess()
    self._loadTrainingDataAndProcess()

    # Initiate Model
    self.model = Model(inputSize=len(self.inputColumns),
                    hiddenSize=self.hiddenDim,
                    numberOfLayers=self.layerDim,
                    outputSize=self.outputDim)

    criterion = nn.CrossEntropyLoss(weight=self.classWeightsOfNextLevel)
    optimizer = torch.optim.Adam(self.model.parameters(), lr=0.001)

    # Training Loop
    for epoch in range(self.numEpochs):
        self.model.train()
        for inputs, labels in self.trainLoader:
            inputs = inputs.float().reshape(inputs.shape[0], 1, -1)
            labels = labels.long()
            optimizer.zero_grad()
            outputs = self.model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

        print(f'Epoch [{epoch+1}/{self.numEpochs}], Loss: {loss.item():.4f}')


    self._saveModel()
    print("Training Finished!")
```

Figure Appendix D (d): Train.py – Part 4 of 4

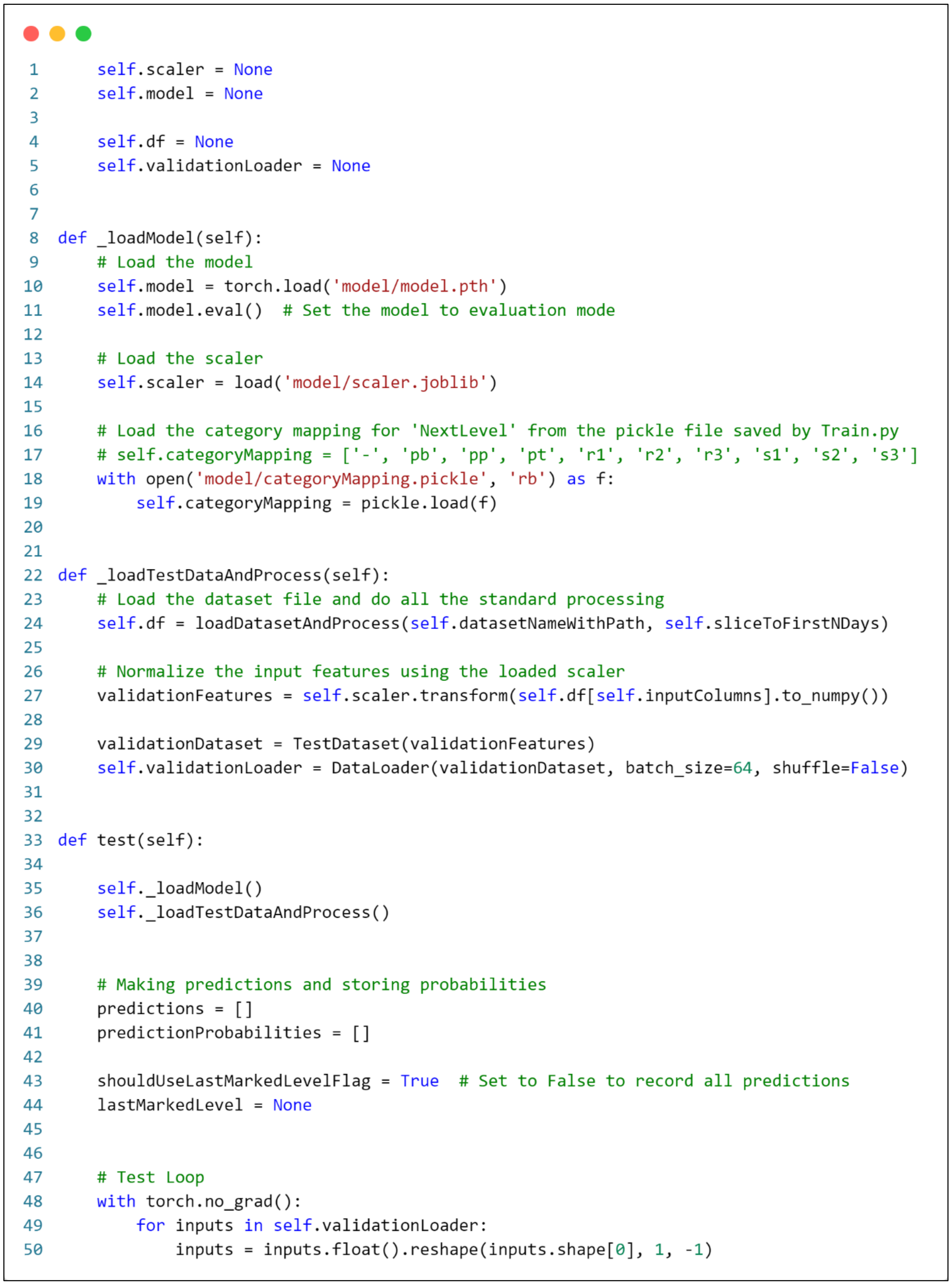
APPENDIX E

PYTHON SCRIPT: TEST MODEL



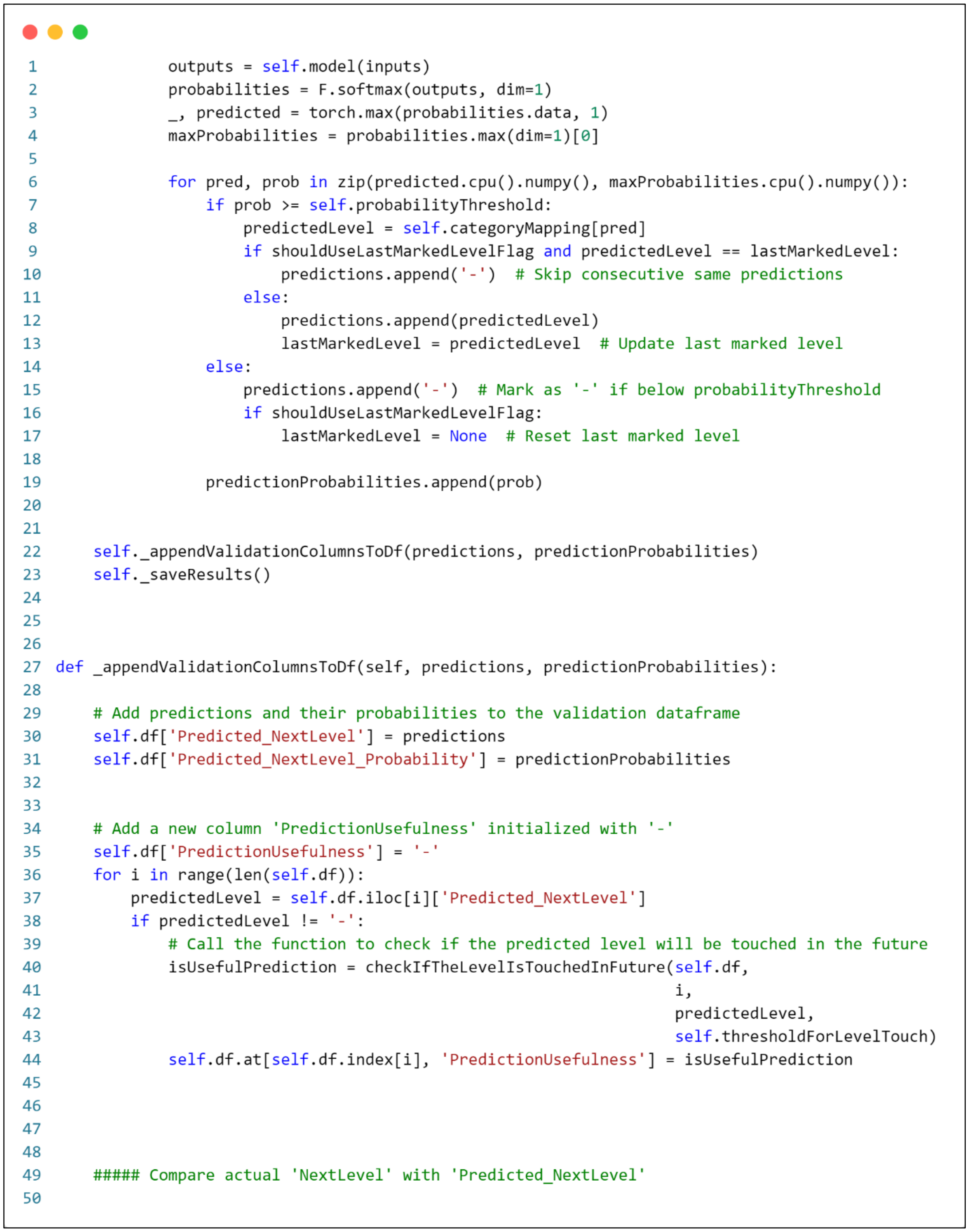
```python
'''
This is to independently evaluate the model effectiveness
This has more evaluation matrices like Accuracy and Confusion Matrix
'''

import torch
import torch.nn.functional as F
import numpy as np
import pickle
import json

from joblib import load
from torch.utils.data import DataLoader

from common.commonFunctions import plotStockDataWithLevels, getAccuracyAndConfusionMatrix
from common.commonFunctions import checkIfTheLevelIsTouchedInFuture, loadDatasetAndProcess
from common.TestDataset import TestDataset


class Test():

    def __init__(self, datasetNameWithPath, probabilityThreshold, inputColumns,
                 subdirectoryToSaveResults='', sliceToFirstNDays=0,
                 configDict=None, thresholdForLevelTouch=20):
        self.datasetNameWithPath = datasetNameWithPath
        # Define the threshold for the prediction probabilities. Below this will be marked as '-'
        self.probabilityThreshold = probabilityThreshold
        self.subdirectoryToSaveResults = subdirectoryToSaveResults
        self.inputColumns = inputColumns
        self.sliceToFirstNDays = sliceToFirstNDays
        self.configDict = configDict
        self.thresholdForLevelTouch = thresholdForLevelTouch

        # model related
        self.categoryMapping = None
```

Figure Appendix E (a): Test.py – Part 1 of 5



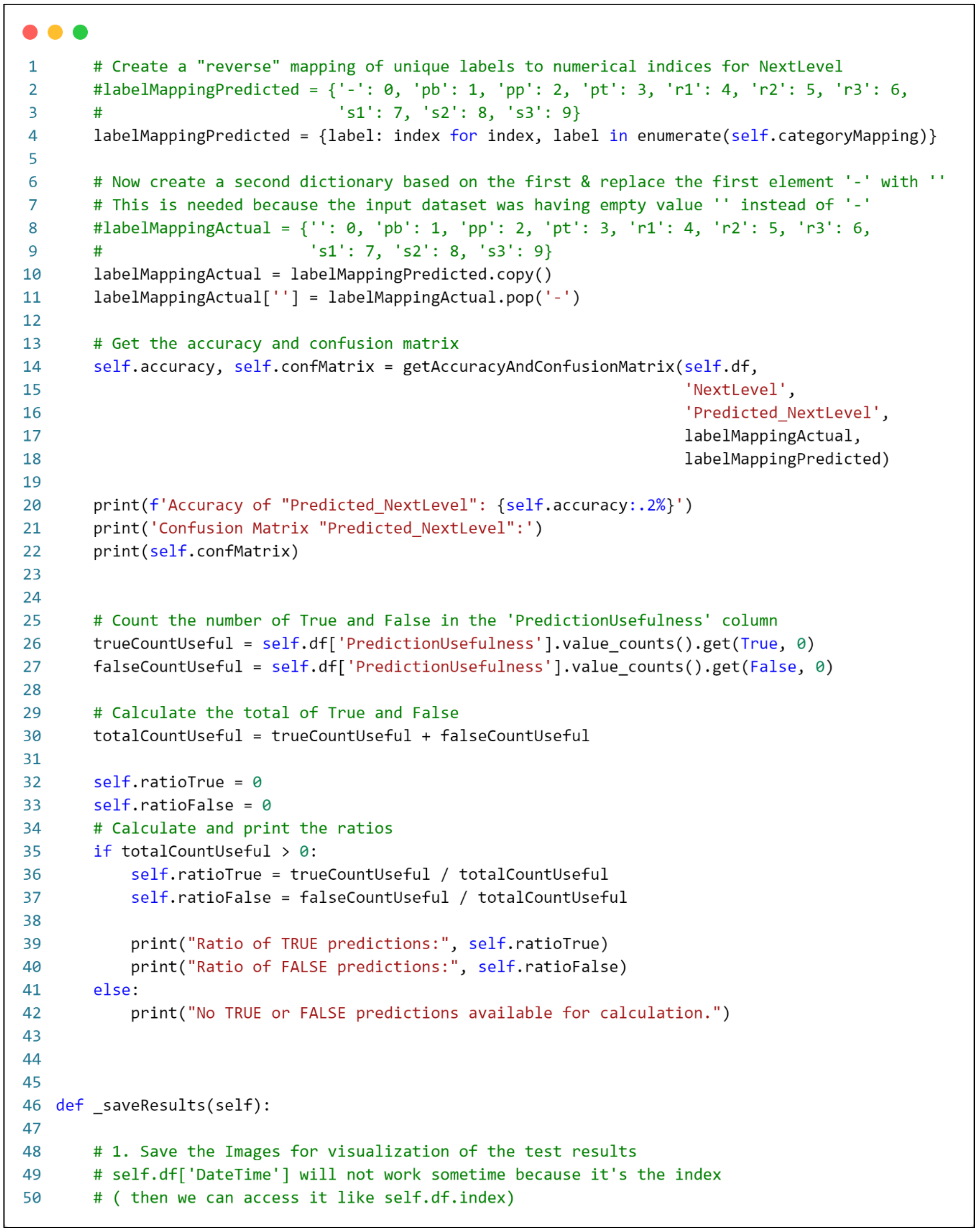
```python
    self.scaler = None
    self.model = None

    self.df = None
    self.validationLoader = None


def _loadModel(self):
    # Load the model
    self.model = torch.load('model/model.pth')
    self.model.eval()  # Set the model to evaluation mode

    # Load the scaler
    self.scaler = load('model/scaler.joblib')

    # Load the category mapping for 'NextLevel' from the pickle file saved by Train.py
    # self.categoryMapping = ['-', 'pb', 'pp', 'pt', 'r1', 'r2', 'r3', 's1', 's2', 's3']
    with open('model/categoryMapping.pickle', 'rb') as f:
        self.categoryMapping = pickle.load(f)


def _loadTestDataAndProcess(self):
    # Load the dataset file and do all the standard processing
    self.df = loadDatasetAndProcess(self.datasetNameWithPath, self.sliceToFirstNDays)

    # Normalize the input features using the loaded scaler
    validationFeatures = self.scaler.transform(self.df[self.inputColumns].to_numpy())

    validationDataset = TestDataset(validationFeatures)
    self.validationLoader = DataLoader(validationDataset, batch_size=64, shuffle=False)


def test(self):

    self._loadModel()
    self._loadTestDataAndProcess()


    # Making predictions and storing probabilities
    predictions = []
    predictionProbabilities = []

    shouldUseLastMarkedLevelFlag = True  # Set to False to record all predictions
    lastMarkedLevel = None


    # Test Loop
    with torch.no_grad():
        for inputs in self.validationLoader:
            inputs = inputs.float().reshape(inputs.shape[0], 1, -1)
```

Figure Appendix E (b): Test.py – Part 2 of 5



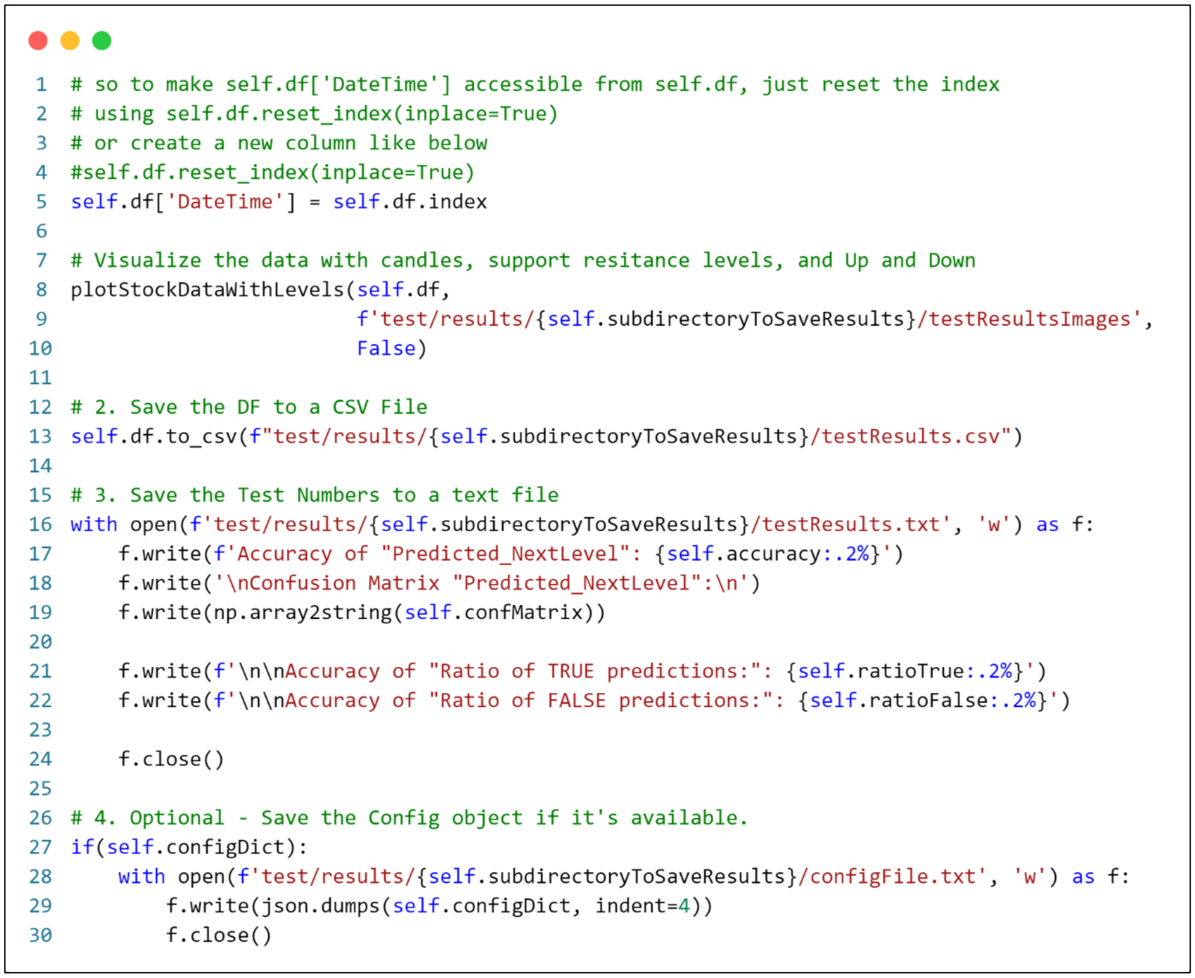
```
            outputs = self.model(inputs)
            probabilities = F.softmax(outputs, dim=1)
            _, predicted = torch.max(probabilities.data, 1)
            maxProbabilities = probabilities.max(dim=1)[0]

            for pred, prob in zip(predicted.cpu().numpy(), maxProbabilities.cpu().numpy()):
                if prob >= self.probabilityThreshold:
                    predictedLevel = self.categoryMapping[pred]
                    if shouldUseLastMarkedLevelFlag and predictedLevel == lastMarkedLevel:
                        predictions.append('-')  # Skip consecutive same predictions
                    else:
                        predictions.append(predictedLevel)
                        lastMarkedLevel = predictedLevel  # Update last marked level
                else:
                    predictions.append('-')  # Mark as '-' if below probabilityThreshold
                    if shouldUseLastMarkedLevelFlag:
                        lastMarkedLevel = None  # Reset last marked level

                predictionProbabilities.append(prob)


    self._appendValidationColumnsToDf(predictions, predictionProbabilities)
    self._saveResults()



def _appendValidationColumnsToDf(self, predictions, predictionProbabilities):

    # Add predictions and their probabilities to the validation dataframe
    self.df['Predicted_NextLevel'] = predictions
    self.df['Predicted_NextLevel_Probability'] = predictionProbabilities


    # Add a new column 'PredictionUsefulness' initialized with '-'
    self.df['PredictionUsefulness'] = '-'
    for i in range(len(self.df)):
        predictedLevel = self.df.iloc[i]['Predicted_NextLevel']
        if predictedLevel != '-':
            # Call the function to check if the predicted level will be touched in the future
            isUsefulPrediction = checkIfTheLevelIsTouchedInFuture(self.df,
                                                                  i,
                                                                  predictedLevel,
                                                                  self.thresholdForLevelTouch)
            self.df.at[self.df.index[i], 'PredictionUsefulness'] = isUsefulPrediction




    ##### Compare actual 'NextLevel' with 'Predicted_NextLevel'

```

Figure Appendix E (c): Test.py – Part 3 of 5



```
    # Create a "reverse" mapping of unique labels to numerical indices for NextLevel
    #labelMappingPredicted = {'-': 0, 'pb': 1, 'pp': 2, 'pt': 3, 'r1': 4, 'r2': 5, 'r3': 6,
    #                         's1': 7, 's2': 8, 's3': 9}
    labelMappingPredicted = {label: index for index, label in enumerate(self.categoryMapping)}

    # Now create a second dictionary based on the first & replace the first element '-' with ''
    # This is needed because the input dataset was having empty value '' instead of '-'
    #labelMappingActual = {'': 0, 'pb': 1, 'pp': 2, 'pt': 3, 'r1': 4, 'r2': 5, 'r3': 6,
    #                      's1': 7, 's2': 8, 's3': 9}
    labelMappingActual = labelMappingPredicted.copy()
    labelMappingActual[''] = labelMappingActual.pop('-')

    # Get the accuracy and confusion matrix
    self.accuracy, self.confMatrix = getAccuracyAndConfusionMatrix(self.df,
                                                                   'NextLevel',
                                                                   'Predicted_NextLevel',
                                                                   labelMappingActual,
                                                                   labelMappingPredicted)

    print(f'Accuracy of "Predicted_NextLevel": {self.accuracy:.2%}')
    print('Confusion Matrix "Predicted_NextLevel":')
    print(self.confMatrix)


    # Count the number of True and False in the 'PredictionUsefulness' column
    trueCountUseful = self.df['PredictionUsefulness'].value_counts().get(True, 0)
    falseCountUseful = self.df['PredictionUsefulness'].value_counts().get(False, 0)

    # Calculate the total of True and False
    totalCountUseful = trueCountUseful + falseCountUseful

    self.ratioTrue = 0
    self.ratioFalse = 0
    # Calculate and print the ratios
    if totalCountUseful > 0:
        self.ratioTrue = trueCountUseful / totalCountUseful
        self.ratioFalse = falseCountUseful / totalCountUseful

        print("Ratio of TRUE predictions:", self.ratioTrue)
        print("Ratio of FALSE predictions:", self.ratioFalse)
    else:
        print("No TRUE or FALSE predictions available for calculation.")



def _saveResults(self):

    # 1. Save the Images for visualization of the test results
    # self.df['DateTime'] will not work sometime because it's the index
    # ( then we can access it like self.df.index)
```

Figure Appendix E (d): Test.py – Part 4 of 5



```python
# so to make self.df['DateTime'] accessible from self.df, just reset the index
# using self.df.reset_index(inplace=True)
# or create a new column like below
#self.df.reset_index(inplace=True)
self.df['DateTime'] = self.df.index

# Visualize the data with candles, support resitance levels, and Up and Down
plotStockDataWithLevels(self.df,
                        f'test/results/{self.subdirectoryToSaveResults}/testResultsImages',
                        False)

# 2. Save the DF to a CSV File
self.df.to_csv(f"test/results/{self.subdirectoryToSaveResults}/testResults.csv")

# 3. Save the Test Numbers to a text file
with open(f'test/results/{self.subdirectoryToSaveResults}/testResults.txt', 'w') as f:
    f.write(f'Accuracy of "Predicted_NextLevel": {self.accuracy:.2%}')
    f.write('\nConfusion Matrix "Predicted_NextLevel":\n')
    f.write(np.array2string(self.confMatrix))

    f.write(f'\n\nAccuracy of "Ratio of TRUE predictions:": {self.ratioTrue:.2%}')
    f.write(f'\n\nAccuracy of "Ratio of FALSE predictions:": {self.ratioFalse:.2%}')

    f.close()

# 4. Optional - Save the Config object if it's available.
if(self.configDict):
    with open(f'test/results/{self.subdirectoryToSaveResults}/configFile.txt', 'w') as f:
        f.write(json.dumps(self.configDict, indent=4))
        f.close()
```

Figure Appendix E (e): Test.py – Part 5 of 5

REFERENCES

Ali, M., Khan, D.M., Aamir, M., Ali, A. & Ahmad, Z. (2021) Predicting the Direction Movement of Financial Time Series Using Artificial Neural Network and Support Vector Machine. *Complexity*. 2021. doi:10.1155/2021/2906463.

Anon (n.d.a) *3.3. Metrics and scoring: quantifying the quality of predictions — scikit-learn 1.4.0 documentation*. https://scikit-learn.org/stable/modules/model_evaluation.html [Accessed: 7 February 2024].

Anon (n.d.b) *Average Directional Index (ADX) [ChartSchool]*. https://school.stockcharts.com/doku.php?id=technical_indicators:average_directional_index_adx [Accessed: 13 February 2024].

Anon (n.d.c) *Average True Range (ATR) [ChartSchool]*. https://school.stockcharts.com/doku.php?id=technical_indicators:average_true_range_atr [Accessed: 13 February 2024].

Anon (n.d.d) *Exponential Moving Average (EMA) - Overview, How To Calculate*. https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/exponential-moving-average-ema/ [Accessed: 6 February 2024].

Anon (n.d.e) *How are the pivot points derived while using the Central Pivot Range (CPR)?* https://support.zerodha.com/category/trading-and-markets/kite-web-and-mobile/charts/articles/how-are-pivot-points-derived-while-using-the-cpr [Accessed: 2 February 2024].

Anon (n.d.f) *How are the pivot points derived while using the Central Pivot Range (CPR)?* https://support.zerodha.com/category/trading-and-markets/kite-web-and-mobile/charts/articles/how-are-pivot-points-derived-while-using-the-cpr [Accessed: 14 February 2024].

Anon (n.d.g) *Indicator function - Wikipedia*. https://en.wikipedia.org/wiki/Indicator_function [Accessed: 7 February 2024].

Anon (n.d.h) *Law of large numbers - Wikipedia*. https://en.wikipedia.org/wiki/Law_of_large_numbers [Accessed: 11 February 2024].

Anon (n.d.i) *Linear — PyTorch 2.2 documentation*. https://pytorch.org/docs/stable/generated/torch.nn.Linear.html [Accessed: 7 February 2024].

Anon (n.d.j) *Momentum; Rate Of Change (ROC); Relative Strength Index (RSI); Stochastic Oscillator (%K); Fast %D; Slow %D; Williams %R*. https://thismatter.com/money/technical-analysis/momentum.htm [Accessed: 6 February 2024].

Anon (n.d.k) *sklearn.metrics.accuracy_score — scikit-learn 1.4.0 documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html [Accessed: 7 February 2024].

Anon (n.d.l) *sklearn.preprocessing.StandardScaler — scikit-learn 1.4.0 documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [Accessed: 6 February 2024].

Anon (n.d.m) *TA-Lib*. https://ta-lib.github.io/ta-lib-python/ [Accessed: 6 February 2024].

Anon (n.d.n) *Trading strategy - Wikipedia*. https://en.wikipedia.org/wiki/Trading_strategy [Accessed: 12 February 2024].

Anon (n.d.o) *Using Pivot Points in Forex Trading*. https://www.investopedia.com/articles/forex/07/pivotpointstrategy.asp [Accessed: 2 February 2024].

Barber, B.M., Lee, Y.-T., Liu, Y.-J. & Odean, T. (2005) *Do Day Traders Make Money?* www.gsm.ucdavis.edu/~bmbarber.

Burns, S. & Burns, H. (2021) *Trading Habits: 39 of the World's Most Powerful Stock Market Rules*.

Chague, F., De-Losso, R. & Giovannetti, B. (2019) *Day trading for a living? CEQEF-Nº57 Working Paper Series*. www.eesp.fgv.br.

Dwivedi, R., Datta Gupta, K., Sharma, T., Raizada, R., Yadav, S. & Bhatia, V. (2022) ANALYSING TRADING STRATEGIES AND FORECASTING STOCK PRICES USING LSTM. *Journal of Theoretical and Applied Information Technology*. 15 (15). www.jatit.org.

Fischer, T.; & Krauss, C. (2017) *Deep learning with long short-term memory networks for financial market predictions*. http://hdl.handle.net/10419/157808.

Foot, P. (n.d.) *Japanese Candlesticks: What They Are + How to Trade in the UK | IG International*. https://www.ig.com/en/trading-strategies/japanese-candlestick-trading-guide-200615 [Accessed: 2 February 2024].

Frattini, A., Bianchini, I., Garzonio, A. & Mercuri, L. (2022) Financial Technical Indicator and Algorithmic Trading Strategy Based on Machine Learning and Alternative Data. *Risks*. 10 (12). doi:10.3390/risks10120225.

Frykmer, D. & Johnsson, O. (2019) *Pivot Point Trading in the Foreign Exchange Market A Test for Market Efficiency*.

Jordan, D.J. & Diltz, J.D. (2003) The Profitability of Day Traders. *Financial Analysts Journal*. 59 (6), 85–94. doi:10.2469/faj.v59.n6.2578.

Katz, J. & McCormick, D. (2000) *The Encyclopedia of Trading Strategies*. McGraw Hill Professional.

Kumbhare, P., Kolhe, L., Dani, S., Fandade, P. & Theng, D. (2023) Algorithmic Trading Strategy Using Technical Indicators. *International Conference on Emerging Trends in Engineering and Technology, ICETET*. 2023-April. doi:10.1109/ICETET-SIP58143.2023.10151614.

Loukas, S. (2020) *How and why to Standardize your data: A python tutorial | Towards Data Science*. 2020. https://towardsdatascience.com/how-and-why-to-standardize-your-data-996926c2c832 [Accessed: 6 February 2024].

Mittal, S. & Chauhan, A. (2021) A RNN-LSTM-Based Predictive Modelling Framework for Stock Market Prediction Using Technical Indicators. *International Journal of Rough Sets and Data Analysis*. 7 (1), 1–13. doi:10.4018/ijrsda.288521.

Nirob, F.A. & Hasan, M.M. (2023) Predicting Stock Price from Historical Data using LSTM Technique. *Journal of Artificial Intelligence and Data Science (JAIDA)*. 3 (1), 36–49. https://dergipark.org.tr/pub/jaida.

Person, J.L. (2004) *A Complete Guide to Technical Trading Tactics*.

rmsharma@gmail.com (n.d.) *oneminutedata - Google Drive*. https://drive.google.com/drive/folders/0B8e3dtbFwQWUZ1I5dklCMmE5M2M?resourcekey=0-__hqGabgp_MZvBFGo140Xg [Accessed: 2 February 2024].

Tian, X., Quan, C., Zhang, J. & Cai, H.J. (2012) Optimization of Intraday Trading Strategy Based on ACD Rules and Pivot Point System in Chinese Market. *Journal of Intelligent Learning Systems and Applications*. 04 (04), 279–284. doi:10.4236/jilsa.2012.44029.

Yadav, A., Jha, C.K. & Sharan, A. (2020) Optimizing LSTM for time series prediction in Indian stock market. In: *Procedia Computer Science*. 2020 Elsevier B.V. pp. 2091–2100. doi:10.1016/j.procs.2020.03.257.

Yan, S. (2016) *Understanding LSTM and its diagrams | by Shi Yan | ML Review*. 2016. https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714 [Accessed: 4 February 2024].

Zaheer, S., Anjum, N., Hussain, S., Algarni, A.D., Iqbal, J., Bourouis, S. & Ullah, S.S. (2023) A Multi Parameter Forecasting for Stock Time Series Data Using LSTM and Deep Learning Model. *Mathematics*. 11 (3). doi:10.3390/math11030590.