“A Comparative Study on Different LSTM architectures to predict the stock price ”.

CHARLIE THOMAS

STUDENT ID: 1080424

MSC IN MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

LIVERPOOL JOHN MOORES UNIVERSITY

Final Thesis Report

August 2023

DEDICATIONS

I dedicate my research and thesis work to my family and many friends. A special feeling of gratitude to my loving parents, Mr. Thomas and Mrs. Annamma Thomas whose words of encouragement and push for tenacity ring in my ears. My Wife, Archana Rajkumar and My Son, Matthias Thomas have never left my side and are very special. Their playful character has kept me relaxed and focused. Both of you have been my best cheerleaders.

I also dedicate this work to my friends, Pankaj Bhardwaj, Deepak Kumar, Rohan Tandon, Siddharth Dubey and Fahim Ahmed and I will always appreciate all they have done.

ACKNOWLEDGMENTS

This work would not have been possible without the help, thoughtful guidance, and incredible support of my Thesis Supervisor, Dr. Ganesh S.; A massive Thanks to provided me with the tools and inspiration that I needed to carry out this research. I would like to my Thank Program Co-ordinator, Ms. Karen Dsouza. Her support was very crucial in successful completion of my thesis. Thank you for advising me and for always finding the time to help, even from different time zones. I would also like to recognize the professors from Liverpool John Moores University and IIIT-Bangalore whose interactions have helped me to gain immense knowledge to work on my thesis. I would also like to Thank Liverpool John Moores University's machine learning department for enabling me to pursue this research during the spring and summer of 2023.

ABSTRACT

Stock price prediction has welcomed the development of advanced capabilities due to the speedy development of machine learning techniques and Artificial Intelligence, Improved computational speed, and ever-increasing data Availability. The stock market has become more volatile and complex due to access to investment opportunities is easier in current times. This research compares three different types of LSTM architecture(single-layer, multi-layer, and bi-directional) to predict the stock price of Indian banking stock, Yes Bank Ltd. The Structure of the Thesis is as follows: Chapter is 1, an introduction to the Thesis carried out; Chapter 2 , highlights the review of the Literature in the studied area; Chapter 3, explores the Research Methodology; Section 4, describes the Analysis of the study; Section 5, Results of the analysis are discussed; conclusions and future work are presented in Chapter 6, followed by references, and appendices.

The study uses predictors such as Historical prices(open, close, high, and low). Historical price information has been extracted from the Yahoo finance data by using the yfinance library for the period from 2005-05-07 to 2023-08-09. The bi-variate analysis highlights the high correlation between historical price features. Hence Date and Close price were utilized for further analysis. Data was transformed using a combination of Log-square root transformers to make the data stationary. A new Data set with features Date, input_1, Input_2, Input_3 and Target was created by the feature engineering process. Three baseline models and 45 experimental models with different combinations of batch size and number of epochs were built and compared on the performance based on MSE(Loss on validation data). A bidirectional LSTM with 10 epochs and 64 batch size was selected for hyperparameter tuning because of the low score of MSE(validation data). The hyperparameter tuning will be performed to get the most optimized model to predict the stock price. Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). A tuned Bi-directional model was proposed to be implemented on unseen/ test data. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1)(refer to table 5.3.1). The Proposed model was also able to predict the trend closely to the real trend(refer to plot 5.3.1). This paper established that simple LSTM architecture such as single-layer LSTM was less efficient than complex LSTM architecture such as multi-layer LSTM and bi-directional LSTM. This study also established that bi-directional LSTM was able to predict the trend of the moment of the stock efficiently(refer to plot 5.3.1).

LIST OF TABLES

LIST OF FIGURES

LIST OF Illustrations

LIST OF PLOTS

LIST OF ABBREVIATIONS

Abbreviations	Full-Form
ARIMA	AUTO-REGRESSIVE INTEGRATED MOVING AVERAGE
RNN	RECURRENT NEURAL NETWORKS
LSTM	LONG SHORT TERM MEMORY
RMSE	ROOT MEAN SQUARE ERROR
MSE	MEAN SQUARE ERROR
BSE	BOMBAY STOCK EXCHANGE
LTD.	LIMITED
CSI300	CHINA SECURITIES INDEX
CNN-LSTM	CONVOLUTIONAL NEURAL NETWORK- LONG SHORT TERM MEMORY
S&P 500	STANDARD & POOR 500
NIFTY 50	NATIONAL STOCK EXCHANGE 50
DJIA	DOW JONES INDUSTRIAL AVERAGE
SVM	SUPPORT VECTOR MACHINE
NIKKIE 225	NIHON KEIZAI 225
MAE	MEAN ABSOLUTE ERROR
LSTM-DNN	LONG SHORT TERM MEMORY- DEEP NEURAL NETWORK
ARMP	AUTO-REGRESSIVE MOVING POINTER
GRU	GATED RECURRENT UNITS
LTP	LAST TRADED PRICE
MAPE	MEAN ABSOLUTE PERCENTAGE ERROR
S_I_LSTM	SENTIMENT INDICATOR LONG SHORT TERM MEMORY
R/R-VALUE	CORRELATION COEFFICIENT
MI-LSTM	MULTIPLE INTERACTIONS LONG SHORT TERM MEMORY MODEL
BI-LSTM	BI-DIRECTIONAL LONG SHORT TERM MEMORY
LSTM-DRNN	LONG SHORT TERM MEMORY- DEEP RECURRENT NEURAL NETWORK
GC-LSTM	GRAPH CONVOLUTIONAL- LONG SHORT TERM LSTM
CNN-TLSTM	CONVOLUTIONAL NEURAL NETWORK- LONG SHORT TERM MEMORY
CNN-BI LSTM	CONVOLUTIONAL NEURAL NETWORK- BIDIRECTIONAL LONG SHORT TERM MEMORY
ADAM	ADAPTIVE MOMENT ESTIMATION
NN	NEURAL NETWORK
RELU	RECTIFIED LINEAR ACTIVATION FUNCTION

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF THE STUDY

The reasons behind uncertain stock price fluctuations are complex and interconnected. The cause could be global economic data, unemployment rate changes, monetary and fiscal policies, policies related to immigration, natural calamities, health indicators of the state, etc(Biswas et al.,2023). Stakeholders in the share market focus on maximizing profits and minimizing adversity through in-depth stock market evaluations. The main challenge is consolidating multifaceted information and building a robust, dependable, and well-grounded model for well-aimed and precise predictions(Perry,2023). It is difficult to predict stock prices accurately due to the inherent complexity and noise of stock markets. Predicting future prices is a difficult task that requires both the effectiveness and efficiency of the process. Many approaches have been suggested for selecting features from financial data in stock prediction, a difficult task. Some Fact finders solely utilize fundamental indicators and some utilize data derived through technical analysis while others use historical data, leading to suboptimal predictive model performance due to limited features. If all available financial features are considered, then the respective model may become intricate, and hard to explain and explicate(Usmani and Shamsi,2023). Collinearity among variables may lead to worse model performance. A model with optimal attributes can predict stock prices and inform markets. Research has studied variables that correlate with stock behaviour. Achieving the expected outcome may be hindered by inaccurate and weak models resulting from poor variable selection. A well-thought-out combination of features can significantly improve the performance of the developed model. This study's contribution lies in the meticulous selection of variables, creating new features to improve the price predictions.

There is a rich and diverse history of analysis(quantitative) in the field of finance, with numerous models developed to address financial issues. However, it's important to note that not all research or models based on calculable analyses of quantifiable data are universally undertaken or broadly implemented. In the 1970s, esteemed British statisticians Box and Jenkins made an initial attempt at utilizing mainframe computers for this purpose(Yang, Wang and Li,2022). An Auto-Regressive Integrated Moving Average model, popularly known as the ARIMA model was developed to analyse the historical data of price and volume. This powerful tool has the ability to make predictions based on past trends. It is crucial to note that ARIMA

is solely designed to manage and predict time series data that are stationary by nature. Its use for non-stationary data can result in abysmal performance. Hence, it is evident and practical to manage the time series data with non-stationary characteristics before utilization, despite the potential loss of the actual and authentic structure and expandability of the feature. Classical models assume a linear relationship in data, with very few exceptions. However, this presupposition uprears important aspects of the reliability and dependability of classical time series models, given that these data are often non-linear in real-world scenarios(Ma et al.,2021).

During the 1980s, advancements in data analysis tools and techniques led to more exciting developments(Yadav et al.,2020). The creation of the spreadsheet allowed for the modelling of financial data to analyse the asset quality and its performance, while the collection of the data through automated processes became a reality due to advancements in computational capability and machine performance. Predictive and forecasting models were able to speedily and methodically analyse data due to these advancements. With the availability of large-scale data, researchers began to explore the prowess of artificial intelligence by utilizing deep learning techniques for predictive analysis and modelling of sequential data. These architectures excel at learning intricate non-linear and complex relationships among various features in time series data that conventional and long-established methods struggle with, and can efficiently extract the information from explanatory variables in an efficient and methodical manner. Numerous deep learning architectures have been created to tackle diverse issues and the inherent structure of datasets. In a basic feedforward neural network architecture, information only flows in the forward direction. As each input is processed independently, it does not retain any information from the previous step. Therefore, these models are ineffective in dealing with sequential data, where prior events are crucial in predicting future events.

Recurrent neural networks (RNN) excel in the realm of sequential data modelling and time series applications(Shahi et al.,2020). They are particularly adept at tasks such as predicting stock market trends, facilitating language translations, providing auto-completion in messages/emails, and processing signals. The RNN architecture's loops are specifically designed to ensure that relevant information persists over time, passing information from one timestep to the next internally within the network(Wu et al.,2021). In the process of training an RNN, the error is calculated between predicted and actual values from a labelled dataset. The goal is to minimize this error by updating the network's parameters, including weights and biases until the lowest possible value is achieved. This is done using a gradient, which determines the rate at which cost changes with respect to each parameter. By adjusting the

parameters iteratively, the gradient provides a direction to move in the error surface. This process is known as backpropagation, where the error is propagated backwards from the output layer to the input layer. However, one of the challenges of this technique is that parameters can be located anywhere within the network, making it difficult to find a gradient. This can result in the vanishing gradient problem, where gradients decay back through the network, especially in earlier layers. Unfortunately, this issue is also present in the RNN architecture. The LSTM architecture is a well-known recurrent neural network that effectively solves the vanishing gradient problem(Shah, Vaidya and Shah,2022). With its default behaviour of retaining information for an extended period, the LSTM model is ideal for time series data.

This study focuses on utilizing the LSTM model to predict stock index prices, which is an improved version of neural network architecture for time series analysis. The study utilized selected fundamental data features to build the model. First, the data is transformed with the log-square root transformation technique. Then, an intuitive feature engineering approach is applied to create new 4 features and new data set. Hyperparameters such as the number of Units, epochs, learning rate, batch size, number of layers and optimizers are skilfully incorporated into the model. By finely tuning the hyperparameters, the LSTM model expertly uses input data to accurately predict the stock market index's closing price. When evaluating the effectiveness of a stock market prediction model, it's important to consider metrics such as RMSE and MSE(Rather,2021). Many researchers have delved into this field, but some have focused solely on complex statistical or machine-learning techniques without taking into account all the variables that could potentially impact stock market prediction. Others have only used fundamental data without exploring other factors that could influence the market. The goal of the study is to create a model that strikes a balance between simplicity and comprehensiveness, incorporating a well-rounded set of variables to capture the behaviour of the stock market from multiple angles. The result is a model that doesn't add any complexity to the architecture but still maintains the ability to accurately predict market trends.

1.2 PROBLEM STATEMENT

Yes Bank was incorporated in the year 1999 by Ashok Kapur, Harkirat Singh and Rana Kapoor. Yes Bank’s Non-banking financial business has a shareholding pattern of 25 % of the business with former named incorporators and 75 % held by Rabo Bank from the Netherlands. Yes Bank Ltd. was listed on the BSE, Bombay Stock Exchange in May 2005 and also had a follow-up public offering in July 2020. The Bank faced a steep fall in stock price from mid-2019 because

of mismanagement of loan lending, governance issues, false Assurance and lack of insight by the management. The Yes Bank collapsed financially and the government and other banks restructured the Yes Bank and helped it revamp its operation in 2019-20. Yes Bank Ltd. has fallen from an all-time high to an all-time in the last 5 years. This led to an influx of retail and corporate investors hoarding the shares of the bank. The turnaround of the bank, traders and investors' anticipation about the stock price and the high volume of the stock's trade on the stock exchange motivated this study to develop an LSTM model that can predict the stock price of Yes Bank.

The research aims to compare the different forms of LSTM architectures such as Single-layer LSTM, bi-directional LSTM, and Multi-layer LSTM to predict the future price of Yes Bank Ltd. The study will utilize Yes Bank Ltd. Historical price data. Yes Bank has been one of the most traded stocks on the NSE and BSE, the stock exchange of India. Investors(retail and corporate) have shown keen interest in the stock for long-term as well as short-term investments. The research aims to propose a model that would be able to predict the future price as well as trend in the moment of the stock. The research proposes to develop an LSTM model that will be less complex and would require historical price data to predict the future price of the stock.

1.3 OBJECTIVES AND AIM

The aim of this study is to propose an effective LSTM model that can provide stock price prediction with low variability. This will be achieved by comparing the performance of various LSTM architectures such as Single-layer LSTM, bi-directional LSTM, and Multi-layer LSTM to predict the future price of Yes Bank Ltd. To accomplish this, the research will incorporate predictors such as the daily historical prices such as Open price, Close price, High price and Low price on the given day. The research proposes to develop an LSTM model that will be less complex and would require historical price data to predict the future price of the stock. The research objectives are as follows:

1. The examination of patterns among various predictors, which comprises historical fundamental indicators like historical prices (open, high, low, close).
2. A thorough comparison of the performance of various predictive models such as single-layer LSTM, bi-directional LSTM, and multi-layer LSTM.
3. To run the trial for hyperparameter tuning to improve the performance of the model.
4. To predict the trend of the stock.

1.4 RESEARCH QUESTIONS

1. Is the data stationary or non-stationary?

   The following are the test hypotheses:

   Null hypothesis (H0): The time series data is non-stationary.

   Alternate hypothesis (H1): The time series is stationary (or trend-stationary).

2. Did the Transformation of the data change the nature of the data to stationary?

   The following are the test hypotheses:

   Null hypothesis (H0): The applied transformation technique on time series data did not change the characteristics to stationary.

   Alternate hypothesis (H1): The Applied transformation technique on time series data changed the characteristics to stationary. (or trend-stationary).

   Research question 1 and 2 answered in section 4.4, Hypothesis testing

3. Was the proposed model able to predict the trend of the stock on unseen data?

   Research Question is Answered in Section 5.5 and Plot 5.5.1

1.5 STUDY'S SCOPE

The analysis of Yes Bank Ltd.'s stock examined its historical prices such as Open, Close, High, and low to propose an LSTM model that can predict the stock price. The study solely relied on Historical price information and information such as technical indicators, macroeconomic data, market sentiments etc. has not been incorporated. The study did not compare other machine learning algorithms and statistical models but three LSTM algorithms such as single-layer LSTM, multi-layer LSTM and Bi-directional LSTM.

1.6 IMPORTANCE OF THE STUDY

The accurate prediction of stock prices is a subject of immense interest to many stakeholders, who are increasingly turning to machine learning and deep learning algorithms to achieve this goal. Several academic studies have conclusively demonstrated that various LSTM models

consistently outperform other machine learning algorithms. This is because stock prices exhibit non-linear behaviour and are influenced by a wide range of economic and financial variables. Moreover, the presence of noise in the time series data and auto-correlation further adds to the difficulty of making accurate predictions. With this in mind, this research proposed a novel approach that involves utilizing an LSTM architecture to predict Yes Bank Ltd.'s future price based on an analysis of historical data (open, close, high, low).To achieve this, the research will compare single-layer LSTM, multi-layer LSTM, and bi-directional LSTM architectures, and evaluate their performance using different metrics such as MSE and RMSE to identify the optimal model. Importantly, this research seeks to make an invaluable contribution by proposing a less complex model that relies on the historical price of the stock.

1.7 STRUCTURE OF THE STUDY

The flow of this paper is organized as follows. Chapter 2 highlights the review of the Literature in the studied area. Chapter 3 explores the Research Methodology, Section 4 describes the Analysis of the study. Section 5 Results of the analysis are discussed. conclusions and future work are presented in Chapter 6, followed by acknowledgements, references, and appendices.

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION

In the realm of time series prediction, an LSTM model is a widely used architecture in the Recurrent Neural Network(RNN). Its application is not limited to stock market prediction as it is also used in rainfall-run-off modelling, MRI data analysis, anomaly detection, and mobile traffic prediction. Unlike traditional networks, standard RNNs are better at preserving information. However, it falls short when it comes to learning long-term dependencies due to the vanishing gradient problem. LSTM overcomes this issue by using memory cells. Its standard architecture contains an input layer, an output layer, at least one hidden layer and a cell state. The cell state is the most crucial component of the LSTM architecture as it runs through the chain, keeping the flow of data/information unchanged. The gate mechanism allows for the deletion or modification of cell state information.

2.2 LSTM IN STOCK PRICE PREDICTION

It is crucial to note that transaction data such as historical prices strongly correlates with the Time function, and generally, machine learning algorithms fall short when it comes to considering contextual information while processing time series data. Hochreiter and Schmidhuber (1997) proposed the LSTM neural network, a form of the RNN that excels at considering Short-term and long-term dependency. The LSTM-based model is superior to RNN in terms of performance due to its three gate networks, namely the output gate, forget gate and input gate. The incorporation of new information into the cell state is done by the input gate. The output gate plays a significant role in determining the amount of information that should be displayed. The Forget gate discards the unnecessary information from the cell state and also tackles the issue of gradient disappearance. Hence all three gates play an important role in keeping the flow of information.

One architectural design that proves effective in modelling sequence data is the LSTM network. This type of network can utilize the information retained from the preceding state to successfully accomplish the current task. Its efficiency in processing time series data has led to its widespread adoption in works concerning stock market prediction. According to Wu et al.(2021), both Li et al.(2017) and Vargas et al.(2017) utilized advanced techniques to predict the performance of the CSI300 index. Wu et al.(2021) confirmed that Li et al.(2017) integrated investors' sentiment into the LSTM neural network, while Vargas et al.(2017) analysed

complex patterns by combining the CNN-LSTM architectures and examined the impact of incorporating technical indicators using the CNN and LSTM methods. Wu et al.(2021,p. 24) proposed a new hybrid recurrent neural network (CH-RNN) based on intermodal tracking is proposed to predict the stock market. Experiments demonstrate the effectiveness of CH-RNN.

Usmani and Shamsi(2023, pp. 12-13) highlighted that Jiawei and Murata proposed a business forecasting system based on an LSTM network, the experiment showed that an iterative neural network with LSTM can manage financial time information better than forecasting system time. A combined deep learning model with multiple CNNs and bidirectional LSTM units is proposed, and the experiment shows that the Combined model can improve prediction performance by 9% over the standard model. Chen, Zhou, and Dai (2015) used the LSTM model to estimate the calculated returns (Bhandari et al.,2022). Historical data is converted to a 30-day long interval with 10 study features and a 3-day study schedule. The Study utilized Adam as the optimizer for its established better performance in predicting stock price. The authors claim that the LSTM model improves accuracy from 14.3% to 27.2% compared to random prediction methods. The Haar wavelet was utilised to transform and denoise financial time series data, then the stacked autoencoder was implemented to learn the characteristics of the data, and the LSTM model was proposed to predict the closing price of the S &P 500 stock index (Bao, Yue and Rao, 2017). Average R-scores are above 88% for the LSTM model that predicted the future price of the S&P 500 index.

An LSTM model was applied to the NIFTY 50's index price for the period data ranging from 2011 to 2016 (Roondiwala, Patel and Varma, 2017). The author uses key data (opening price, closing price, low price and high price) to predict the closing price without the utilizing of macroeconomic and technical indicators. Fischer and Krauss(2018) used LSTM networks to solve the classification problem of predicting changes in the S&P 500 index between 1992 and 2015. The authors conclude that LSTM networks can extract valuable insights from real-time financial data. The study incorporated Adam as the optimizer as it outperformed other optimizers. LSTMs outperform normal forest, deep regression models, and logistic regression to predict the daily returns on exchange rates.

Qiu, Wang and Zhou(2020) proposed an LSTM model that used based historical price data from the S&P 500, Dow Jones Industrial Average (DJIA), and Hang Seng Index datasets. They used monitoring techniques to extract information from the media to measure price changes. The authors used the wavelet transform to identify the data and then followed the observation-

based LSTM function to estimate the open price of the index using only the relevant business data. Lanbouri and Achchab (2020) proposed an LSTM model for frequency market visualization and used S&P 500 stock market data to predict future index prices at 1, 5 and 10-minute intervals.

Yadav, Jha, and Sharan(2020) applied an LSTM model with multiple hidden layers to Indian business data, removing patterns and seasonal factors, to predict closing prices of the nifty50 index. The study concluded that Multi-layer LSTM outperforms the single-layer LSTM. Kara et al.(2011) used support vector machine (SVM), negative logistic regression and LSTM to estimate daily fluctuations in the Country 100 Index of the Istanbul Stock Exchange from 1997 to 2007(Bhandari et al.,2022) The author selected ten instructional activities as an input to the model. Experimental results show that the average performance of the LSTM model is better than the SVM and negative logistic regression model.

Karmiani et al.(2019) conducted a study that compared the performance of LSTM, SVM and Kalman filters for stock price prediction(Perry,2023). The stock prices of the nine selected companies were taken into account in the estimation. LSTMs are the best choice for accuracy and low variance. Yu and Yan(2019) combined the phase space reconstruction method of time series analysis and the LSTM model to predict stock price(Yang, Wang and Li, 2022). Historical prices of various market indices such as S&P 500, Dow Jones Industrial Average, Nikkei 225, Hang Seng, CSI 300 and ChiNext were taken into account in the analysis. The study compared the support vector regressor and ARIMA and LSTM and established that the LSTM model outperforms other models in predicting the price and moment of the studied indices. Rana, Uddin and Hoque (2019) conducted a study that compared the prediction performance of LSTM, Linear Regression and Support vector machine to predict the stock price of Spanish company named Acciona. LSTM model outperformed linear regression and SVM model. Researchers further analysed the effect of different activation function and optimizers. The Linear activation function and ReLU activation function with ADAM as optimizer performed better than tanh and adamax optimizer. Models with Adam as optimizer and Activation function such as Linear and ReLU gave approximately 98.5% accuracy.

Shahi et al.(2020) conducted a comparative study of four machine learning algorithms (Multilayer Perceptron, LSTM, Convolutional Neural Network, and Uncertainty-Aware Attention) to predict the next day's stock price. The S&P 500, CSI 300, and Nikkei 225 represent the most innovative, underdeveloped, and developed markets, respectively. The

opening price, closing price, trading volume, moving average convergence-divergence, average true range, exchange rates and interest rates are considered predictors. The results showed that Distraction-Aware Attention outperformed the other models. In addition, additional estimators such as the volatility index and unemployment rate can improve the performance of the model.

2.3 COMPARATIVE APPROACH

Orsel and Yamada(2022) incorporated Kalman filter, single-layer LSTM, dual-layer LSTM, Bi-directional LSTM and CNN-LSTM algorithms to historical stock prices of different stocks over a 10-year range for the period between 01-01-2011 to 01-01-2021. In the study, Tesla and Microsoft stock prices were analysed. Karman filter performed better than LSTM algorithms on the Microsoft stocks, a very low volatile stock. LSTM algorithms outperformed the Karman filter on the Tesla stock, a very volatile stock. Researchers utilised Root mean square error(RMSE), Mean Absolute Error(MAE) and R-value to compare the models. The Study concluded that a linear Karman filter predicted the stock price of a low-volatile nature better than complex LSTM algorithms whereas the LSTM algorithm performed better on high-volatile stocks.

Rather(2021) compared the performance of LSTM-DNN with the Auto-regressive moving pointer(ARMP) model and other statistical models such as ARIMA, Exponential Smoothing, and Holt-Winters for Stock Prediction and Portfolio Optimization. The researcher incorporated historical prices of all 50 stocks of nifty 50 for the period 06-10-2010 to 05-10-2020. Initially, all five algorithms were used to build the model on non-stationary data. The author has used metrics such as RMSE to compare the performance of all the models. LSTM-DNN outperformed the ARMP and other statistical models such as ARIMA, Exponential Smoothing, and Holt-Winters. The non-stationary time series was transformed into stationary time series data by utilizing the log transformation and then the square root transformation. After the transformation, All models were applied to the stationary times series data to compare the performance. The application of the transformation improved the performance of all models and LSTM-DNN managed to improve the performance difference from other models. Later, the LSTM-DNN were utilised to optimise the portfolio by inputting the respective weights and the expected return obtained, target variable as the expected return sought. The results were compared with Markowitz's mean-variance model and Mean Semi-variance model. LSTM-

DNN model outperformed both models in reference to portfolio optimization by diversifying the risk better and giving better overall returns.

Shahi et al.(2020) proposed a study that compared the performance of LSTM and gated recurrent units (GRU) for forecasting the stock market. The features such as financial news sentiment scores and historical price data such as open price, last traded price (LTP), high price, low price and volume traded from 2011-03-20 to 2019-11-19 of Agricultural Development Bank. The researcher combined the sentiment score and price data into a single vector to produce a sequence of time series. A min-max scalar was applied to transform the data by scaling the value between the range 0 to 1. LSTM and GRU models were implemented with an epoch value of 100 and batch size of 30 on the generated sequence to predict the last trading price of the stock. The study incorporated metrics such as MAE, RMSE, co-efficient of determination and Directional Accuracy to compare the performance of the model. When only scores of news sentiment were inputted, then LSTM model performed better than then GRU model as GRU performed marginally better when historical price data was inputted. When the combined vector of the sentimental score and historical price data was inputted the performance of the LSTM and GRU model improved considerably.

Althelaya et al.(2021) proposed a study to predict the index price of S & P 500 by utilising the Bi-directional LSTM and Stacked LSTM. The study also tried to compare the different transformation techniques such as empirical wavelet transformation and stationary wavelet transformation and their effect on the performance of both LSTM models. The study incorporated metrics such as MAE, RMSE, and co-efficient determination to compare the performance of the model. Empirical wavelet transformation outperformed stationary wavelet transformation in both models. The bi-directional model performed exceptionally well on long-term and short-term predictions but Stacked LSTM performed better on short-term predictions.

Saud and Shakya(2020) conducted a study that compared the performance of deep learning algorithms such as GRU, LSTM and Vanilla RNN. All three models were used to predict the stock price of two banking stocks namely, Himalayan bank Ltd. and Everest bank Ltd. listed on the stock exchange in Nepal. The performance metrics MAPE was used to compare the performance. The study incorporated look look-back period ranging from 2 to 30. Researchers conducted 10 experiments on every set of values for the look-back period. The LSTM outperformed GRU in 6 experiments. The average MAPE score has established that vanilla RNN was outperformed by GRU and LSTM in all 10 experiments.

2.4 APPLICATION OF LSTM ON DIFFERENT DATASETS

Stock price forecasting is currently a hot topic with broad prospects, but it also encounters problems. Recently, many stock price prediction methods have emerged. LSTM has been the most preferred architecture to predict the stock price because of its ability to retain long-term memory. This section looks at three different data sets to analyse the performance of different standalone and combined LSTM models as well as other deep learning models.

2.4.1 FIVE SHARES OF COMPANIES LISTED UNDER Shanghai A-market

Wu et al.(2021) introduced a new approach to LSTM to estimate the stock price known as S_I_LSTM. Researchers discuss the effect of conventional information (business histories and reports) and non-conventional information (announcements and financial news) on bid prices. The information used as input includes background information, technical price indicators, and non-conventional information such as scores of financial news sentiment. The stock price of the five companies used in the study is CITIC Securities Co. Ltd., Bank of China Ltd., ICBC Ltd, China Construction Bank Ltd and Agricultural Bank of China. Next, the Authors used a convolutional neural network-based sentiment analysis method on non-normal data to estimate the score for investor sentiment.

The study incorporated the requirements, performance indicators, and historical market data as a method for price prediction and used a network based on short-term memory to predict the price of the index of the Shanghai stock market in China. Experiments show that the estimated closed value is much nearer to the actual closed value compared to the individual data, and the error of the mean absolute error reached to value of 2.40. The data utilised in the research contained 3,500 newspapers and 24,600 forums published over 4800 business days. China's Internet financial reporting, which corresponds to the period July 1, 2017 - April 30, 2020, was also recorded by capturing economic historical data for the same period. The background file consists of 4800 lines, and every line consists of six lines of information, that also includes the transaction date.

Wu et al,.(2021) found that features extracted from news media were more influential on prediction results than news content. Based on this influence, only financial news remains in the records. The analysis incorporated data from 2017-07-01 to 2019-12-31 as training data, and information from 2020-01-01 to 2020-04-30 as a test data set. Researchers also investigated whether the indicators are relevant to the stock forecast and proposed an in-depth study based on various sources to analyse the Shanghai A-market stocks. The plan incorporates

the investor's view and sentiment based on financial news into the stock price prediction. The plan can also provide investment advice to investors and can be used to bring in real capital, which has some key features. However, due to limited time and research capacity, there are some limitations to this article that need to be developed into further research. For example, less judgment was made when learning about domain names. Additionally, the cycle or level of granularity of stock estimates that may add up to research results should be taken into account.

2.4.2 S&P 500 DATASET

Ma et al.(2021) conducted a study to compare the performance of single-layer LSTM, multi-layer LSTM and Combined model of LSTM-Markow. In this study, the popular US stock index S&P 500 is used for model prediction. The feature selection process involves identifying key points that cause changes in parameter values. Certain variables such as sentiment information about the market from news and technical price indicators are omitted from the study. Variables such as Micro-economic and macroeconomic features were selected based on their influence on the broader economy. Data were collected for a total of 15 years, from 2006 to 2020. The period chosen includes 2 important bearish markets namely, the Financial crunch in 2008-09 and the Coronavirus pandemic in 2020. Closing prices of index S & P 500, Micro & macro-economic data, and underlying index indicators. Micro and macro-economic data consists of unemployment and consumer confidence information. The combined vector of all 3 features forms 3 classes and was features engineered to obtain 9 new features.

Ma et al.(2021) used the Adam optimizer as the study indicated that Adam outperform other optimizers. All three proposed algorithms were used to predict the next day's closing price of the S&P 500 Index. The study used performance metrics such as the root mean square error (RMSE), MAPE, and correlation coefficient (R) to compare the performance of all three proposed models. The results established that the LSTM-Markow model outperforms other LSTM models with comparatively lower scores on RMSE and MAPE and also scores high on R-value compared to other standalone LSTM models.

2.4.3 CSI 300 INDEX CORE STOCK DATA SET

Shah, Vaidya and Shah (2022) Conducted a study to compare the performance of ARIMA model, the LSTM model, the MI-LSTM model, the Bi-LSTM model, the LSTM-DRNN model, the CNN model, the GC-CNN model, the CNN-LSTM model, the CNN-TLSTM model, and the CNN-Bi LSTM model to predict the next day opening price of 261 stocks from CSI 300

index and also treated CSI 300 index price as feature. The Stokes historical opening prices were collected from the period April 25, 2013, to May 15, 2017. Researchers used a time window of size T = 10 days and a step size of 1 day to generate a total of 249,516 samples from 262 features, 171,216 training samples, 52,200 valid samples, and 26,100 test samples. The metrics such as RMSE, MSE and MAE were used to compare the performance of different models.

In this paper, researchers propose an improved MI-LSTM based on LSTM and a tracking mechanism, which is more effective in extracting confidential information and filtering noise. The MI-LSTM unit assigns different weights to different inputs to maintain control of the average when pulling data from the input ports. The output of MI LSTM is further processed by automatic tracking of time. Based on these levels of thought, the model not only focuses on the most important information but also adapts to the timing of the most important steps. Using 3 additional functions and Gaussian noise, researchers created experiments to demonstrate improvement over the other models. The Proposed MI-LSTM model outperformed all other models.

2.5 SUMMARY

Using historical data to predict future market prices has been a hot topic for decades. Stock prices are often affected by many factors. The researchers have to be intuitive as well as empirically establish the use of certain features and their efficacy in predicting the stock price or underlying asset. The literature review has motivated the proposed study to incorporate research flow, methodology to retrieve data, data preparation, selection of certain hyperparameters, selection of certain performance metrics such as RMSE, and MSE as Loss metrics , Adam as optimizer and ReLU as activation function in the hidden layer and Linear activation function in the output layer in the study. The literature review was helpful in selecting optimizers such as ADAM over other optimizers based on better performance. The literature review sheds light on the performance of activation functions such as Linear and ReLU, when combined with ADAM optimizer has performed better than other activation functions. The insight provided by the literature review has helped immensely to conduct a comparative study of single-layer, multi-layer and bi-directional LSTM, and propose an LSTM-based model to predict the selected stock price of Yes Bank Ltd.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 INTRODUCTION

Yesbank's historical price data were extracted from Yahoo finance data using the yfinance library in Python. The initial Features utilised for the rudimentary stage of analysis were the Close price, Open price, High price and Low price of the Yes bank for the period from 2005-05-12 to 2023-08-09. Pandas profiling library was utilized to conduct univariate, bi-variate analysis. 80% per cent of the data was kept as Training data and 20% of the data was treated as test data. 30% of the data in the training dataset was treated as Validation data.

Hypothesis testing was done on the Yes bank stock price data set to determine if the data was stationary or non-stationary. According to Usmani and Shamsi(2023), Analysis of times series data is more efficient when the non-stationary element of the data is removed and the Dickey-Fuller statistical test can be utilized to perform non-stationary check and a combination of log-square root transformation can be applied to remove the non-stationary nature of the data. In this study, The Dickey-Fuller test was used as a statistical tool to perform hypothesis testing. Log transformation and then square root transformation were performed to flatten it. Dicky-fuller test confirmed that transformed data is stationary with more than 99% confidence. An intuitive method was implemented to create a new data set from transformed data. The previous three days' close prices were used to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. Training and test data were converted into a NumPy array to feed into the LSTM model.

A comparative approach has been implemented in the previous studies by Orsel and Yamada(2022), Shah, Vaidya and Shah (2022), Saud and Shakya(2020) and Shahi et al.(2020) as discussed in Chapter 2, Literature Review, subsection 2.2, 2.3.2 and 2.3.3. Initially, three baseline models were built, namely single-layer LSTM, multi-layer LSTM and bi-directional LSTM. 15 experiments were conducted on each baseline model. Hence total of 45 experiments were conducted. One best model was selected from the experiments conducted on three baseline models with different combinations of the number of epochs[10,20,50,100] and batch size[32,64,128,256]. Loss scores on the validation dataset of the baseline model and selected Experimental model were compared. An experimental Bi-directional model with epochs:10

and batch size: 64 was selected for hyperparameter tunning because it has the lowest Loss scores on the validation dataset of 322.02.

Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was implemented on the test data set. Hyperparameters of the proposed model are as follows: Epochs:10, batch size:64, Dropout rate: 0.4, learning rate: 0.0013489304109835225, Number of layers:12. Input units are the number of RNN units in each layer. input_unit1: 80, input_unit2: 32, input_unit3: 16, input_unit4: 16, input_unit5: 16, input_unit6: 16, input_unit7: 16, input_unit8: 16, input_unit9: 16, input_unit10: 16, input_unit11: 16, input_unit12: 16(refer to Table 4.8.1, trial 1). Ma et al.(2021) used the Adam optimizer as the study indicated that Adam outperform other optimizers. Adam optimizer was selected based on understanding from the literature review.

The proposed model predicted the stock price very close to actual values with Loss and RMS E scores of 25.004 and 5.02. and also captured the moment of the stock very well as discussed in Chapter 5, Results.

3.2 ARCHITECTURE

3.2.1 RNN ARCHITECTURE

Recurrent neural network (RNN) is a type of neural network (NN) used for real-time prediction. The weights are obtained from the input process and proceed to the first layer. The previous layer provides the weight to the succeeding layer. This study has used RELU as an activation function for the hidden layer. The output layer by default utilizes the Linear function. The Keep(back) occurred between the first hidden layer(previous time t-1) and the input layer will be used in the current time(t). The next input manages the noise at the previous input(refer to Figure 3.2.1.1).

Consider the input time as x(t) and output time y(t) for time series data. The weight matrices for the three connections are W_{LH}, W_{HH}, and W_{HO}; hidden. f_O and f_H are the activation functions for the output unit. Equation (1) and (2) defines the nature of the RNN.

$$h(t+1) = f_H\left(W_{IH}x(t) + W_{HH}h(t)\right) \quad (1)$$

$$y(t+1) = f_O\left(W_{HO}h(t+1)\right). \quad (2)$$

Past behaviour's information of the system is summed up by a set of values known as h(t) that gives a definition to its future behaviour.

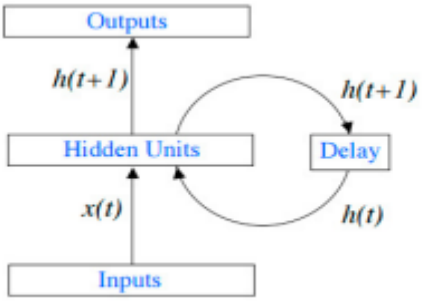


Fig 3.2.1.1 Recurrent Neural Network Model

Shah et al.,(2021). Available at: https://ieeexplore.ieee.org/document/9510147.

3.2.2 LSTM ARCHITECTURE (SINGLE-LAYER AND MULTI-LAYER LSTM)

Hochreiter and Schmidhuber (1997) proposed the LSTM neural network, a form of the RNN that excels at considering Short-term and long-term dependency. A single-layer LSTM has one layer of LSTM whereas Multi-layer LSTM has at least 2 layers of LSTM in the architecture. LSTM are more precise compared to the traditional RNNS because of its long-range dependencies. LSTM can manage the backflow error issues in RNNs efficiently in Backpropagation algorithms. In the recurrent hidden layer of LSTM exist memory blocks. These special units consist of cells(memory)that manage the temporal state of the network and also gates to manage the information's flow. There are three Gates in the memory block as follows:

- Input gate: manages cell activation's input flow into the cell(memory).
- Output gate: manages cell activation's output flow to the network.
- Forget gate: The cell's memory is forgotten or reset by this gate.

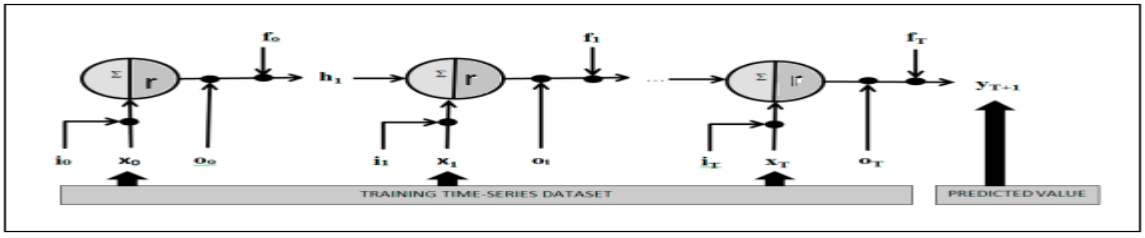


Fig 3.2.2.1 LSTM architecture

Shah et al.,(2021). Available at: https://ieeexplore.ieee.org/document/9510147.

Each LSTM block receives input signal (x), input gate signal (i), recurrent signal (h), and forget gate signal (f); and infers the output gate signal (o). Memory block of the LSTM's process flow is shown in Fig 3.2.2.2.

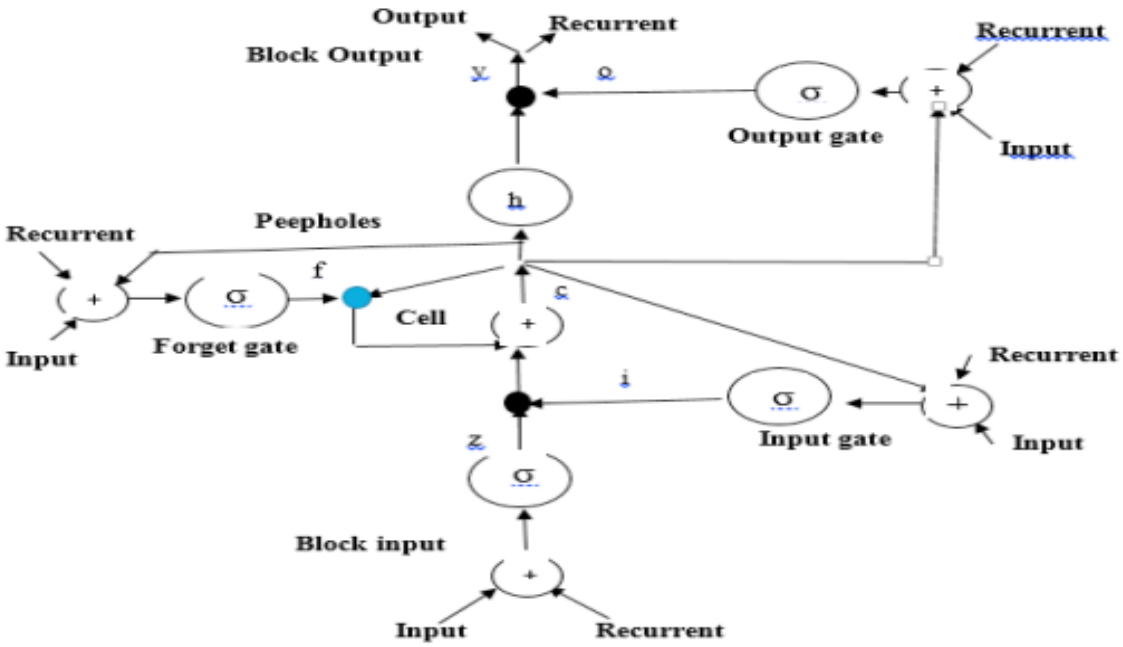


Fig 3.2.2.2 LSTM's Computational flow diagram
Shah et al.,(2021). Available at: https://ieeexplore.ieee.org/document/9510147.

A sequence of input x = (x1, . . , xT) is mapped by the LSTM model to a sequence of output y = (y1, . . , yT) by calculating the network unit activations using the following equations iteratively from t = 1 to T as follows.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (3)$$
$$z_t = tanh(W_z x_t + U_z h_{t-1} + b_z) \quad (4)$$
$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \quad (5)$$
$$C_t = i_t * z_t + f_t * C_{t-1} \quad (6)$$

$$o_t = \mathbf{L}(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \quad (7)$$
$$h_t = o_t * \mathbf{relu}(C_t) \quad (8)$$

Wi, Wz,Wf, Wo,Ui,Uz,Uf,Uo are the parameters of the model calculated by the process in the training model. Activation functions in the output layer and hidden layer in the study are L(Linear activation) and ReLU(Rectified Linear Unit) respectively. Biases are represented by b in the above equation. ReLU with input X can be defined as the max function(X,0). ReLU outputs value between X and 0. All negative value are given 0 by the ReLU activation function. The Output layer has a Linear activation function. It gives out the output as the same value X.

3.2.3 BI-DIRECTIONAL LSTM

This special form of LSTM has similar flow of process and it has additional mechanism of the Backward Layer. Weight gets updated when the flow of information through backward propagation as shown in fig 3.2.3.1(Shah et al., 2021).

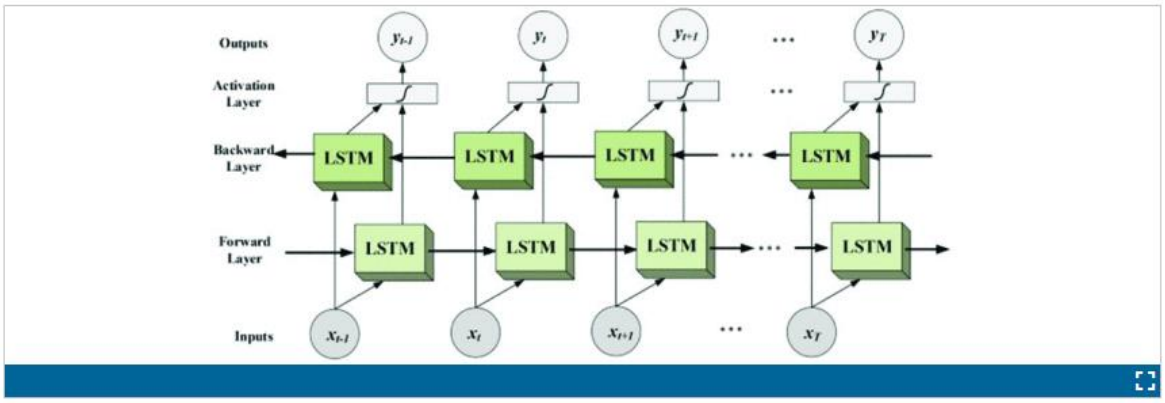


Fig 3.2.3.1 Bi-directional LSTM
Shah et al.,(2021). Available at: https://ieeexplore.ieee.org/document/9510147.

3.2.4 EVALUATION METRICS

1. Mean Square Error (MSE): In the below formula, the number of data points n , Y_i is the actual share price and $\hat{Y}_i$ is our predicted stock price. It will be used as metrics for Loss.

$$MSE = \left(\frac{1}{n}\right)\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

2. Root mean square error, which is a metric that tells us the average distance between the predicted values from the model and the actual values in the dataset.

   The lower the RMSE, the better a given model is able to “fit” a dataset.

   The formula to find the root mean square error, often abbreviated RMSE, is as follows:

   $RMSE = \sqrt{\Sigma(P_i - O_i)^2 / n}$

   where:

   Σ is means summation
   P_i is the predicted value for the i^{th} observation in the dataset
   O_i is the observed value for the i^{th} observation in the dataset
   n is the sample size

3.3 DATA PREPARATION

3.3.1 DATA PROCUREMENT

Yesbank historical price data with features, Date, Open price, High price, Low price, and Close price were retrieved from Yahoo finance data using the yfinance library in Python.

3.3.2 DATA PRE-PROCESSING/ TRANSFORMATION/FEATURE ENGINEERING

The data was checked for inconsistency such as missing value, and high co-relation among explanatory variables. There were 0% of missing values in all five features. There is no missing value in the data set. Hence, the study did not incorporate any missing value treatment.

There exists a high correlation among Close, Open, High and Low. Hence, only the Date and Close features were utilized for further analysis. 80% per cent of the data was kept as Training data and 20% of the data was treated as test data. 30% of the data in the training dataset was treated as Validation data by keeping the value of validation split at 30% in the model fitting process. According to Usmani and Shamsi(2023), Analysis of times series data is more efficient when the non-stationary element of the data is removed and the Dickey-Fuller statistical test can be utilized to perform non-stationary check and a combination of log-square root transformation can be applied to remove the non-stationary nature of the data. Hence, The Dickey-Fuller test was conducted to check the non-stationary characteristics of the data. The test highlighted the existence of the non-stationary nature of the data.log transform, and then the square root transform was applied to make the data stationary. The Dickey-Fuller test was conducted again to check whether the transformation of the data changed the nature of the data to stationary.

A new data set was created with new features by implementing feature engineering. The previous three days' close prices were used to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. The new data set has five new features as follows:

Date: Date of the trading day.

Input_1: The closing price of the 1^{st} day used for prediction.

Input_2: The closing price of the 1^{st} day used for prediction.

Input_3: The closing price of the 1^{st} day used for prediction.

Target: The closing price of the 4^{th} day is used as the target/predictor variable.

The Data was converted into a NumPy array to be fed into the LSTM model.

3.4 MODEL AND EXPERIMENT DESIGN PHASE

A comparative approach has been implemented in the previous studies by Orsel and Yamada(2022), Shah, Vaidya and Shah (2022), Saud and Shakya(2020) and Shahi et al.(2020) as discussed in Chapter 2, Literature Review, subsection 2.2, 2.3.2 and 2.3.3.Three baseline LSTM models (Single-layer, Multi-layer and bi-directional) were created.

This approach was influenced by different past studies. Ma et al.(2021) conducted a study to compare the performance of single-layer LSTM, multi-layer LSTM and Combined model of LSTM-Markow. Shah, Vaidya and Shah (2022) Conducted a study to compare the performance of ARIMA model, the LSTM model, the MI-LSTM model, the Bi-LSTM model, the LSTM-DRNN model, the CNN model, the GC-CNN model, the CNN-LSTM model, the CNN-TLSTM model, and the CNN-Bi LSTM model to predict the next day opening price of 261 stocks from CSI 300 index and also treated CSI 300 index price as feature. Zou and Qu(2022) conducted a study to perform quantitative trading by comparing different models such as Stacked LSTM, ARIMA model, and LSTM combined with the Attention model.

Single-layered LSTM had one LSTM layer while others had two LSTM layers. For all baseline models, the activation function for layer/layers was RELU, RNN units were 128, the dropout value was 0.2, the optimizer was Adam with a learning rate of 0.004, the loss function was MSE and the performance metrics as RMSE. Early stop and model check was implemented to optimise the resource usage and select the best performing model. 45 experiments were conducted on three baseline models, 15 experiments each on a particular baseline model with the possible number of combinations for the number of epochs[10,20,50,100] and batch size[32,64,128,256] The best models with the lowest Validation loss have been saved from the experiment for three forms of LSTM. The performance of Three baseline models and Three experimental models were compared to select the model with the lowest Validation loss score(MSE)for hyperparameter tuning. One best model each was selected from the experiments conducted on three baseline models with different combinations of the number of epochs[10,20,50,100] and batch size[32,64,128,256]. Loss scores on the validation dataset of the baseline model and selected Experimental model were compared. An experimental Bi-directional model with epochs:10 and batch size: 64 was selected for hyperparameter tunning because it has the lowest Loss scores on the validation dataset of 322.02.

3.5 HYPERPARAMETER TUNING PHASE

Hyperparameter tuning was conducted on the selected model by a Keras tuner with three experimental trials. Each Experimental Trial incorporated three execution trials. Keras tuner utilised a random search algorithm by tracking the model for the lowest loss.

The range of Hyperparameter detected for tunning are as follows:

- Number of Layers: 2 to 20.
- Drop out: 0 to 0.5 with step size =0.1.
- RNN unit in each layer:16 to 128 with step size =16.
- The learning rate for Adam optimizer: 1e-4 to 1e-2 with sampling as log. Adam optimizer was selected after reviewing various studies(Ma et al.(2021), Wu et al,.(2021), Saud and Shakya(2020), and Usmani and Shamsi(2023)) included in Chapter 2, Literature review, Adam optimizer outperform other optimizers in predicting stock price.

Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was implemented on the test data set.

3.6 TEST DESIGN PHASE

The unseen data for the period from 2020-01-03 to 2023-08-04 from the data set created after feature engineering was utilized in the testing phase. The Data has to go through a similar Data preparation technique as implemented on the training data set. The Dickey-Fuller test was conducted to check the non-stationary characteristics of the data as discussed in study conducted by Usmani and Shamsi(2023). The test highlighted the existence of the non-stationary nature of the data. Initially, the log transform, and then the square root transform were applied to make the data stationary. The Dickey-Fuller test was conducted again to check whether the transformation of the data changed the nature of the data to stationary.

A new data set was created. The previous three days' close prices were used to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. The Proposed model was implemented on the unseen data. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1)(refer to table 5.3.1). The Proposed model was also able to predict the trend closely to the real trend(refer to plot 5.3.1).

The Actual data and predicted data were applied with Square transformation(each value of the data was squared, X**2) and then exponential transformation(np. Exp()) to get the real value of each value point in the data for visualization purposes.

3.7 WORKFLOW OF THE RESEARCH

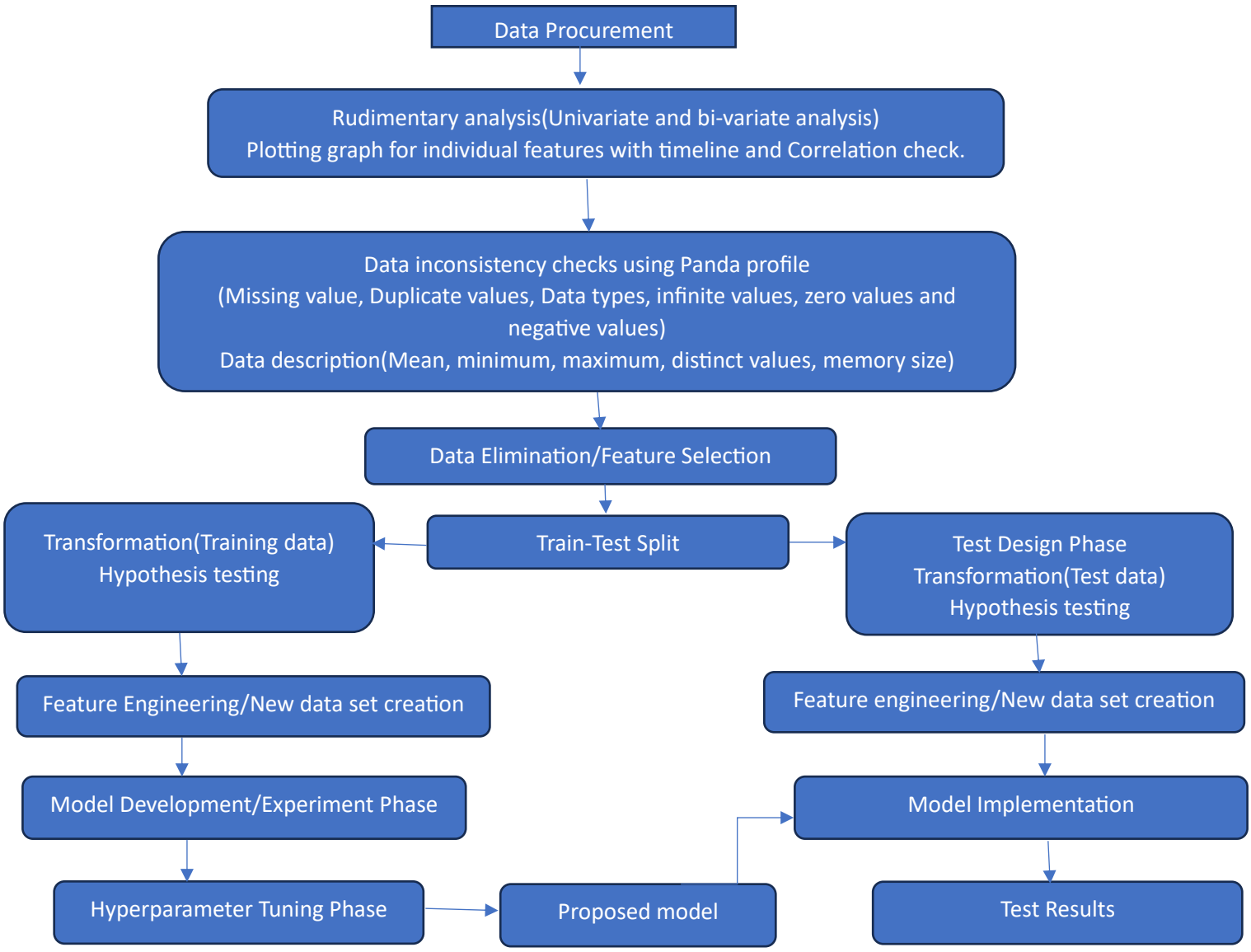


Fig 3.7.1 Workflow of the research

3.8 PROPOSED MODEL (BI-DIRECTIONAL LSTM WITH 12 LAYERS)

The proposed model is as follows(refer to illustration 5.3.1): Epochs:10, batch size:64, Dropout rate: 0.4, learning rate: 0.0013489304109835225, Number of layers:12. Input units are the number of RNN units in each layer. Input_unit1: 80, input_unit2: 32, input_unit3: 16,

input_unit4: 16, input_unit5: 16, input_unit6: 16, input_unit7: 16, input_unit8: 16, input_unit9: 16, input_unit10: 16, input_unit11: 16, input_unit12: 16(refer to Table 4.8.1, trial 1). The total parameters in the model are 175,009. There are 175,009 trainable parameters and 0 Non-trainable parameters.

3.9 SUMMARY

In depth analysis of various studies has laid the foundation for formulating research methodology, research and transformation techniques, hyperparameter tuning techniques, comparative approach of the study, selection of Adam as optimizer and general workflow of the research.

A study by Usmani and Shamsi(2023) helped to formulate the hypothesis testing using dickey-fuller test to check the non-stationary element in data and application of combination of log-square root transformation to make the data stationary. A comparative approach has been implemented in the previous studies by Orsel and Yamada(2022), Shah, Vaidya and Shah (2022), Saud and Shakya(2020) and Shahi et al.(2020). Various studies(Ma et al.(2021), Wu et al,.(2021), Saud and Shakya(2020), and Usmani and Shamsi(2023)) helped to select the Adam as Optimizer.

Review of Studies such as Bhandari et al.(2022),Perry(2023), Biswas et al.(2023),Rather(2021), Yang, Wang and Li (2022), Shah, Vaidya and Shah (2022) for formulating research design and methodology, workflow of the research and understanding of approach of LSTM in stock price prediction. Based on the In depth understanding gained through Literature review and research methodology, Chapter 4, Analysis section of the research was formulated.

CHAPTER 4
ANALYSIS

4.1 INTRODUCTION

Yesbank's historical price data with features, Open price, High price, Low price, and Close price were retrieved from Yahoo finance data. Pandas profiling library was utilized to conduct univariate, bi-variate analysis. Bi-variate analysis finds an existence of high co-relation among the features. The stock's closing price was selected as a feature to be incorporated into the study because of the high correlation among the features.

Hypothesis testing was done on the Yes bank stock price data set to determine if the data was stationary or non-stationary. The Dickey-Fuller test was used as a statistical tool to perform hypothesis testing. The test established that data is non-stationary with more than 99% confidence. Log transformation and then square transformation were performed to flatten it. Dicky-fuller test confirmed that transformed data is stationary with more than 99% confidence. An intuitive method was implemented to create a new data set from transformed data. The previous three days' close prices were used to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. 80% per cent of the data was kept as Training data and 20% of the data was treated as test data. 30% of the data in the training dataset was treated as Validation data. Training and test data were converted into a NumPy array to feed into the LSTM model.

Initially, three baseline models were built, namely single-layer LSTM, multi-layer LSTM and bi-directional LSTM. All three models were built with a number of RNN units=128, Adam as optimizer with a learning rate of 0.004.15 experiments were conducted on each baseline model. Hence total of 45 experiments were conducted. One best model was selected from the experiments conducted on three baseline models with different combinations of the number of epochs[10,20,50,100] and batch size[32,64,128,256]. Loss scores on the validation dataset of the baseline model and selected Experimental model were compared. An experimental Bi-directional model with epochs:10 and batch size: 64 was selected for hyperparameter tunning because it has the lowest Loss scores on the validation dataset of 322.02. Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was implemented on the test data set.

Hyperparameters of the proposed model are as follows: Epochs:10, batch size:64, Dropout rate: 0.4, learning rate: 0.0013489304109835225, Number of layers:12. Input units are the number of RNN units in each layer. Input_unit1: 80, input_unit2: 32, input_unit3: 16, input_unit4: 16, input_unit5: 16, input_unit6: 16, input_unit7: 16, input_unit8: 16, input_unit9: 16, input_unit10: 16, input_unit11: 16, input_unit12: 16(refer to Table 4.8.1, trial 1). Adam optimizer was selected based on understanding from the literature review. The proposed model predicted the stock price very close to actual values with Loss and RMSE scores of 25.004 and 5.02. and also captured the moment of the stock very well. The proposed model was implemented on the unseen test data set containing Yes Bank's share price for the period from 03-01-2020 to 04-08-2023. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1).

4.2 DATASET DESCRIPTION

Yesbank historical price data with features, Date, Open, High, Low, and Close were retrieved from Yahoo finance data using the yfinance library in python. There is 4461 observation in each column with 5 features in the data set. There are 0% of missing values and duplicate rows in the data set. There are 0% negative or zero values. There are 0% of duplicate values in the row. The date has a Date Time variable type and the rest has a numeric data type. (refer to table 4.2.1)

Refer to Table 4.2.2 for the Statistical Description of Individual features in the data set.

Date: Date of the day. Date has 4461(100%) distinct values. Starting date is 2005-07-12 and the ending date is 2023-08-09.

Open: The opening price of the day. Open has 3814(85.5%) distinct values. The minimum open price was INR 7.66 and the maximum open price was INR 386.19. The mean of the open price is INR 87.58.

High: The Highest price of the day. High has 3838(86%) distinct values. The minimum open price was INR 7.67 and the maximum open price was INR 393.20. The mean of the open price is INR 89.04.

Low: The lowest price of the day. Low has 3802(85.2%) distinct values. The minimum open price was INR 5.65 and the maximum open price was INR 382.06. The mean of the open price is INR 85.90.

Close: The closing price of the day. Close has 3559(79.8%) distinct values. The minimum open price was INR 7.42 and the maximum open price was INR 383.47. The mean of the open price is INR 87.40.

Table 4.2.1 Yes Bank Ltd. Historical data Statistics

Dataset statistics

Number of variables	5
Number of observations	4461
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	174.4 KiB
Average record size in memory	40.0 B

Variable types

DateTime	1
Numeric	4

Table 4.2.2 Statistical Description of Individual features in the data set.

Date
Date
UNIQUE

Distinct	4461	Minimum	2005-07-12 00:00:00
Distinct (%)	100.0%	Maximum	2023-08-09 00:00:00
Missing	0		
Missing (%)	0.0%		
Memory size	35.0 KiB		

Open
Real number ($\mathbb{R}_{\geq 0}$)
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	3814	Minimum	7.66620175
Distinct (%)	85.5%	Maximum	386.1922986
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	87.58037058	Memory size	35.0 KiB

High
Real number ($\mathbb{R}_{\geq 0}$)
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	3838	Minimum	7.6662029
Distinct (%)	86.0%	Maximum	393.1998322
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	89.04496585	Memory size	35.0 KiB

Low
Real number ($\mathbb{R}_{\geq 0}$)
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

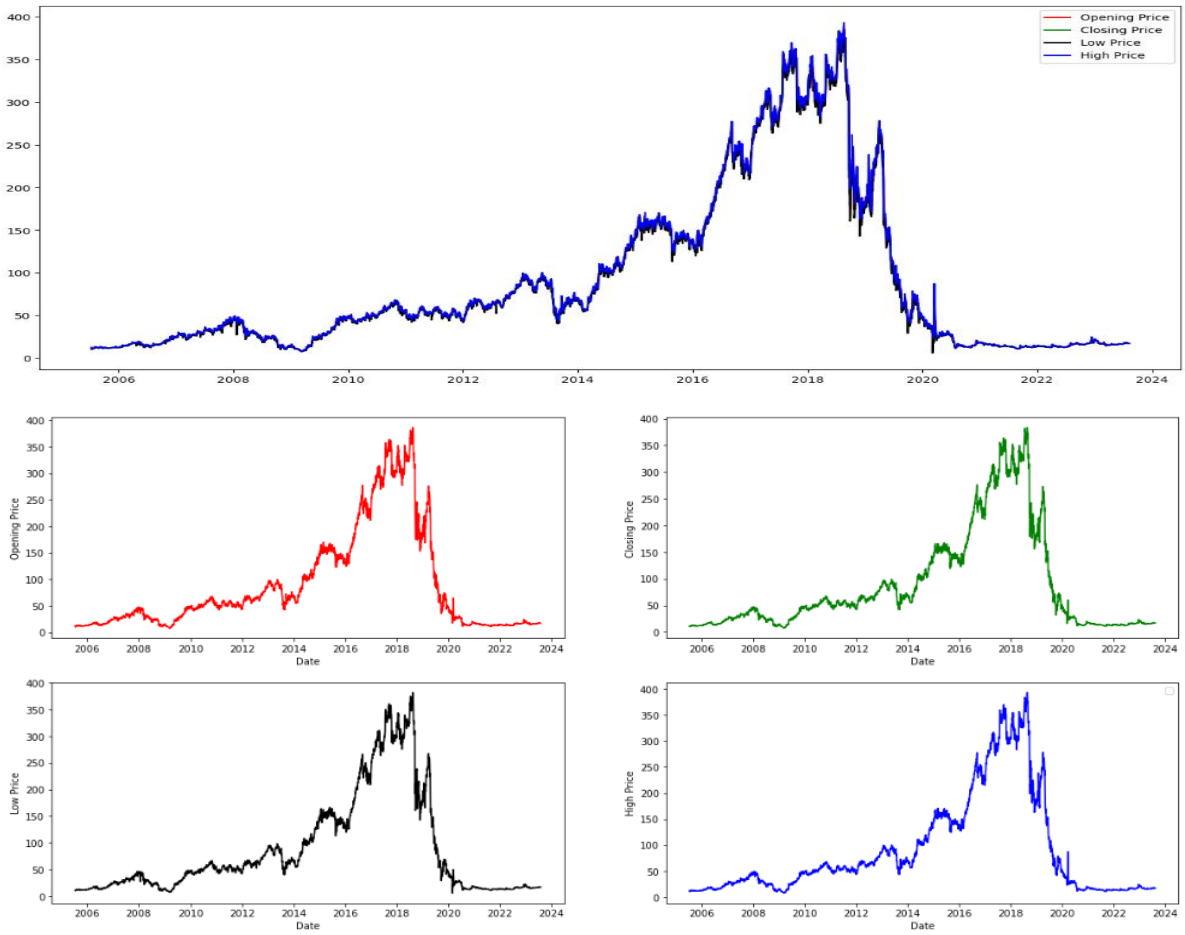
Distinct	3802	Minimum	5.650000095
Distinct (%)	85.2%	Maximum	382.0559142
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	85.89999199	Memory size	35.0 KiB

Close
Real number ($\mathbb{R}_{\geq 0}$)
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	3560	Minimum	7.416604996
Distinct (%)	79.8%	Maximum	383.4671631
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	87.40491556	Memory size	35.0 KiB

4.2.1 UNIVARIATE ANALYSIS

There has been a consistent rise in the price from the year 2005 to 2008 then stock price experienced a consistent fall from the first quarter of the year 2008 to mid-2009 due to the effect of sub-prime loan risk. The stock experienced a consistent rise in price from mid-2009 to the first quarter-2019 then the stock experienced a deep fall in price till mid-2020. The stock has been trading range-bound since mid-2019.



Plot 4.2.1.1 Trend analysis of various features in dataset w.r.t price and time

4.2.2 Bi-variate analysis

Bi-variate analysis establishes a strong correlation among features. The existence of the high correlation resulted in the elimination of highly correlated variables. The date and Close price were selected for further analysis. Open price, High price and Low price were eliminated from the data set. A statistical tool called Spearman's p was used to calculate the correlation among various features(refer to plot 4.2.2.1).

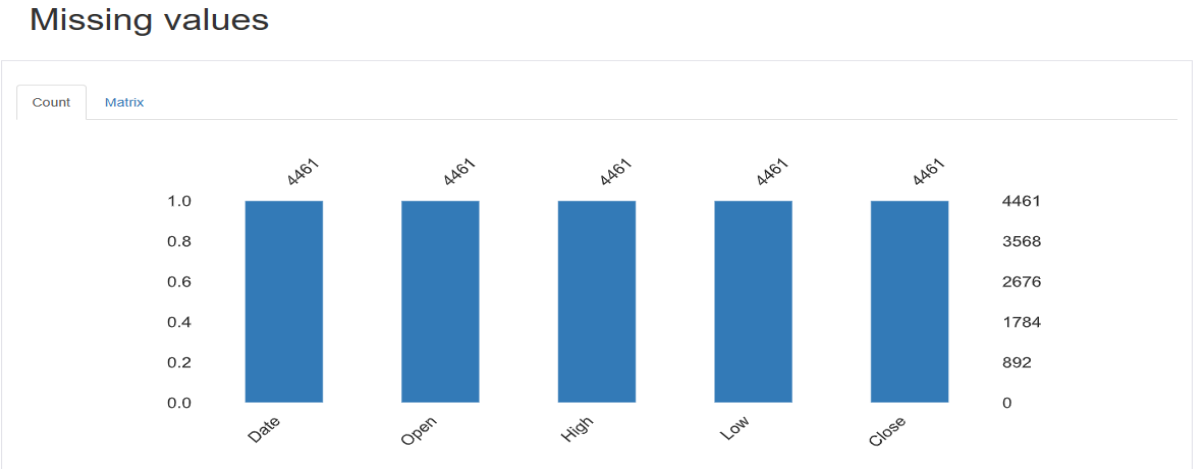
Plot 4.2.2.1 Correlation among various price features.

4.3 DATA PREPARATION

The data was checked for inconsistency such as missing value, and high co-relation among explanatory variables. There were 0% of missing values in all five features. There exists a high correlation among Close, Open, High and Low. Hence, only the Date and Close features were utilized for further analysis. The dickey-Fuller test was conducted to check the non-stationary characteristics of the data. The test highlighted the existence of the non-stationary nature of the data.log transform, and then the square root transform was applied to the data to flatten it.

4.3.1 IDENTIFICATION OF MISSING VALUES

There is no missing value in the data set. Hence, the study did not incorporate any missing value treatment.



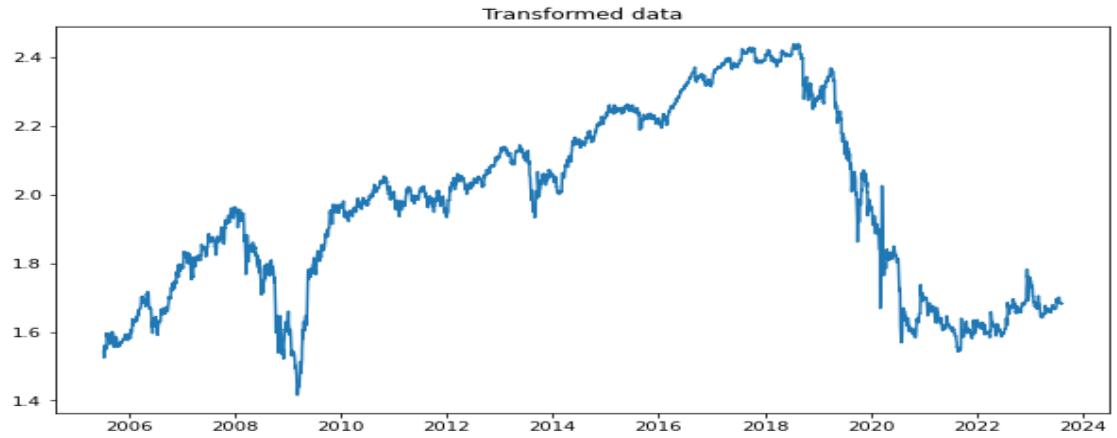
Plot 4.3.1.1 Missing values in the data set

4.3.2 ELIMINATION OF VARIABLES

A high correlation of close to 1 exists among the features. Hence Date and Close price were utilized for further analysis(refer to plot 4.2.2.1).

4.3.3 TRANSFORMATION INTO STATIONARY DATA

Dickey Fuller's experiment confirmed that Data is non-stationary. So, Data was applied with the log transformation and then, square root transformation was applied to make the data stationary(). Then, we conducted the dickey-fuller test to check whether the non-stationary data changed to stationary data as discussed in section 4.4.



plot 4.3.3.1 transformed data after application of log and square transformation

4.3.4 CREATING A NEW DATA SET AND FEATURE(FEATURE ENGINEERING)

The previous three days' close prices were used to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. The new data set has five features as follows:

Date: Date of the trading day.

Input_1: The closing price of the 1^{st} day used for prediction.

Input_2: The closing price of the 1^{st} day used for prediction.

Input_3: The closing price of the 1^{st} day used for prediction.

Target: The closing price of the 4^{th} day is used as the target/predictor variable.

4.3.5 SPLITTING OF ORIGINAL DATASET

80% per cent of the data was kept as Training data and 20% of the data was treated as test data. 30% of the data in the training dataset was treated as Validation data by keeping the value of the validation split at 30% in the model fitting process as shown in the below illustration.

```
Lstm_MultiLayer_Model_history=Lstm_MultiLayer_Model.fit(X_train,y_train,validation_split=0.3,epochs=epoch, batch_size=batch_size, callbacks=[es,mc])
```

Illustration 4.3.5.1 Formulation of validation split during fitting process

4.4 HYPOTHESIS TESTING

The study used the Dickey-Fuller Test to check the stationary nature of the data. The test statistic used in the augmented Dickey-Fuller statistic is a negative number. The higher the negative number, the stronger the rejection of the hypothesis that there is a unit root at some level of confidence.

4.4.1 HYPOTHESIS TESTING(Research Question 1)

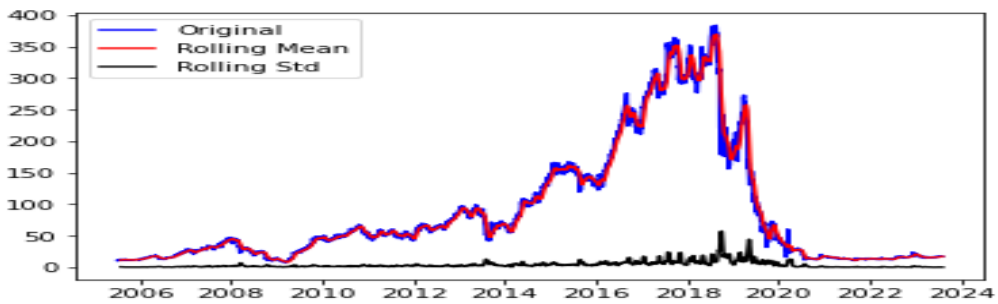
1. Is the data stationary or non-stationary?

The following are the test hypotheses:

Null hypothesis (H0): The time series data is non-stationary.

Alternate hypothesis (H1): The time series is stationary (or trend-stationary).

First, in the below plot 4.4.1.1, the rolling mean is increasing but the standard deviation is more or less constant. The test Statistics of -1.519420 is greater than the critical value at 1%,5% and 10%(refer to table 4.4.1.1). Therefore, It cannot be assumed with any confidence that this is a stationary series Moreover, the p-value is higher than the 1%,5% and 10% threshold(refer to table 4.4.1.1). Hence the null hypothesis cannot be rejected at any confidence level. Meaning that the Dickey-Fuller test verifies that the time series is non-stationary.



Plot 4.4.1.1 Actual price, Rolling mean and Rolling Std before Log-Square root transformation

Table 4.4.1.1 Dickey-Fuller test statistics before log-square Transformation

Results of the Dickey-Fuller Test:	
Test Statistic	-1.519420
p-value	0.523884
Lags Used	30.000000
Number of Observations Used	4430.000000
Critical Value (1%)	-3.431827
Critical Value (5%)	-2.862193
Critical Value (10%)	-2.567117

We applied the log transformation and then we applied the square root transformation to make the data stationary. Then, we conducted the dickey-fuller test to check whether the non-stationary data changed to stationary data.

4.4.2 HYPOTHESIS TESTING(Research Question 2)

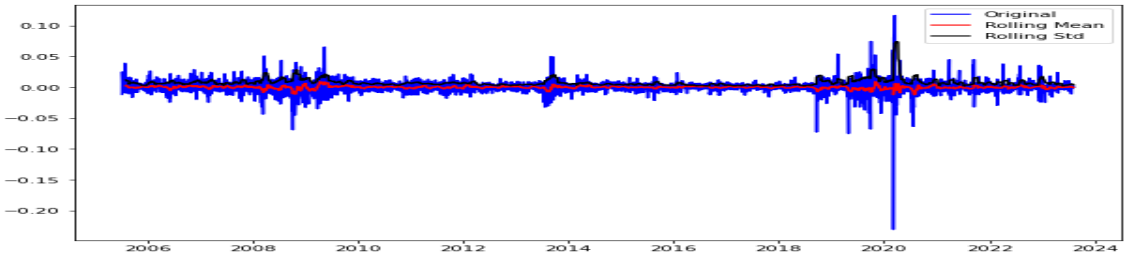
2. Did the transformation of the data change the nature of the data to stationary?

The following are the test hypotheses:

Null hypothesis (H0): The applied transformation technique on time series data did not change the characteristics to stationary.

Alternate hypothesis (H1): The applied transformation technique on time series data changed the characteristics of stationery.

First, in the below plot 4.4.1.1, the rolling mean and standard deviation are more or less consistent with each other's movement over the period. The test statistic of -1.655430e+01 is a lot lower than the critical value at 1%,5%, and 10%(refer to Table 4.4.1.2). Therefore, It can be assumed with more than 99% confidence that this is a stationary series. Moreover, the p-value is smaller than the 1%,5% and 10% threshold, thus the null hypothesis is rejected, meaning that the Dickey-Fuller test verifies that the time series is stationary.



Plot 4.4.2.1 Actual price , Rolling mean and Rolling Std after log-Square root transformation

Table 4.4.2.1 Dickey-Fuller test statistics after log-square root Transformation

Results of Dickey-Fuller Test:	
Test Statistic	-1.655430e+01
p-value	1.932545e-29
#Lags Used	1.500000e+01
Number of Observations Used	4.444000e+03
Critical Value (1%)	-3.431822e+00
Critical Value (5%)	-2.862191e+00
Critical Value (10%)	-2.567116e+00

4.5 Base Line Model design, building and Implementation

Three baseline LSTM models (Single-layer, Multi-layer and bi-directional) were created. Single-layered LSTM had one LSTM layer while others had two LSTM layers. For all baseline models, the activation function for layer/layers was ReLU, RNN units were 128, the dropout value was 0.2, the optimizer was Adam with a learning rate of 0.004, the loss function was MSE and the performance metrics as RMSE. Early stop and model check was implemented to optimise the resource usage and select the best performing model.

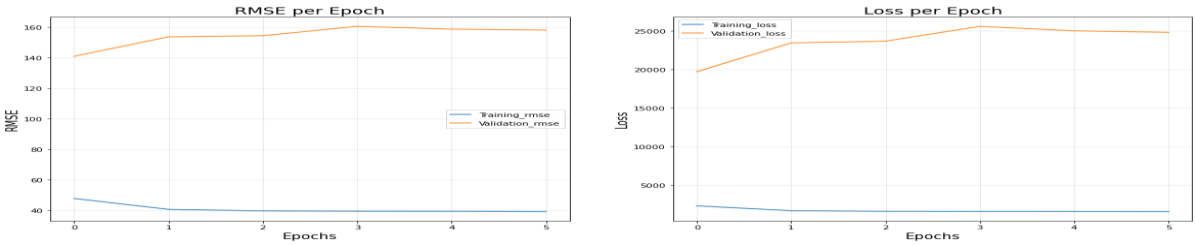
4.5.1 SINGLE-LAYER LSTM BASELINE MODEL

```
# ------------------LSTM----------------------
Lstm_SingleLayer_Model = Sequential()
Lstm_SingleLayer_Model.add(LSTM(activation='relu',units=128, return_sequences=True,
                                input_shape=(X_train.shape[1], X_train.shape[2])))
Lstm_SingleLayer_Model.add(Dropout(0.2))

Lstm_SingleLayer_Model.add(Dense(units=1, activation='relu'))
Lstm_SingleLayer_Model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.004), loss='mse',
                               metrics=[tf.keras.metrics.RootMeanSquaredError()])

Lstm_SingleLayer_Model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 3, 128)            66560

 dropout_2 (Dropout)         (None, 3, 128)            0

 dense_1 (Dense)             (None, 3, 1)              129

=================================================================
Total params: 66,689
Trainable params: 66,689
Non-trainable params: 0
_________________________________________________________________
```

Illustration 4.5.1.1 SINGLE-LAYER LSTM BASELINE MODEL

The metrics score for epoch: 10
Batch Size: 32
Early Stop at Epoch 6 as the model was not improving.
RMSE for training set: 40.88443374633789
RMSE for Validation set: 154.4507064819336
Loss for Training set: 1681.003397623698
Loss for Validation set: 23721.65625



Plot 4.5.1.1 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for baseline single-layer LSTM model

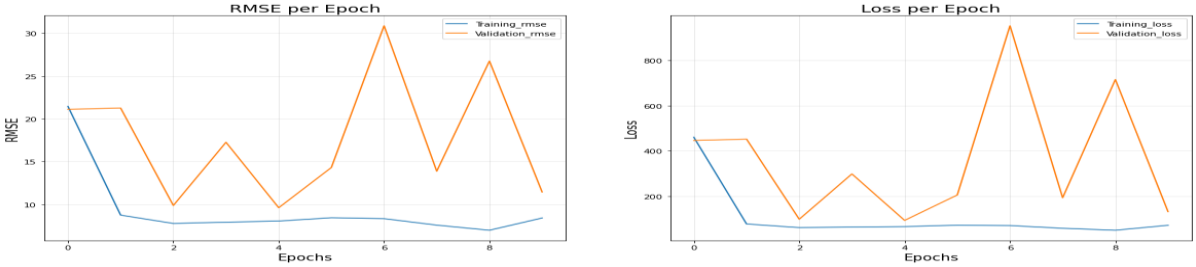
4.5.2 MULTI-LAYER LSTM BASELINE MODEL

```
In [16]: # ------------------LSTM----------------------
Lstm_MultiLayer_Model = Sequential()
Lstm_MultiLayer_Model.add(LSTM(activation='relu',units=128, return_sequences=True,
                               input_shape=(X_train.shape[1], X_train.shape[2])))
Lstm_MultiLayer_Model.add(Dropout(0.2))

Lstm_MultiLayer_Model.add(LSTM(activation='relu',units=128, return_sequences=False))
Lstm_MultiLayer_Model.add(Dropout(0.2))

Lstm_MultiLayer_Model.add(Dense(units=1, activation='relu'))
Lstm_MultiLayer_Model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.004), loss='mse',
                              metrics=[tf.keras.metrics.RootMeanSquaredError()])

Lstm_MultiLayer_Model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_2 (LSTM)               (None, 50, 128)           66560

 dropout_2 (Dropout)         (None, 50, 128)           0

 lstm_3 (LSTM)               (None, 128)               131584

 dropout_3 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 198,273
Trainable params: 198,273
Non-trainable params: 0
_________________________________________________________________
```

Illustration 4.5.2.1 MULTI-LAYER LSTM BASELINE MODEL

The metrics score for epoch: 10
Batch Size: 32
Epoch 10: Early stopping
RMSE for training set: 9.35700888633728
RMSE for Validation set: 17.632488822937013
Loss for Training set: 104.0412826538086
Loss for Validation set: 358.20089492797854



Plot 4.5.3.1 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for baseline bi-directional LSTM model

4.5.3 BI-DIRECTIONAL LSTM BASELINE MODEL

```
# ------------------LSTM----------------------
Lstm_bidirectional_model = Sequential()
Lstm_bidirectional_model.add(Bidirectional(LSTM(activation='relu',units=128, return_sequences=True),
                             input_shape=(X_train.shape[1], X_train.shape[2])))
Lstm_bidirectional_model.add(Dropout(0.2))

Lstm_bidirectional_model.add(Bidirectional(LSTM(activation='relu',units=128, return_sequences=False)))
Lstm_bidirectional_model.add(Dropout(0.2))

Lstm_bidirectional_model.add(Dense(units=1, activation='relu'))
Lstm_bidirectional_model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.004), loss='mse',
                            metrics=[tf.keras.metrics.RootMeanSquaredError()])

Lstm_bidirectional_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 3, 256)           133120
 l)

 dropout_3 (Dropout)         (None, 3, 256)            0

 bidirectional_1 (Bidirectio  (None, 256)              394240
 nal)

 dropout_4 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 1)                 257

=================================================================
Total params: 527,617
Trainable params: 527,617
Non-trainable params: 0
_________________________________________________________________
```

Illustration 4.5.3.1 bi-directional LSTM baseline model

The metrics score for epoch: 10

Batch Size: 32

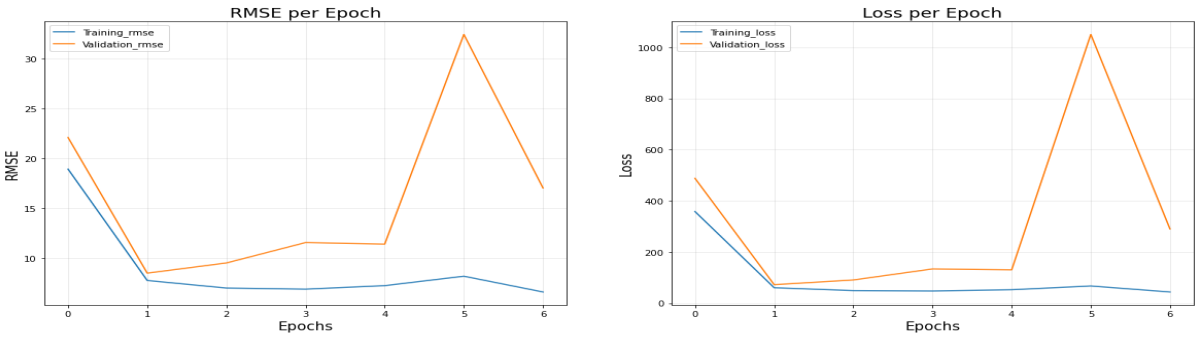
Epoch 7: Early stopping

RMSE for training set: 8.95195620400565

RMSE for Validation set: 16.07170500074114

Loss for Training set: 96.93870053972516

Loss for Validation set: 322.0209285191127



Plot 4.5.3.1 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for baseline bi-directional LSTM model

4.6 EXPERIMENT ON DIFFERENT BASELINE MODELS

4.6.1 SINGLE-LAYER LSTM EXPERIMENTS

Experiment 1:

The number of epochs for this experiment is: 10

The Batch Size is: 64

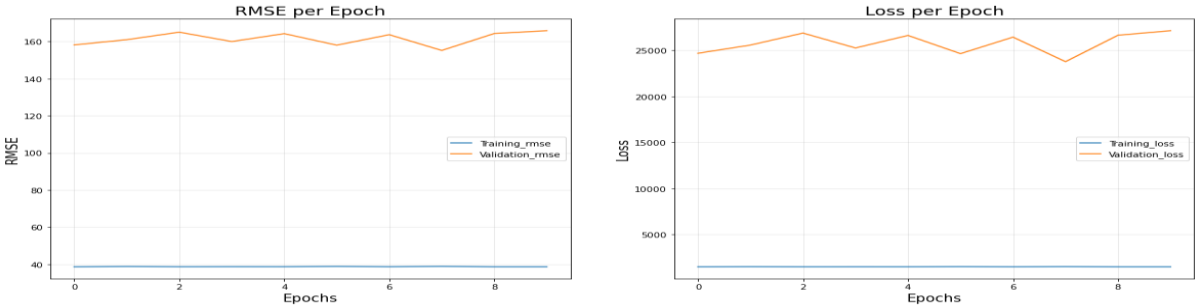
Epoch 10: Early stopping

RMSE for training set: 38.74926414489746

RMSE for Validation set: 161.47740478515624

Loss for Training set: 1501.5107177734376

Loss for Validation set: 25772.4705078125



Plot 4.6.1.1 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 1 for single-layer LSTM

Experiment 2:

The number of epochs for this experiment is: 10

The Batch Size is: 128

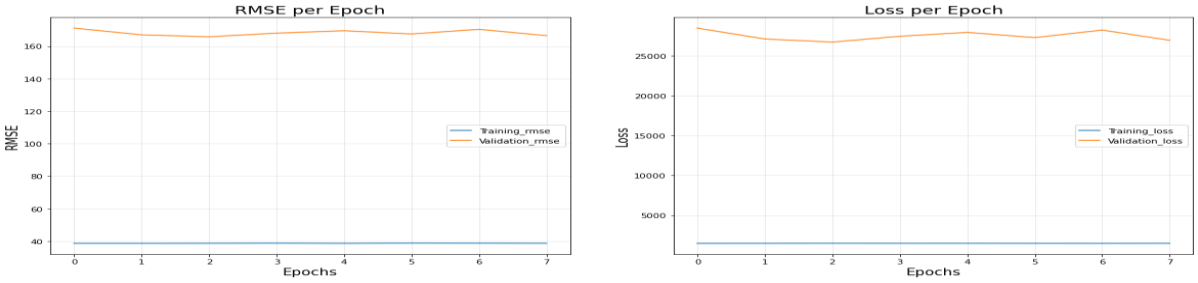
Epoch 8: Early stopping

RMSE for training set: 38.66558027267456

RMSE for Validation set: 168.18088912963867

Loss for Training set: 1497.2456970214844

Loss for Validation set: 27512.62255859375



Plot 4.6.1.2 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 2 for single-layer LSTM

Experiment 3:

The number of epochs for this experiment is: 10

The Batch Size is: 256

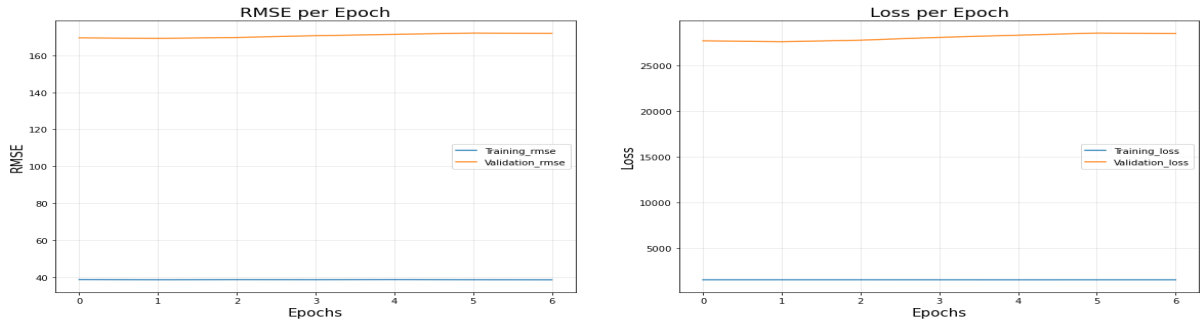
Epoch 7: Early stopping

RMSE for training set: 38.65906034197126

RMSE for Validation set: 170.5809587751116

Loss for Training set: 1492.5531703404017

Loss for Validation set: 28076.549386160714



Plot 4.6.1.3 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 3 for single-layer LSTM

Experiment 4:

The number of epochs for this experiment is: 20

The Batch Size is: 32

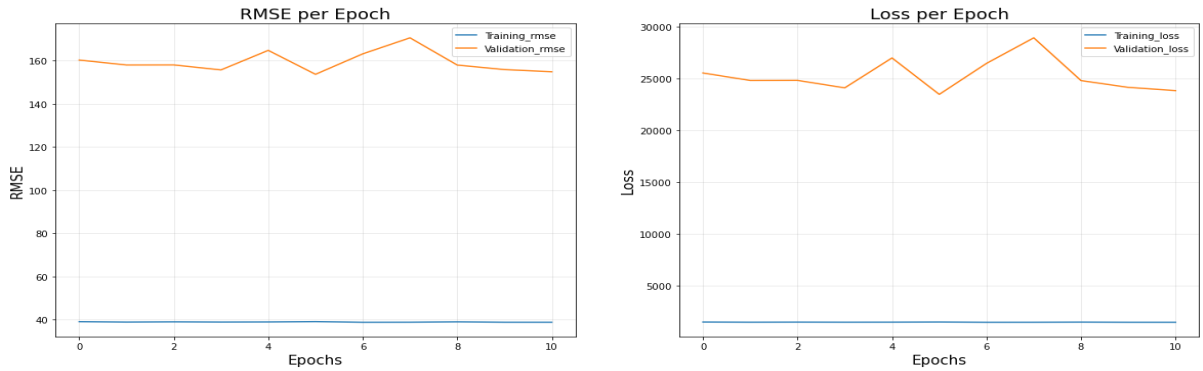
Epoch 11: Early stopping

RMSE for training set: 38.837096821178086

RMSE for Validation set: 159.39634565873578

Loss for Training set: 1508.3310768821023

Loss for Validation set: 25243.608487215908



Plot 4.6.1.4 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 4 for single-layer LSTM

Experiment 5:

The number of epochs for this experiment is: 20

The Batch Size is: 64

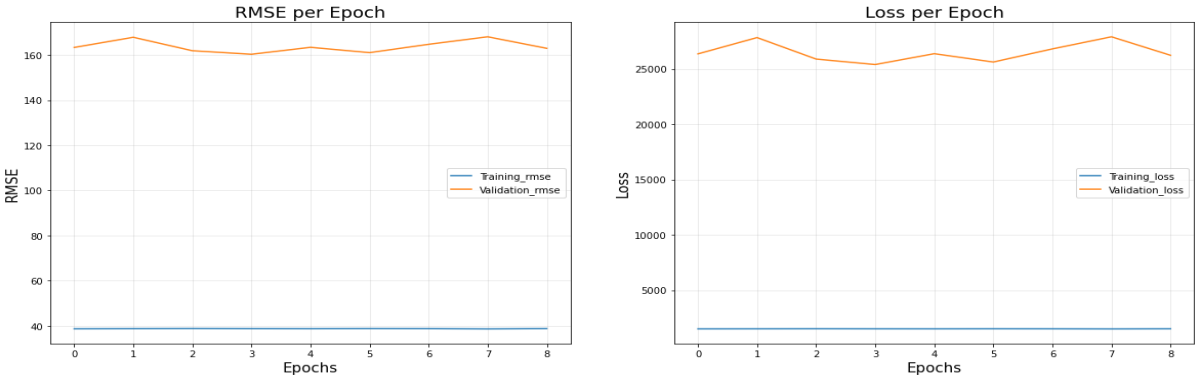
Epoch 9: Early stopping

RMSE for training set: 38.73125966389974

RMSE for Validation set: 163.71492343478732

Loss for Training set: 1500.1130913628472

Loss for Validation set: 26486.465711805555



Plot 4.6.1.5 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 5 for single-layer LSTM

Experiment 6:

The number of epochs for this experiment is: 20

The Batch Size is: 128

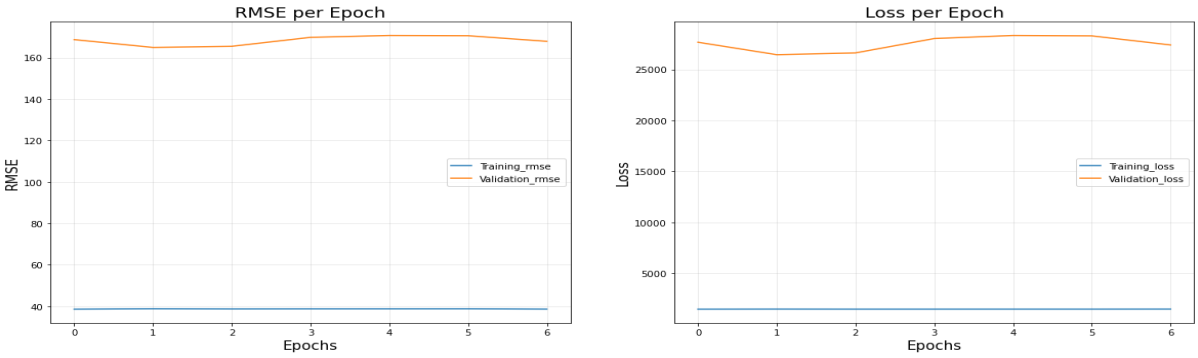
Epoch 7: Early stopping

RMSE for training set: 38.70092337472098

RMSE for Validation set: 168.2726069859096

Loss for Training set: 1496.541294642857

Loss for Validation set: 27543.599051339286



Plot 4.6.1.6 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 6 for single-layer LSTM

Experiment 7:

The number of epochs for this experiment is: 20

The Batch Size is: 256

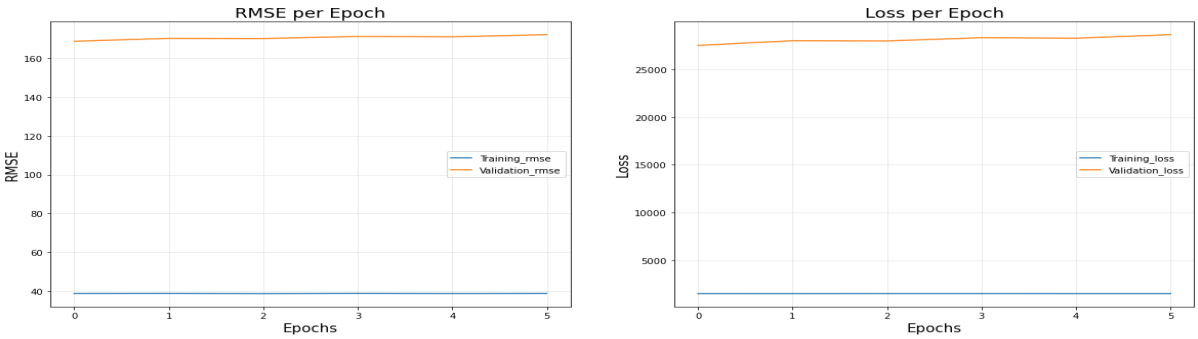
Epoch 6: Early stopping

RMSE for training set: 38.60349973042806

RMSE for Validation set: 170.709841410319

Loss for Training set: 1492.0640869140625

Loss for Validation set: 28118.8896484375



Plot 4.6.1.7 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 7 for single-layer LSTM

Experiment 8:

The number of epochs for this experiment is: 50

The Batch Size is: 32

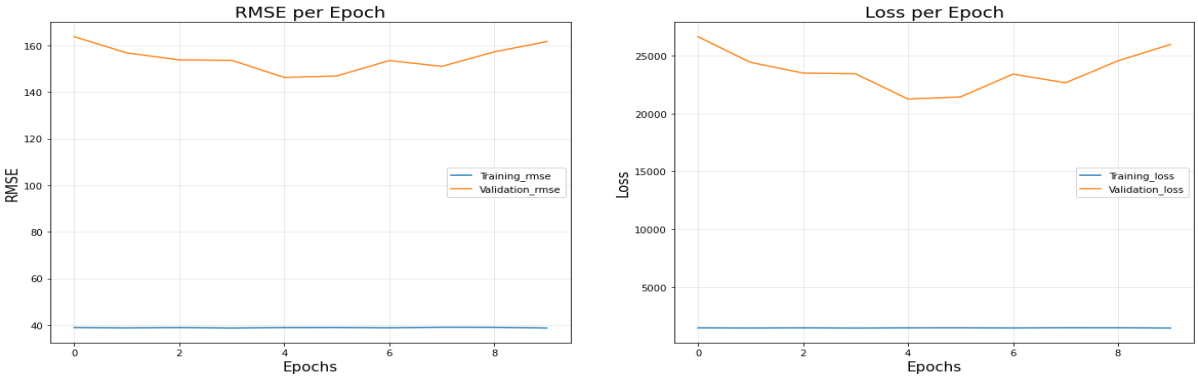
Epoch 10: Early stopping

RMSE for training set: 38.873711013793944

RMSE for Validation set: 154.4233413696289

Loss for Training set: 1511.1796020507813

Loss for Validation set: 23700.488671875



Plot 4.6.1.8 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 8 for single-layer LSTM

Experiment 9:

The number of epochs for this experiment is: 50

The Batch Size is: 64

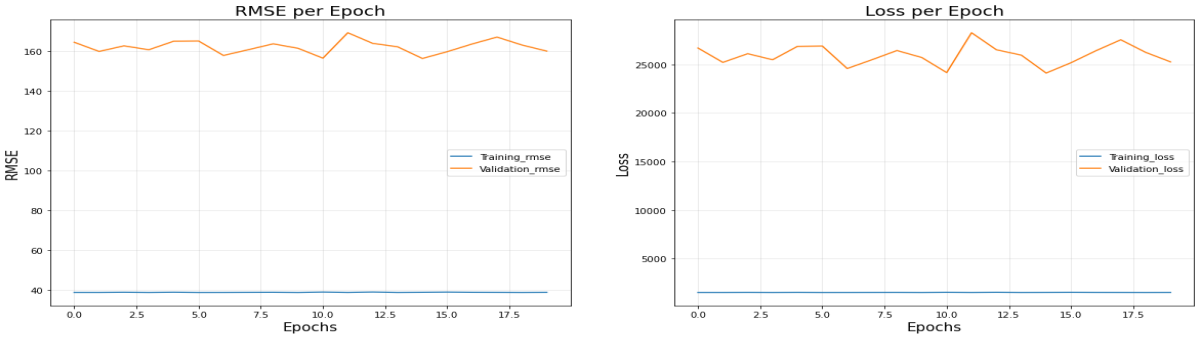
Epoch 19: Early stopping

RMSE for training set: 38.74873218536377

RMSE for Validation set: 162.06483306884766

Loss for Training set: 1501.4699401855469

Loss for Validation set: 25959.1302734375



Plot 4.6.1.9 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 9 for single-layer LSTM

Experiment 10:

The number of epochs for this experiment is: 50

The Batch Size is: 128

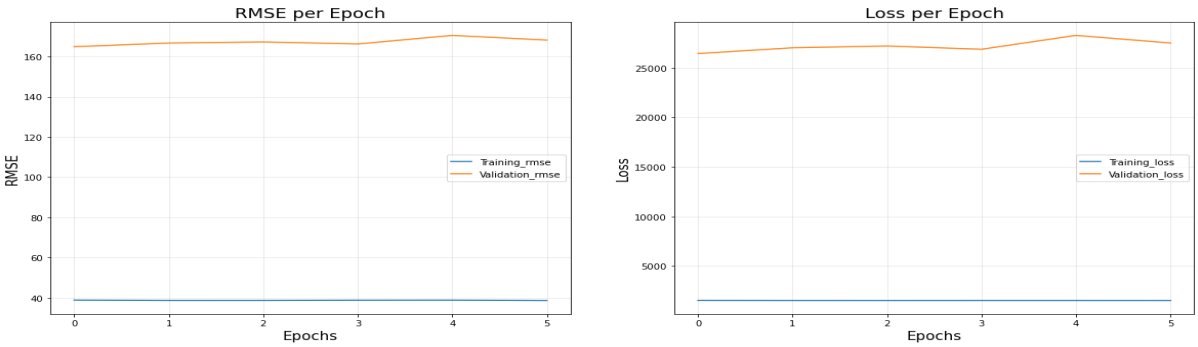
Epoch 6: Early stopping

RMSE for training set: 38.66478157043457

RMSE for Validation set: 167.28048451741537

Loss for Training set: 1495.0347086588542

Loss for Validation set: 27217.763346354168



Plot 4.6.1.10 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 10 for single-layer LSTM

Experiment 11:

The number of epochs for this experiment is: 50

The Batch Size is: 256

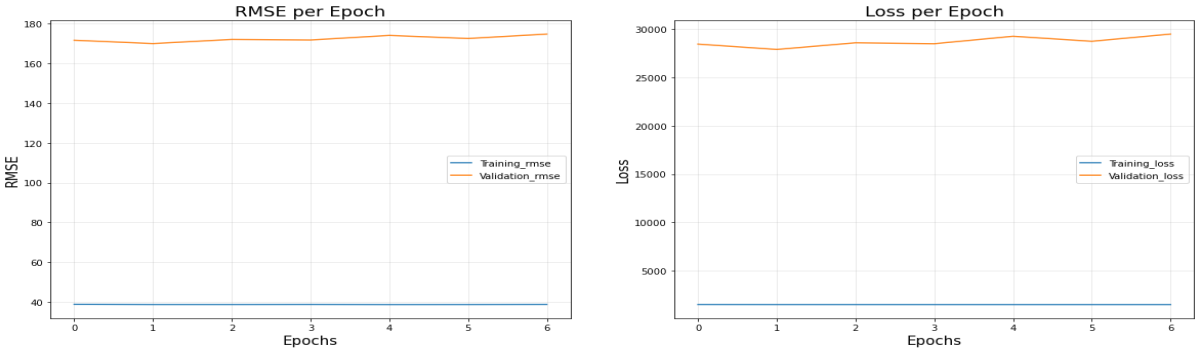
Epoch 7: Early stopping

RMSE for training set: 38.64458465576172

RMSE for Validation set: 172.45238603864397

Loss for Training set: 1491.5854317801338

Loss for Validation set: 28696.774832589286



Plot 4.6.1.11 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 11 for single-layer LSTM

Experiment 12:

The number of epoch for this experiment is: 100

The Batch Size is: 32

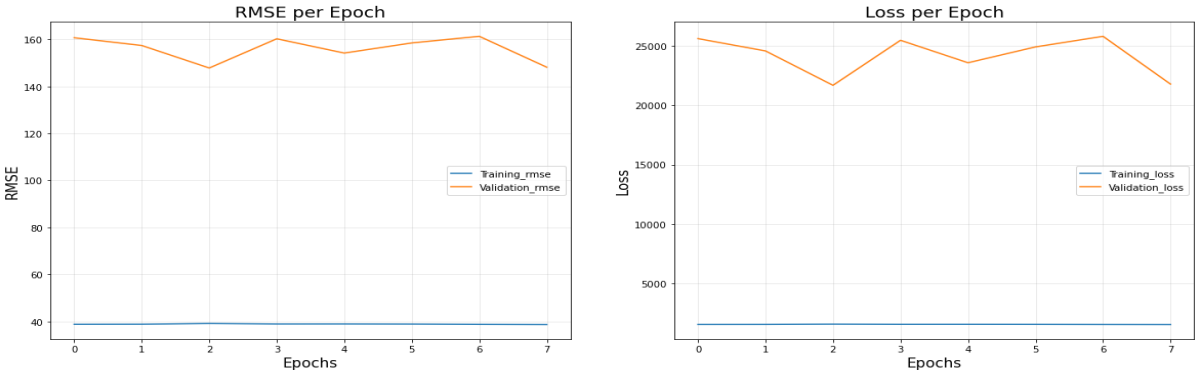
Epoch 8: Early stopping

RMSE for training set: 38.844096660614014

RMSE for Validation set: 156.0451889038086

Loss for Training set: 1508.8807067871094

Loss for Validation set: 24197.555908203125



Plot 4.6.1.12 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 12 for single-layer LSTM

Experiment 13:

The number of epochs for this experiment is: 100

The Batch Size is: 64

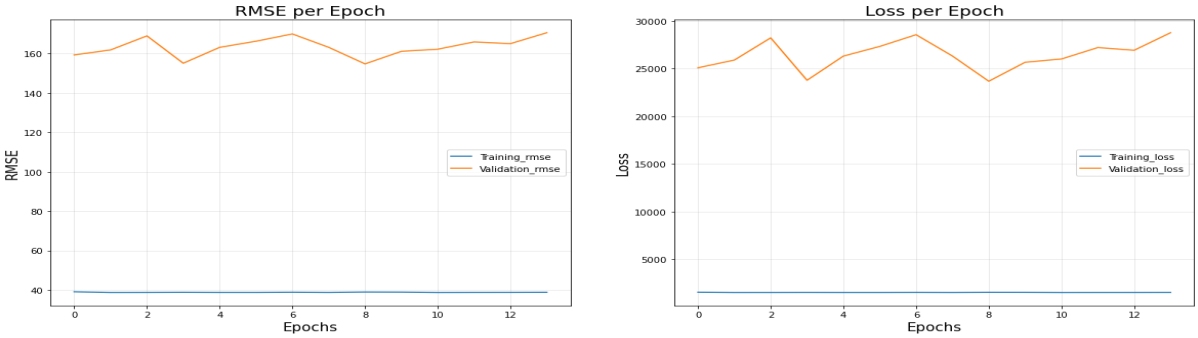
Epoch 14: Early stopping

RMSE for training set: 38.74761799403599

RMSE for Validation set: 163.42625100272042

Loss for Training set: 1501.3886893136162

Loss for Validation set: 26408.633649553572



Plot 4.6.1.13 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 13 for single-layer LSTM

Experiment 14:

The number of epochs for this experiment is: 100

The Batch Size is: 128

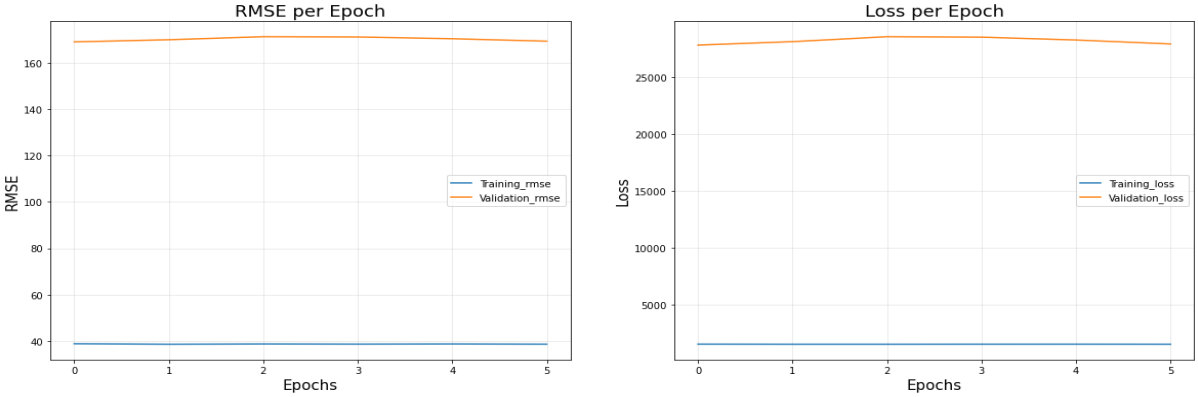
Epoch 6: Early stopping

RMSE for training set: 38.69375737508138

RMSE for Validation set: 170.19012705485025

Loss for Training set: 1498.5138549804688

Loss for Validation set: 28171.270833333332



Plot 4.6.1.14 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 14 for single-layer LSTM

Experiment 15:

The number of epochs for this experiment is: 100

The Batch Size is: 256

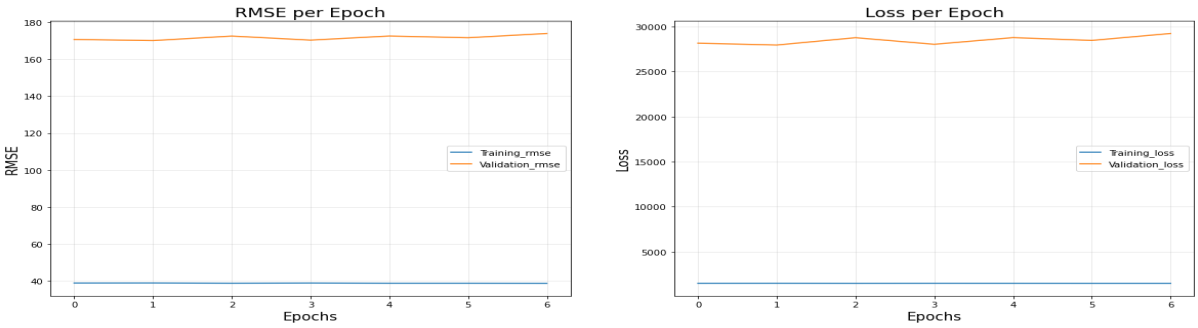
Epoch 7: Early stopping

RMSE for training set: 38.662745884486604

RMSE for Validation set: 171.75885445731026

Loss for Training set: 1493.642054966518

Loss for Validation set: 28466.053292410714



Plot 4.6.1.15 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 15 for single-layer LSTM

4.6.2 MULTI-LAYER LSTM EXPERIMENTS

Experiment 1.

The number of epochs for this experiment is: 10

The Batch Size is: 64

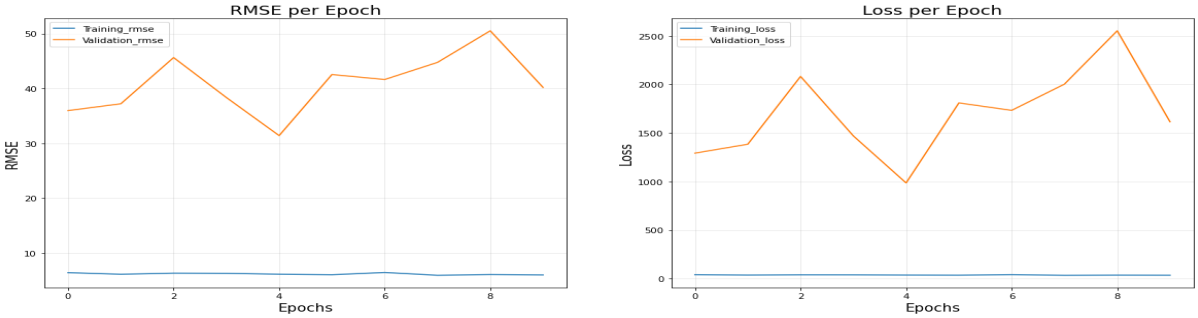
Epoch 10: Early stopping

RMSE for training set: 6.143621349334717

RMSE for Validation set: 40.808166885375975

Loss for Training set: 37.772695541381836

Loss for Validation set: 1692.154034423828



Plot 4.6.2.1 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 1 for multi-layer LSTM

Experiment 2:

The number of epochs for this experiment is: 10

The Batch Size is: 128

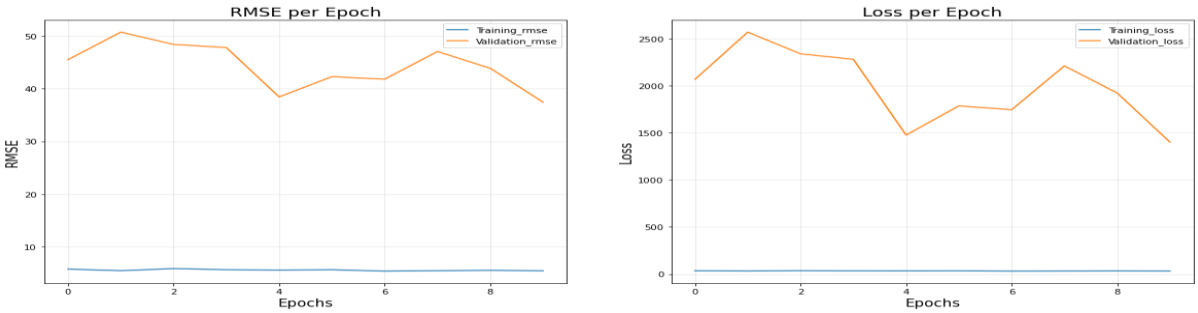
Epoch 10: Early stopping

RMSE for training set: 5.626106452941895

RMSE for Validation set: 44.275393676757815

Loss for Training set: 31.674564361572266

Loss for Validation set: 1977.3612426757813



Plot 4.6.2.2 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 2 for multi-layer LSTM

Experiment 3:

The number of epochs for this experiment is: 10

The Batch Size is: 256

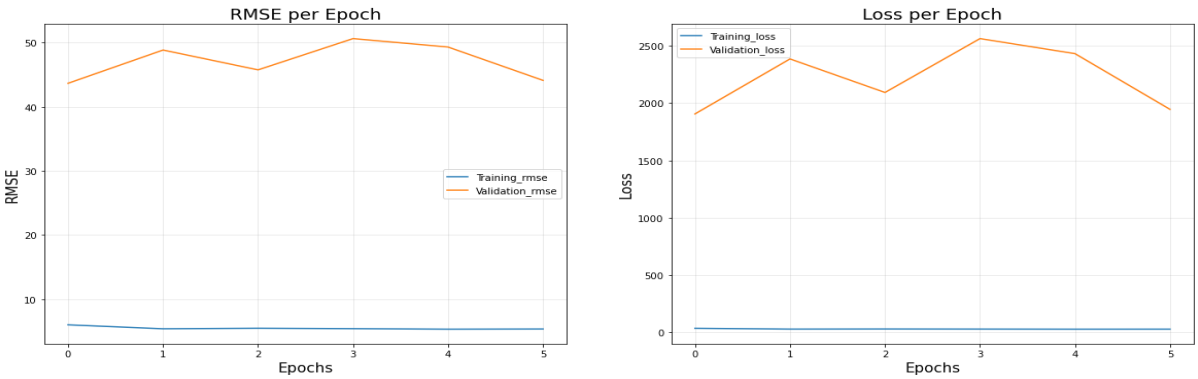
Epoch 6: Early stopping

RMSE for training set: 5.497672080993652

RMSE for Validation set: 47.050201416015625

Loss for Training set: 30.279301325480144

Loss for Validation set: 2220.8766276041665



Plot 4.6.2.3 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 3 for multi-layer LSTM

Experiment 4:

The number of epochs for this experiment is: 20

The Batch Size is: 32

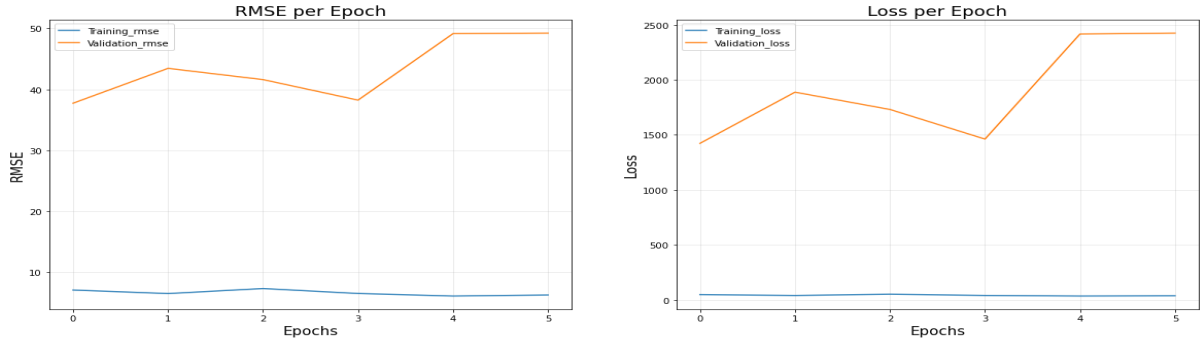
Epoch 6: Early stopping

RMSE for training set: 6.613148291905721

RMSE for Validation set: 43.23515764872233

Loss for Training set: 43.9222838083903

Loss for Validation set: 1890.7832845052083



Plot 4.6.2.4 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 4 for multi-layer LSTM

Experiment 5:

The number of epochs for this experiment is: 20

The Batch Size is: 64

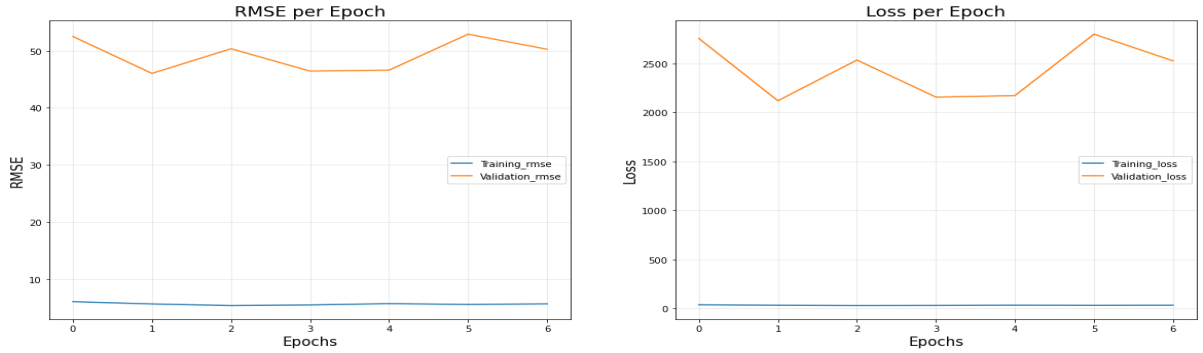
Epoch 7: Early stopping

RMSE for training set: 5.6148907116481235

RMSE for Validation set: 49.308297293526785

Loss for Training set: 31.566520963396346

Loss for Validation set: 2438.643798828125



Plot 4.6.2.5 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 5 for multi-layer LSTM

Experiment 6:

The number of epochs for this experiment is: 20

The Batch Size is: 128

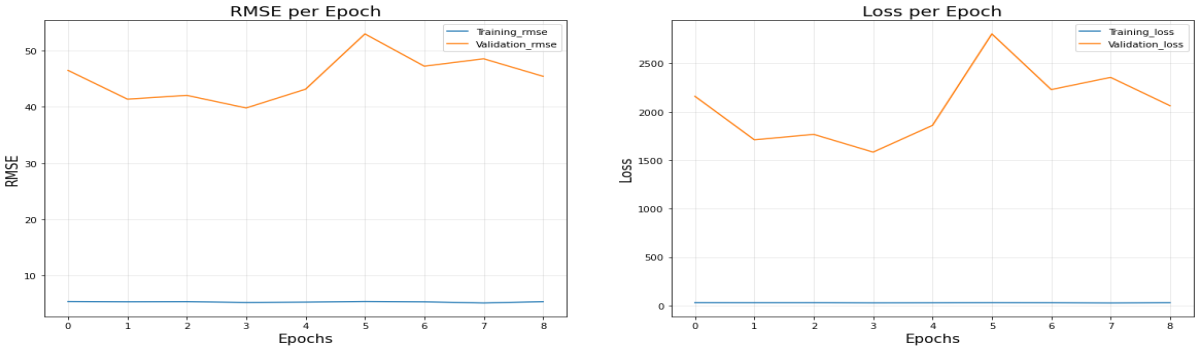
Epoch 9: Early stopping

RMSE for training set: 5.34142271677653

RMSE validation set: 45.21190304226346

Loss for Training set: 28.537779066297745

Loss for Validation set: 2059.1418185763887



Plot 4.6.2.6 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 6 for multi-layer LSTM

Experiment 7:

The number of epochs for this experiment is: 20

The Batch Size is: 256

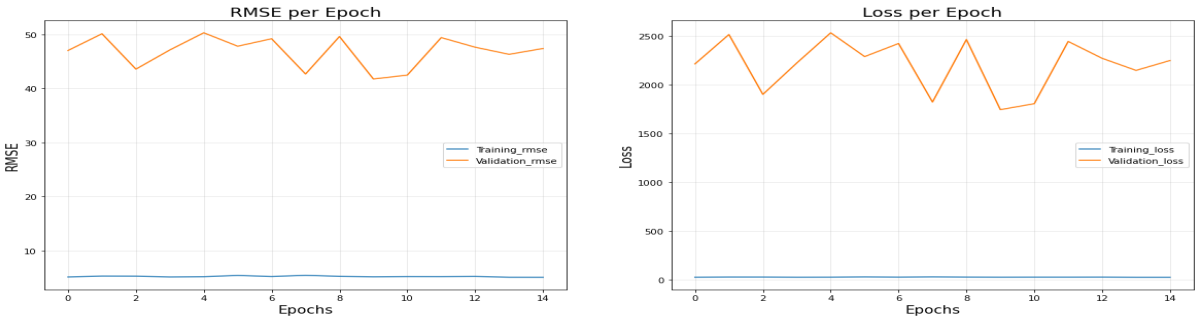
Epoch 15: Early stopping

RMSE for training set: 5.186712106068929

RMSE validation set: 46.848853810628256

Loss for Training set: 26.912051264444987

Loss for Validation set: 2202.6906901041666



Plot 4.6.2.7 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 7 for multi-layer LSTM

Experiment 8:

The number of epochs for this experiment is: 50

The Batch Size is: 32

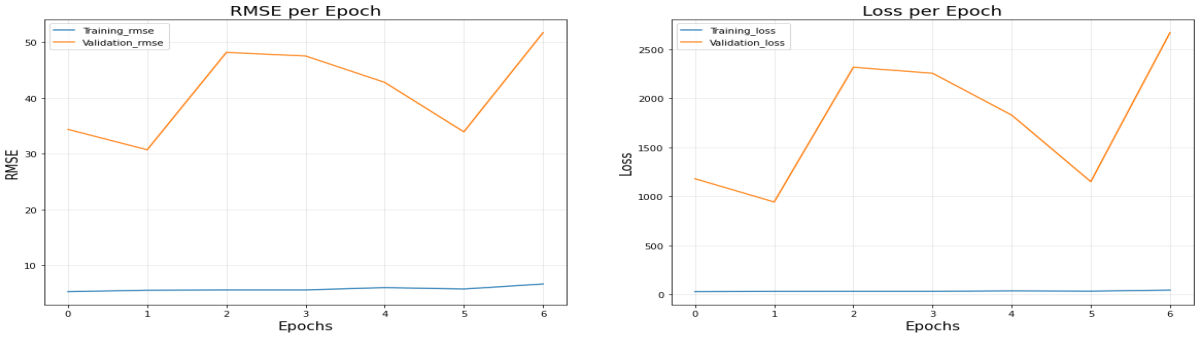
Epoch 7: Early stopping

RMSE for training set: 5.803990636553083

RMSE for Validation set: 41.289608819144114

Loss for Training set: 33.85734122140067

Loss for Validation set: 1763.3665161132812



Plot 4.6.2.8 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 8 for multi-layer LSTM

Experiment 9:

The number of epochs for this experiment is: 50

The Batch Size is: 64

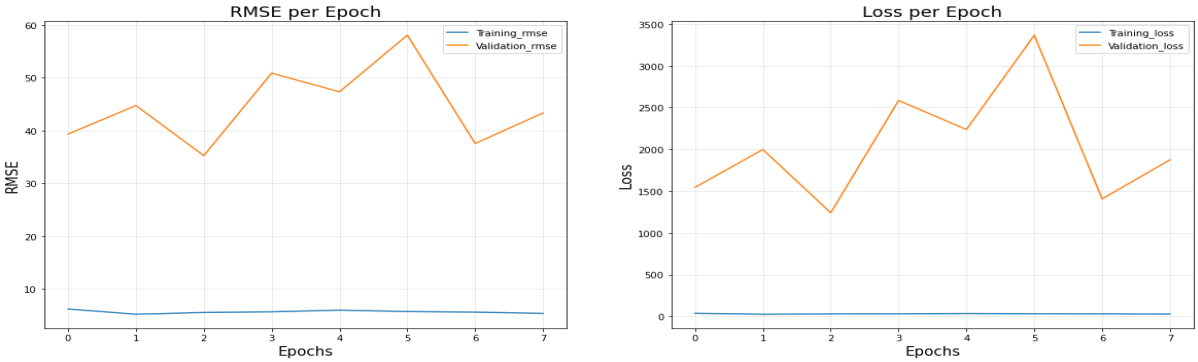
Epoch 7: Early stopping

RMSE for training set: 5.6699138879776

RMSE for Validation set: 44.529083251953125

Loss for Training set: 32.23506832122803

Loss for Validation set: 2032.140396118164



Plot 4.6.2.9 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 9 for multi-layer LSTM

Experiment 10:

The number of epochs for this experiment is: 50

The Batch Size is: 128

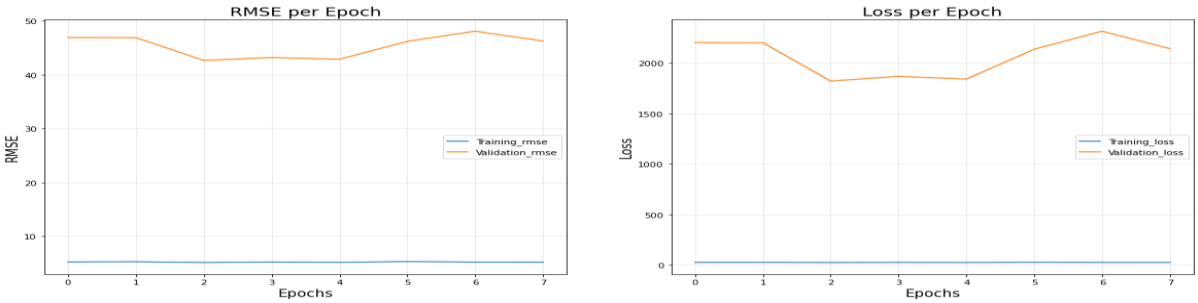
Epoch 8: Early stopping

RMSE for training set: 5.22543740272522

RMSE for Validation set: 45.40724325180054

Loss for Training set: 27.30871558189392

Loss for Validation set: 2065.7951049804688



Plot 4.6.2.10 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 10 for multi-layer LSTM

Experiment 11:

The number of epochs for this experiment is: 50

The Batch Size is: 256

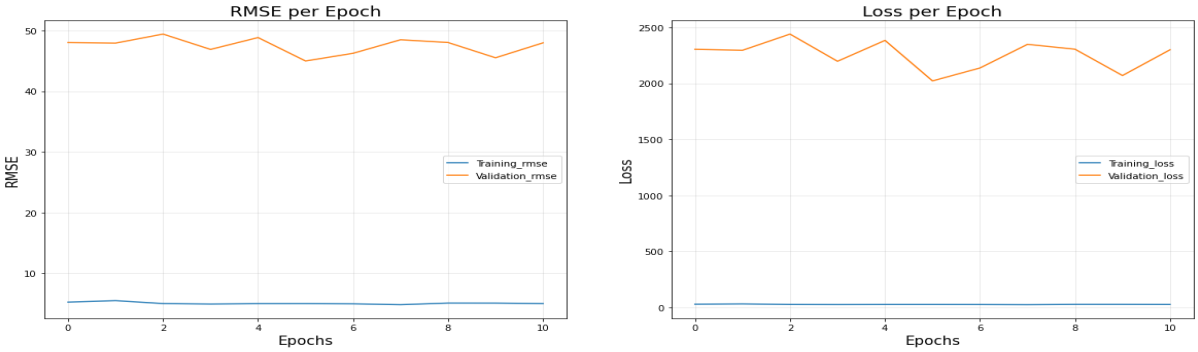
Epoch 11: Early stopping

RMSE for training set: 5.078565554185347

RMSE for Validation set: 47.46565732088956

Loss for Training set: 25.820174130526457

Loss for Validation set: 2254.7696533203125



Plot 4.6.2.11 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 11 for multi-layer LSTM

Experiment 12:

The number of epochs for this experiment is: 100

The Batch Size is: 32

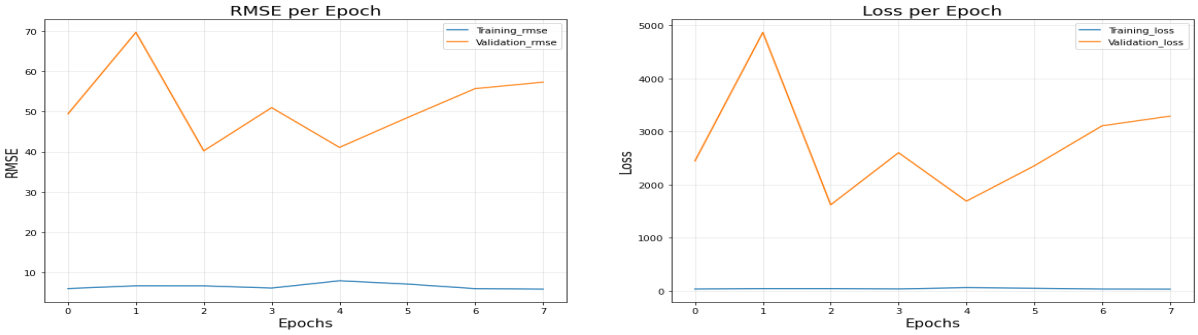
Epoch 8: Early stopping

RMSE for training set: 6.490382134914398

RMSE for Validation set: 51.64999294281006

Loss for Training set: 42.56632328033447

Loss for Validation set: 2746.9314575195312



Plot 4.6.2.12 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 12 for multi-layer LSTM

Experiment 13:

The number of epochs for this experiment is: 100

The Batch Size is: 64

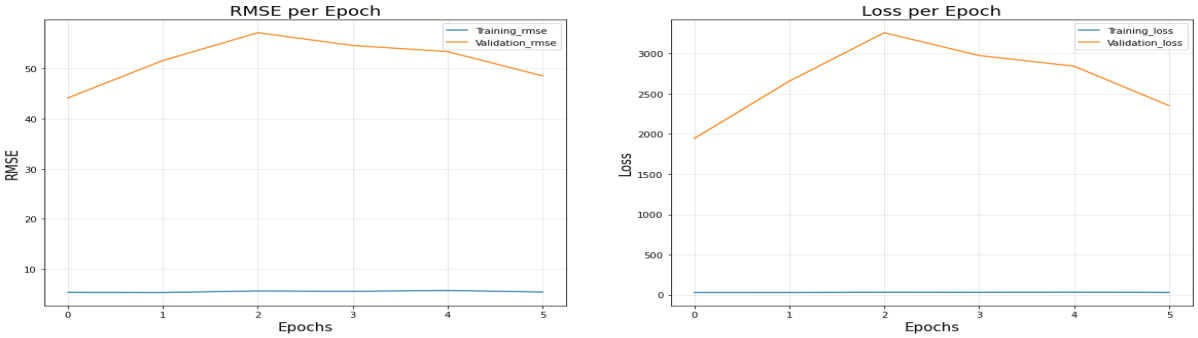
Epoch 6: Early stopping

RMSE for training set: 5.509049733479817

RMSE for Validation set: 51.531575520833336

Loss for Training set: 30.373904546101887

Loss for Validation set: 2673.4107666015625



Plot 4.6.2.13 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 13 for multi-layer LSTM

Experiment 14:

The number of epochs for this experiment is: 100

The Batch Size is: 128

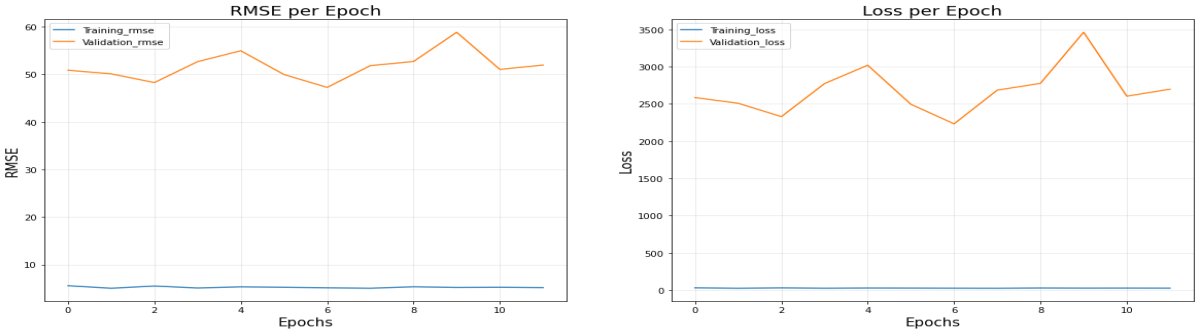
Epoch 11: Early stopping

RMSE for training set: 5.227704564730327

RMSE for Validation set: 51.69479624430338

Loss for Training set: 27.354541619618733

Loss for Validation set: 2680.8528645833335



Plot 4.6.2.14 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 14 for multi-layer LSTM

Experiment 15:

The number of epochs for this experiment is: 100

The Batch Size is: 256

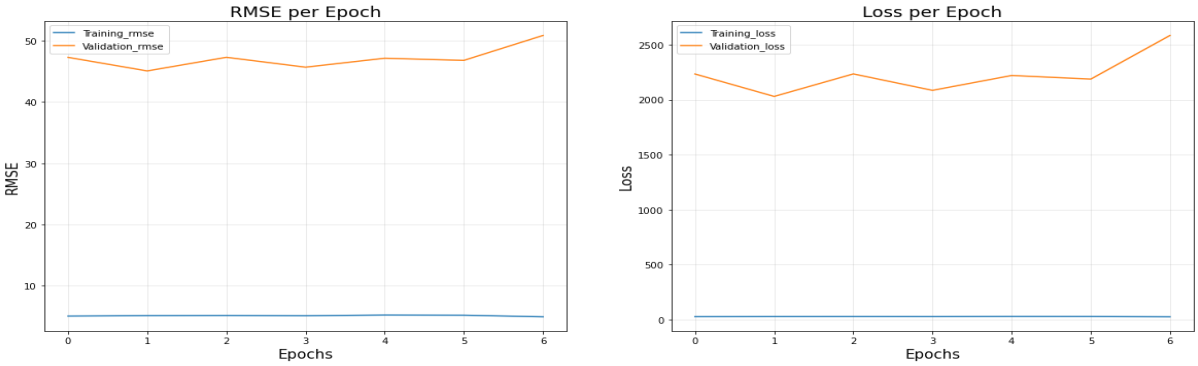
Epoch 7: Early stopping

RMSE for training set: 5.113396031515939

RMSE for Validation set: 47.136368887765066

Loss for Training set: 26.154745646885463

Loss for Validation set: 2224.75137765067



Plot 4.6.2.15 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 15 for multi-layer LSTM

4.6.3 BI-DIRECTIONAL LSTM EXPERIMENTS

Experiment 1:

The number of epochs for this experiment is: 10

The Batch Size is: 64

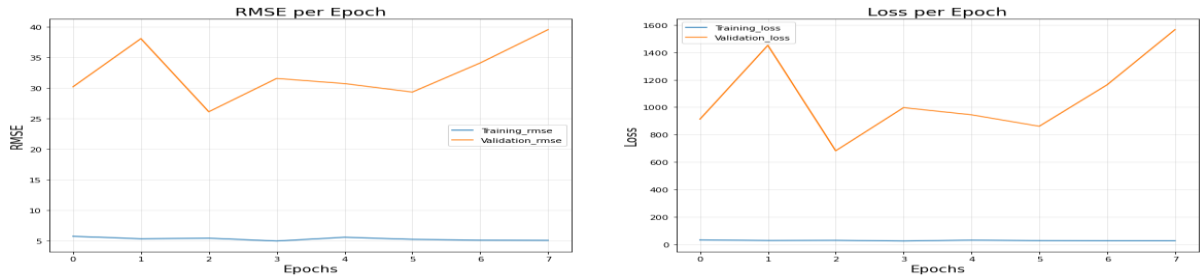
Epoch 8: early stopping

RMSE for training set: 5.300550818443298

RMSE for Validation set: 32.46793222427368

Loss for Training set: 28.158562898635864

Loss for Validation set: 1072.1210403442383



Plot 4.6.3.1 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 1 for Bi-directional LSTM

Experiment 2:

The number of epochs for this experiment is: 10

The Batch Size is: 128

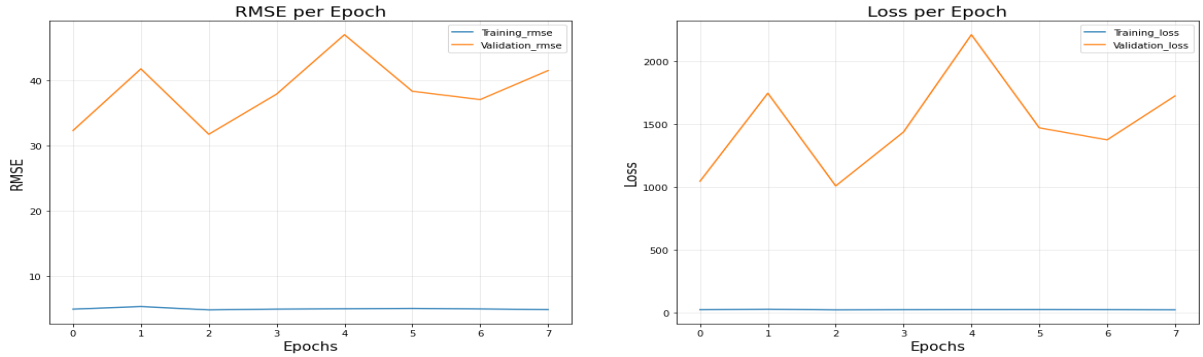
Epoch 8: Early stopping

RMSE for training set: 5.019588053226471

RMSE for Validation set: 38.481415033340454

Loss for Training set: 25.21729564666748

Loss for Validation set: 1503.119773864746



Plot 4.6.3.2 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 2 for Bi-directional LSTM

Experiment 3:

The number of epochs for this experiment is: 10

The Batch Size is: 256

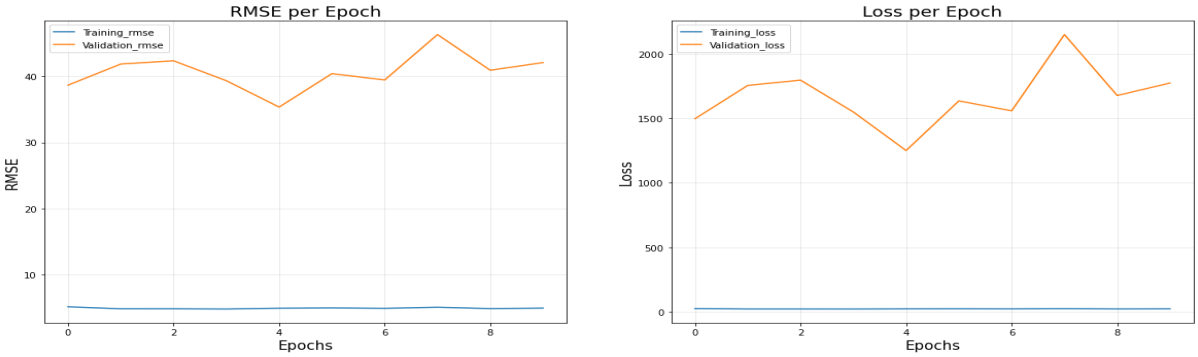
Epoch 10: Early stopping

RMSE for training set: 4.870334100723267

RMSE for Validation set: 40.699750900268555

Loss for Training set: 23.73059196472168

Loss for Validation set: 1663.889697265625



Plot 4.6.3.3 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 3 for Bi-directional LSTM

Experiment 4:

The number of epochs for this experiment is: 20

The Batch Size is: 32

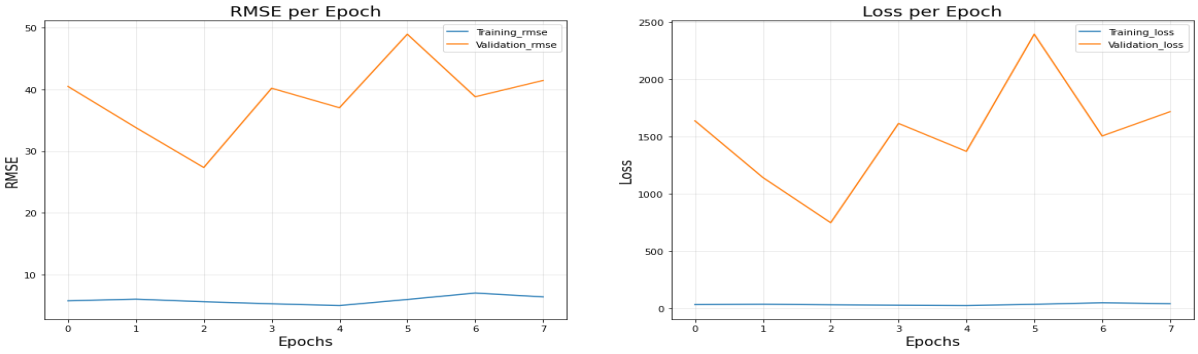
Epoch 8: Early stopping

RMSE for training set: 5.896820604801178

RMSE for Validation set: 38.47724676132202

Loss for Training set: 35.126137495040894

Loss for Validation set: 1514.5411911010742



Plot 4.6.3.4 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 4 for Bi-directional LSTM

Experiment 5:

The number of epochs for this experiment is: 20

The Batch Size is: 64

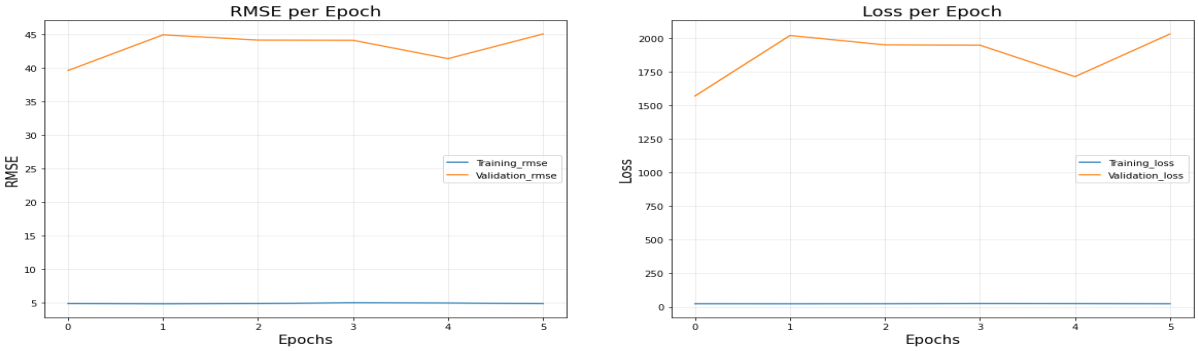
Epoch 6: Early stopping

RMSE for training set: 4.903278350830078

RMSE for Validation set: 43.21588643391927

Loss for Training set: 24.045400619506836

Loss for Validation set: 1871.67236328125



Plot 4.6.3.5 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 5 for Bi-directional LSTM

Experiment 6:

The number of epochs for this experiment is: 20

The Batch Size is: 128

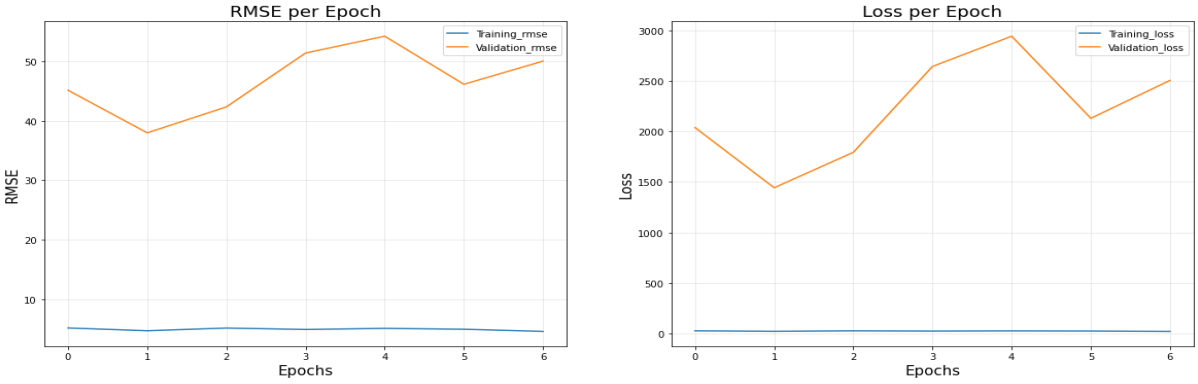
Epoch 7: Early stopping

RMSE for training set: 4.916427952902658

RMSE for Validation set: 46.76645769391741

Loss for Training set: 24.21438080923898

Loss for Validation set: 2213.9742431640625



Plot 4.6.3.6 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 6 for Bi-directional LSTM

Experiment 7:

The number of epochs for this experiment is: 20

The Batch Size is: 256

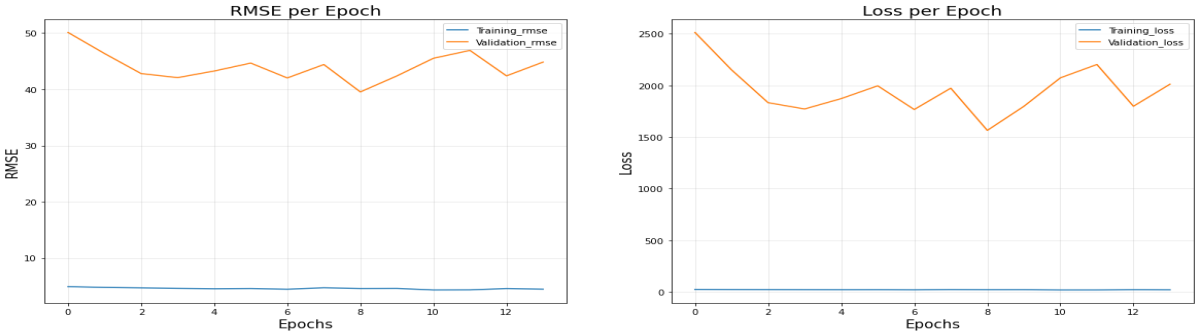
Epoch 14: Early stopping

RMSE for training set: 4.5681549821581156

RMSE for Validation set: 44.09663609095982

Loss for Training set: 20.892179897853307

Loss for Validation set: 1950.9236363002233



Plot 4.6.3.7 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 7 for Bi-directional LSTM

Experiment 8:

The number of epochs for this experiment is: 50

The Batch Size is: 32

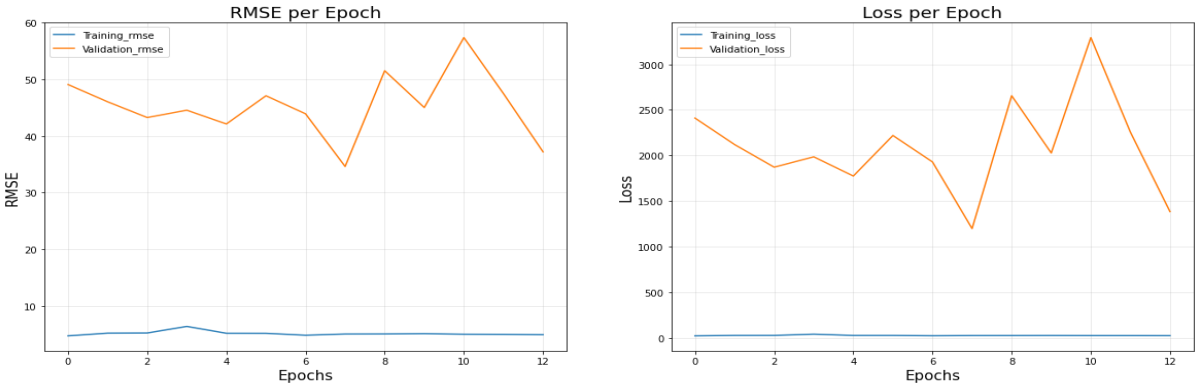
Epoch 13: Early stopping

RMSE for training set: 5.128849359659048

RMSE for Validation set: 45.3367438683143

Loss for Training set: 26.454325162447414

Loss for Validation set: 2086.5198692908652



Plot 4.6.3.8 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 8 for Bi-directional LSTM

Experiment 9:

The number of epochs for this experiment is: 50

The Batch Size is: 64

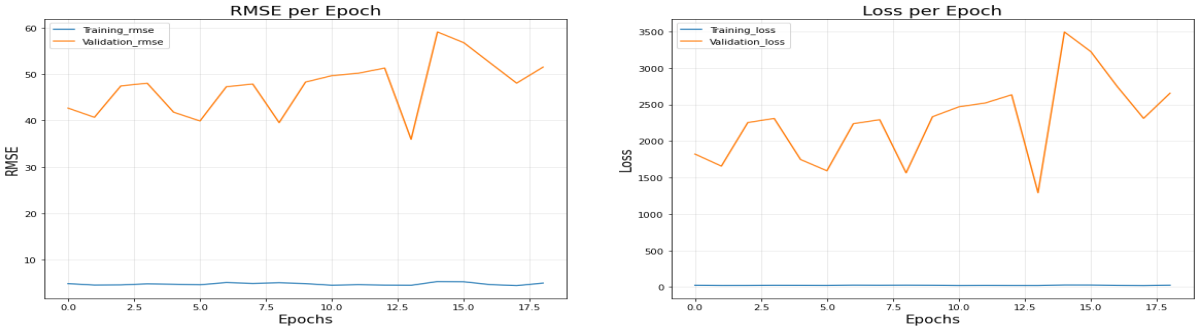
Epoch 19: Early stopping

RMSE for training set: 4.73792196574964

RMSE for Validation set: 47.30170842220909

Loss for Training set: 22.51254513389186

Loss for Validation set: 2271.4289807771383



Plot 4.6.3.9 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 9 for Bi-directional LSTM

Experiment 10:

The number of epochs for this experiment is: 50

The Batch Size is: 128

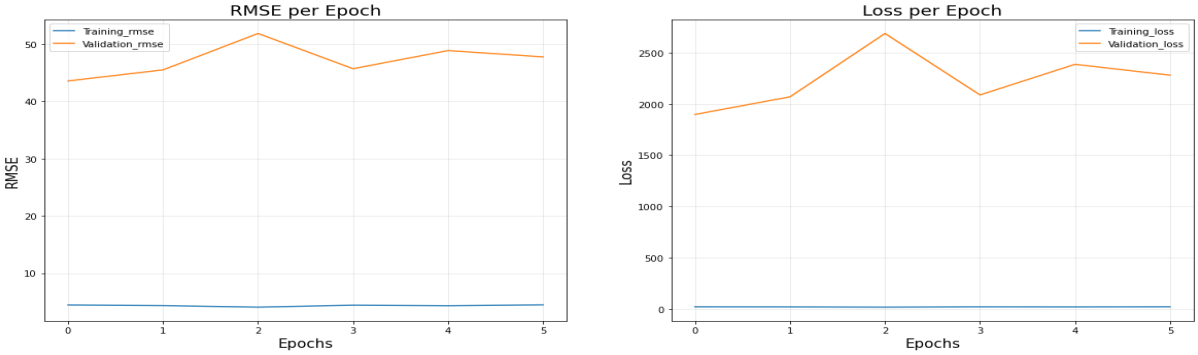
Epoch 6: Early stopping

RMSE for training set: 4.341687758763631

RMSE for Validation set: 47.20260429382324

Loss for Training set: 18.868017832438152

Loss for Validation set: 2235.2625935872397



Plot 4.6.3.10 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 10 for Bi-directional LSTM

Experiment 11:

The number of epochs for this experiment is: 50

The Batch Size is: 256

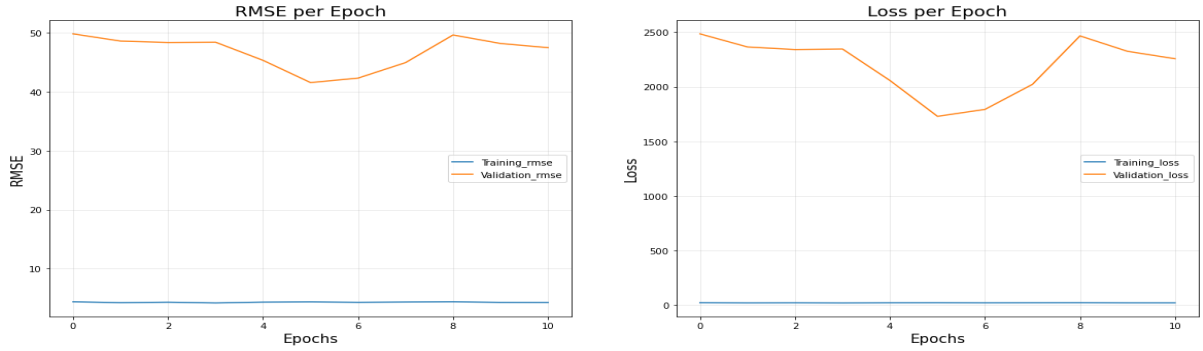
Epoch 11: Early stopping

RMSE for training set: 4.270642410625111

RMSE for Validation set: 46.812339782714844

Loss for Training set: 18.242123517123137

Loss for Validation set: 2198.7756902521305



Plot 4.6.3.11 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 11 for Bi-directional LSTM

Experiment 12:

The number of epochs for this experiment is: 100

The Batch Size is: 32

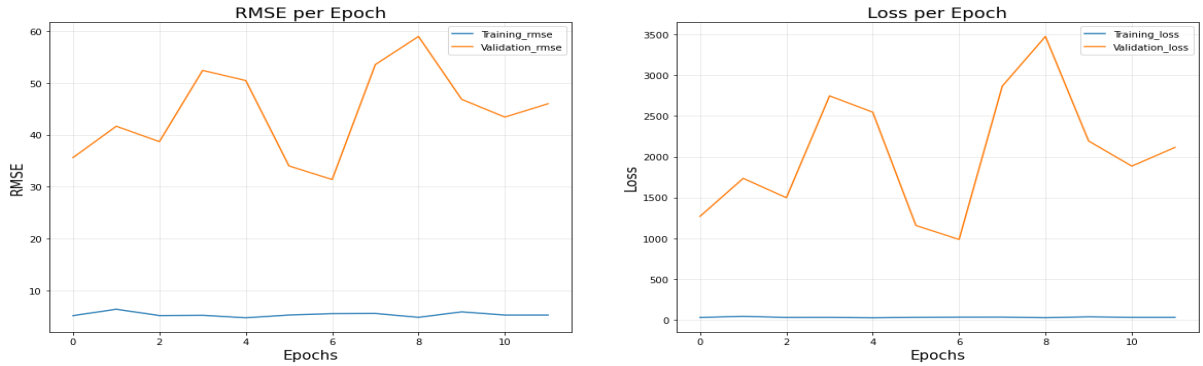
Epoch 12: Early stopping

RMSE for training set: 5.331169207890828

RMSE for Validation set: 44.41632763544718

Loss for Training set: 28.602105617523193

Loss for Validation set: 2039.6321970621746



Plot 4.6.3.12 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 12 for Bi-directional LSTM

Experiment 13:

The number of epochs for this experiment is: 100

The Batch Size is: 64

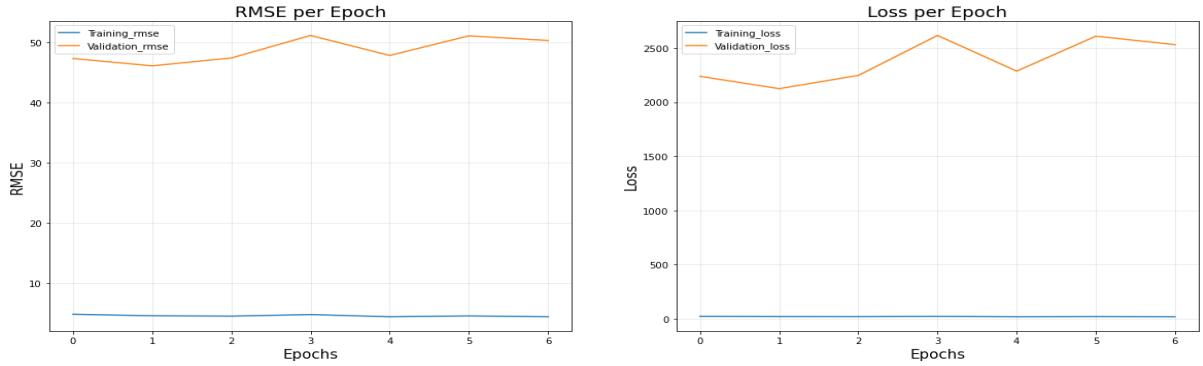
Epoch 7: Early stopping

RMSE for training set: 4.56824254989624

RMSE for Validation set: 48.7288567679269

Loss for Training set: 20.892899649483816

Loss for Validation set: 2378.127615792411



Plot 4.6.3.13 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 11 for Bi-directional LSTM

Experiment 14:

The number of epochs for this experiment is: 100

The Batch Size is: 128

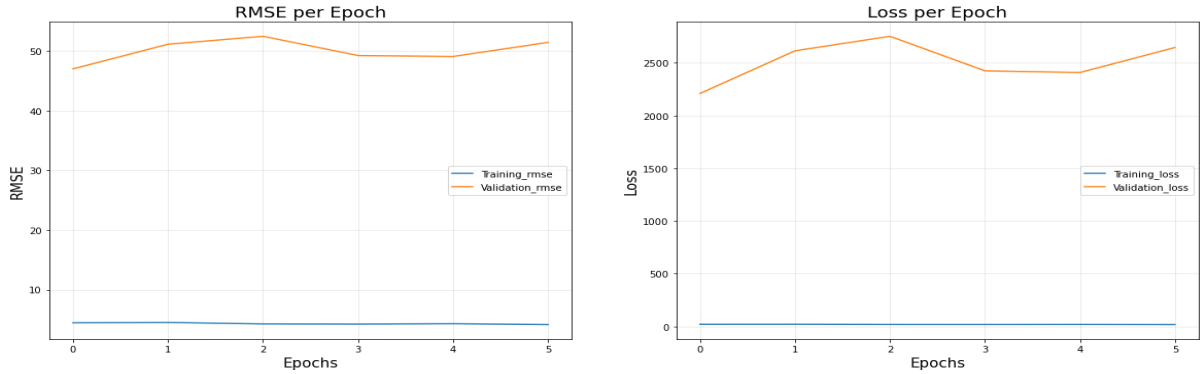
Epoch 6: Early stopping

RMSE for training set: 4.300731499989827

RMSE for Validation set: 50.05041313171387

Loss for Training set: 18.5114844640096

Loss for Validation set: 2508.3330891927085



Plot 4.6.3.14 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 14 for Bi-directional LSTM

Experiment 15:

The number of epochs for this experiment is: 100

The Batch Size is: 256

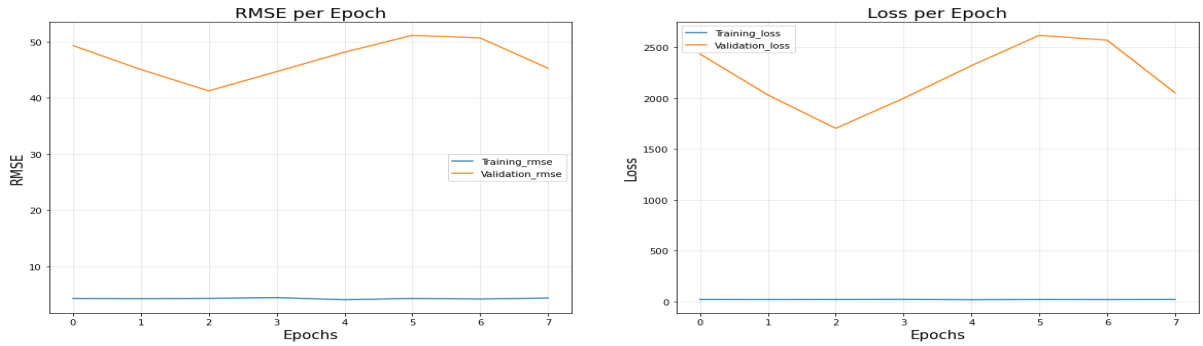
Epoch 8: Early stopping

RMSE for training set: 4.263089299201965

RMSE for Validation set: 46.95091676712036

Loss for Training set: 18.185882329940796

Loss for Validation set: 2214.7040405273438



Plot 4.6.3.15 Comparative plot for RMSE per epoch and Loss per epoch for training and validation sets for Experiment 15 for Bi-directional LSTM

45 experiments were conducted on three baseline models, 15 experiments each on a particular baseline model with the possible number of combinations for the number of epochs[10,20,50,100] and batch size[32,64,128,256] The best models have been saved from the experiment for three different models based on validation Loss. Section 4.7 will compare the performance of the 3 baseline models and 3 selected experiment models based on validation loss scores.

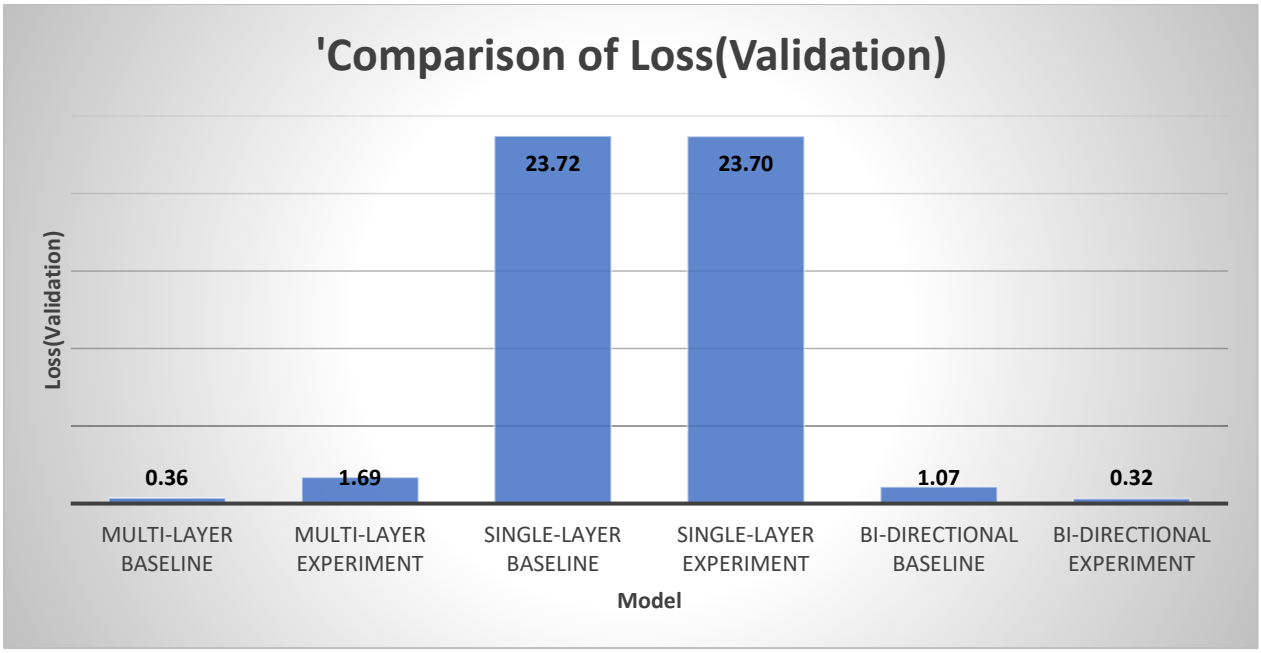
4.7 COMPARING THE PERFORMANCE OF THE BASELINE AND SELECTED EXPERIMENTAL MODEL

In the baseline model, a Multi-layer LSTM with a Loss(Validation data) score of 358.20 has outperformed a bi-directional LSTM with a Loss(Validation data) score of 1072.12 and single-layer LSTM Loss(Validation data) score of 23721.66 (refer to Table 4.7.1). In the experiment model, bi-directional LSTM with a Loss(Validation data) score of 322.02 has outperformed Multi-layer LSTM with a Loss (Validation data) score of 1692.15 and single-layer LSTM Loss(Validation data) score of 23700.49 (refer to Table 4.7.1).

Table 4.7.1 Comparative analysis of metrics of Baseline and Experimental Model

Model	**RMSE (Train)**	**RMSE (Validation)**	**Loss (Train)**	**Loss (Validation)**	**Number of epoch**	**batch size**
Multi-layer baseline	9.36	18.92	104.04	358.20	10	32
Multi-layer experiment	6.14	40.81	37.77	1692.15	10	64
Single-layer baseline	40.88	154.45	1681.00	23721.66	10	32
Single-layer experiment	38.87	154.42	1511.18	23700.49	50	32
Bi-directional baseline	8.95	32.47	96.94	1072.12	10	32
Bi-directional experiment	5.30	17.94	28.16	322.02	10	64

An experimental Bi-directional model with epochs:10 and batch size: 64 was selected for hyperparameter tunning because it has the lowest Loss score for the Validation data set of 322.02 among all models(refer to Plot 4.7.1)(refer to Table 4.7.1)



Plot 4.7.1 Comparison of Loss for validation data set for all models

4.8 HYPERPARAMETER TUNING

Hyperparameter tuning was conducted on the selected experimental bi-directional model by a Keras tuner with three experimental trials. Keras tuner utilised a random search algorithm by tracking the model for the lowest loss.

The range of Hyperparameter detected for tunning are as follows:

- Number of Layers: 2 to 20.
- Drop out: 0 to 0.5 with step size =0.1.
- RNN unit in each layer:16 to 128 with step size =16.
- The learning rate for Adam optimizer: 1e-4 to 1e-2 with sampling as log.

Trail 1 proposed 12 layers, a dropout rate of 0.4 and a learning rate with the lowest loss score of 101.51 compared to Trail 2 and Trial 3 with Loss scores of 127.69 and 4445.03 respectively. Hyperparameter values for trail 1 summary were selected after tunning and the model was saved for further analysis. Adam optimizer, was selected based on understanding from the literature review.

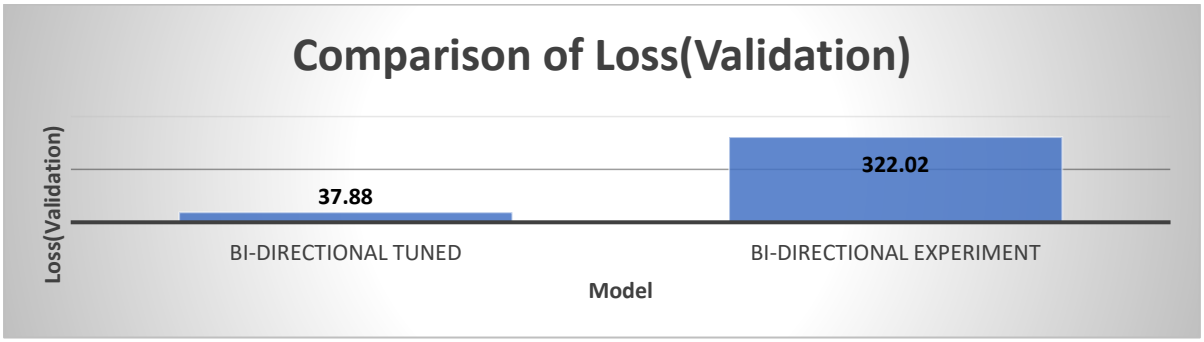
Table 4.8.1 Hyperparameter Tuning's Trail Summary

Trial 1 summary	**Trial 2 summary**	**Trial 3 summary**
Hyperparameters:	Hyperparameters:	Hyperparameters:
num_layers: 12	num_layers: 13	num_layers: 17
input_unit0: 80	input_unit0: 48	input_unit0: 16
input_unit1: 32	input_unit1: 64	input_unit1: 128
Dropout_rate: 0.4	Dropout_rate: 0.4	Dropout_rate: 0.3
lr: 0.00134	lr: 0.0044	lr: 0.0099
input_unit2: 16	input_unit2: 64	input_unit2: 80
input_unit3: 16	input_unit3: 64	input_unit3: 112
input_unit4: 16	input_unit4: 32	input_unit4: 48
input_unit5: 16	input_unit5: 48	input_unit5: 32
input_unit6: 16	input_unit6: 80	input_unit6: 64
input_unit7: 16	input_unit7: 32	input_unit7: 16
input_unit8: 16	input_unit8: 80	input_unit8: 96

input_unit9: 16	input_unit9: 80	input_unit9: 16
input_unit10: 16	input_unit10: 96	input_unit10: 128
input_unit11: 16	input_unit11: 32	input_unit11: 32
Val Loss: 101.51	input_unit12: 16	input_unit12: 80
	Val Loss: 127.69	input_unit13: 16
		input_unit14: 16
		input_unit15: 16
		input_unit16: 16
		Val Loss: 4445.03

4.9 COMPARING TUNED MODEL AND SELECTED EXPERIMENTAL MODEL

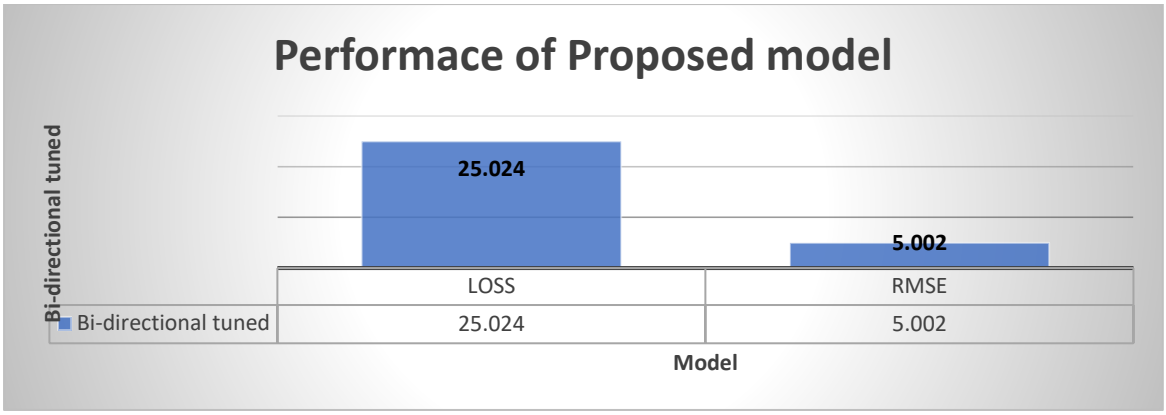
Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was implemented on the test data set



Plot 4.9.1 Compare the Bi-directional tuned model and Bi-directional Experimental model W.r.t Loss on the validation data set.

4.10 IMPLEMENTING THE PROPOSED MODEL ON THE UNSEEN/TEST DATA

The proposed model was implemented on the unseen test data set containing Yes Bank's share price for the period from 03-01-2020 to 04-08-2023. The data was transformed to stationary data by applying log transformation and then square root transformation. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1). The data was reverse transformed by applying the square-exponential transformation to the transformed data to get the actual value of the data.



Plot 4.10.1 Performance of proposed model on test data set.

4.11 SUMMARY

Yesbank's historical price data were retrieved from Yahoo finance data using the yfinance library in Python. The initial Features utilised for the rudimentary stage of analysis were the Close price, Open price, High price and Low price of the Yes bank for the period from 2005-05-12 to 2023-08-09.

Pandas profiling library was utilized to conduct univariate, bi-variate analysis. Uni-variate analysis established a linear upward trend till 2018 and then there was a price crash till mid-2020. Since mid-2020, the price has been experiencing a sideways movement as discussed in section 4.2.1. Bi-variate analysis finds an existence of high co-relation among the features. The stock's closing price was selected as a feature to be incorporated into the study because of the high correlation among the features.

Hypothesis testing was done on the Yes bank stock price data set to determine if the data was stationary or non-stationary. The Dickey-Fuller test was used as a statistical tool to perform hypothesis testing. The test established that data is non-stationary with more than 99% confidence. Log transformation and then square transformation were performed to flatten it. Dicky-fuller test confirmed that transformed data is stationary with more than 99% confidence. An intuitive method was implemented to create a new data set from transformed data. The previous three days' close prices were used to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. 80% per cent of the data was kept as Training data and 20% of the data was treated as test data. 30% of the data in the training dataset was treated as Validation data. Training and test data were converted into a NumPy array to feed into the LSTM model.

Initially, three baseline models were built, namely single-layer LSTM, multi-layer LSTM and bi-directional LSTM. All three models were built with a number of RNN units=128, Adam as optimizer with a learning rate of 0.004.15 experiments were conducted on each baseline model. Hence total of 45 experiments were conducted. One best model was selected from the experiments conducted on three baseline models with different combinations of the number of epochs[10,20,50,100] and batch size[32,64,128,256]. Loss scores on the validation dataset of the baseline model and selected Experimental model were compared. An experimental Bi-directional model with epochs:10 and batch size: 64 was selected for hyperparameter tunning because it has the lowest Loss scores on the validation dataset of 322.02. Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was implemented on the test data set.

Hyperparameters of the proposed model are as follows: Epochs:10, batch size:64, Dropout rate: 0.4, learning rate: 0.0013489304109835225, Number of layers:12. Input units are the number of RNN units in each layer. input_unit1: 80, input_unit2: 32, input_unit3: 16, input_unit4: 16, input_unit5: 16, input_unit6: 16, input_unit7: 16, input_unit8: 16, input_unit9: 16, input_unit10: 16, input_unit11: 16, input_unit12: 16(refer to Table 4.8.1, trial 1). Adam optimizer was selected based on understanding from the literature review. The proposed model predicted the stock price very close to actual values with Loss and RMSE scores of 25.004 and 5.02, and also captured the moment of the stock very well as discussed in Chapter 5, Results. The proposed model was implemented on the unseen test data containing Yes Bank's share price for the period from 03-01-2020 to 04-08-2023. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1)

CHAPTER 5

RESULTS

5.1 INTRODUCTION

This section will highlight the results gained from Chapter 4, Analysis. Yes, bank data was retrieved from the Yahoo Finance data for the period from 2005-05-12 to 2023-08-09. Uni-variate analysis established a linear upward trend till 2018 and then there was a price crash till mid-2020. Since mid-2019, the price has been experiencing a sideways movement as discussed in section 4.2.1. Bi-variate analysis finds an existence of high co-relation among the features as discussed in section 4.4.2. Dickey-Fuller test finds non-stationary elements in the data set. A combination of log-square transformation was applied to the data to flatten it and the Dickey-Fuller test confirmed that transformed data is stationary with more than 99% confidence.

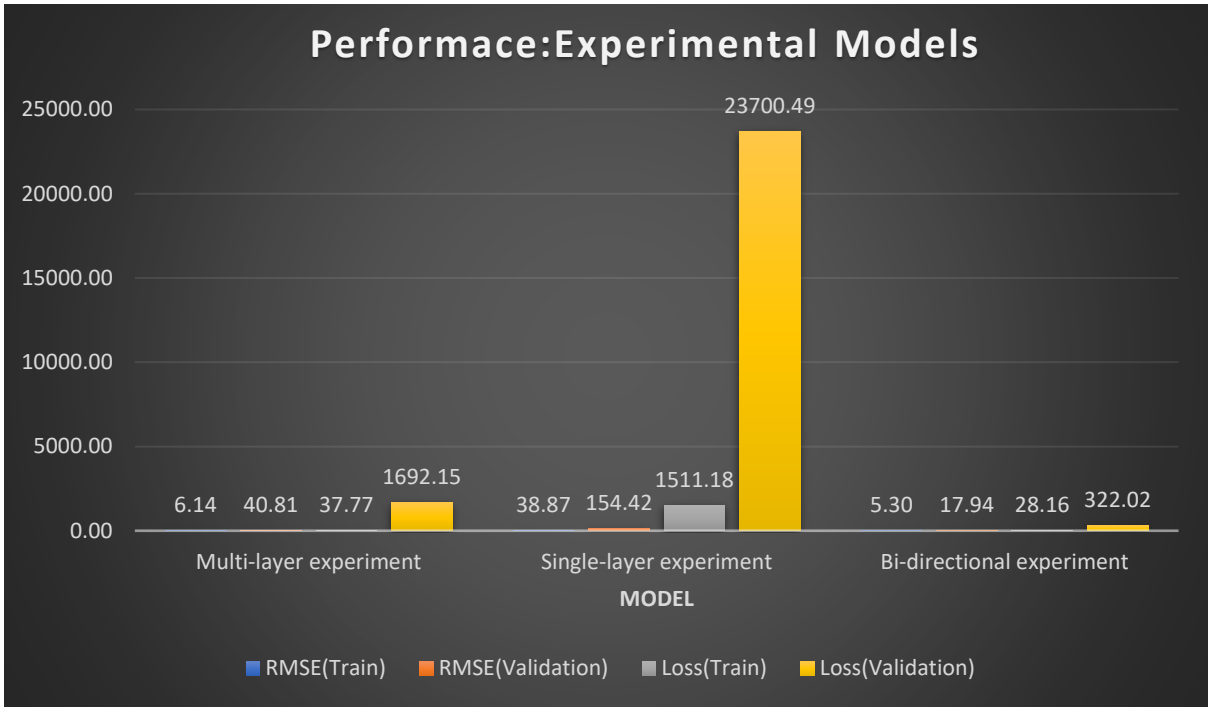
An improved Bi-directional LSTM model was proposed to implement on the unseen data, after applying the Hyperparameter on the model selected by comparing the Loss score on the validation data set among 45 experimental models and three baseline models. The proposed model was implemented on the unseen test data set containing Yes Bank's share price for the period from 03-01-2020 to 04-08-2023. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1). This section discusses the results from the experimentation, comparison of performance after Hyperparameter tunning, illustration of the proposed model, and testing on unseen data.

5.2 RESULTS FROM EXPERIMENTATION

Experiments were performed on baseline models with different combinations of Number of epochs[10,20,50,100] and batch size[32,64,128,256]. Multi-layer LSTM with the number of epochs:10 and batch size 64 was selected from 15 experimental models with the lowest loss score on the validation data set of 1692.15. Multi-layer LSTM has an RMSE score on train and validation data of 6.14 and 40.81 respectively. Multi-layer LSTM has a Loss score on train and validation data of 37.77 and 1692.15 respectively.

Single-layer LSTM with the number of epochs:50 and batch size 32 was selected from 15 experimental models with the lowest loss score on the validation data set of 23700.49. Single-layer LSTM has an RMSE score on train and validation data of 38.87 and 154.42 respectively.

Single-layer LSTM has a Loss score on train and validation data of 1511.18 and 23700.49 respectively. Bi-directional LSTM with the number of epochs:10 and batch size 64 was selected from 15 experimental models with the lowest loss score on the validation data set of 322.02. Bi-directional LSTM has an RMSE score on train and validation data of 5.30 and 17.94 respectively. Single-layer LSTM has a Loss score on train and validation data of 28.16 and 322.02 respectively. Bi-directional LSTM outperforms Single-layer LSTM and Multi-layer LSTM on all four performance metrics. Hence, Bi-directional experimental LSTM was selected for hyperparameter tuning.



Plot 5.2.1 Performance of the selected model after the Experiments

5.3 PERFORMANCE AFTER HYPERPARAMTER TUNING.

A bi-directional LSTM with a loss score on validation data of 322.02(criteria: Loss should be the lowest) was selected from 45 experimental LSTM models and three Baseline LSTM models. Hyperparameter tuning was applied to the selected model with a certain predetermined range of hyperparameters as discussed in section 4.8 Hyperparameter tuning. Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was selected to implement on the test data set.

5.4 PROPOSED/SELECTED MODEL

The proposed model is as follows(refer to illustration 5.4.1): Epochs:10, batch size:64, Dropout rate: 0.4, learning rate: 0.0013489304109835225, Number of layers:12. Input units are the number of RNN units in each layer. input_unit1: 80, input_unit2: 32, input_unit3: 16, input_unit4: 16, input_unit5: 16, input_unit6: 16, input_unit7: 16, input_unit8: 16, input_unit9: 16, input_unit10: 16, input_unit11: 16, input_unit12: 16(refer to Table 4.8.1, trial 1). The total parameters in the model are 175,009. There are 175,009 trainable parameters and 0 Non-trainable parameters.

Illustration 5.4.1 Proposed Model: Bi-directional LSTM Model

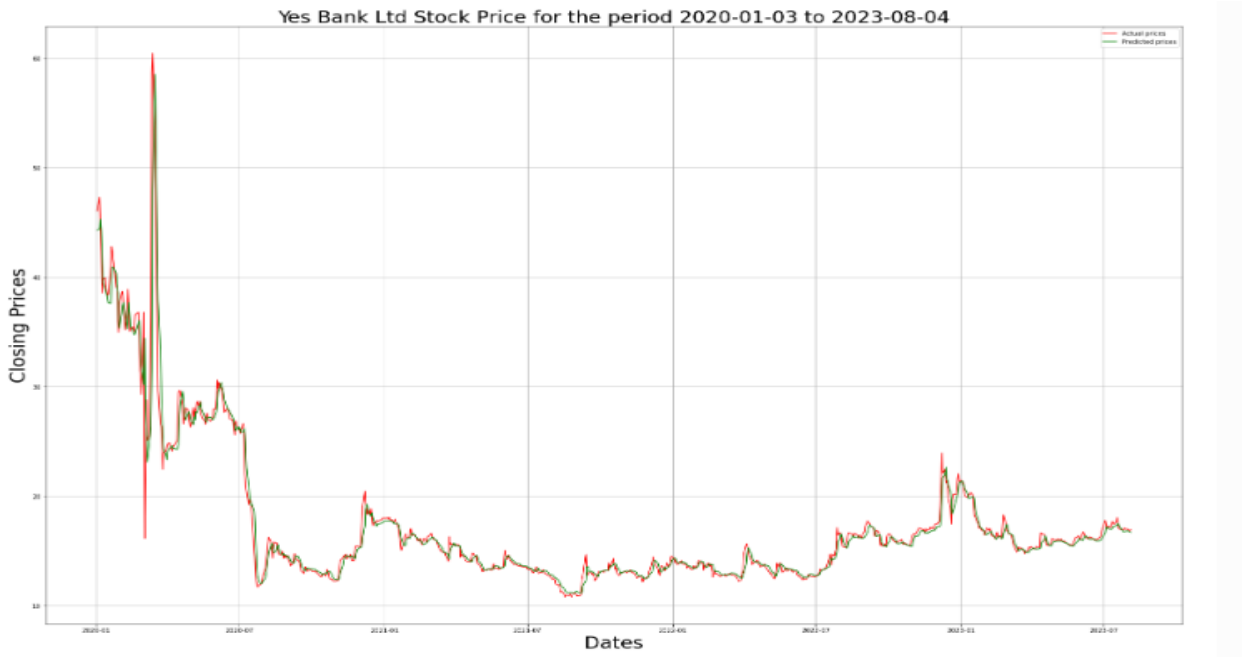
Model: "sequential"

Layer (type)	Output Shape	Param #
bidirectional (Bidirectional)	(None, 3, 80)	52480
bidirectional_1 (Bidirectional)	(None, 3, 32)	49408
bidirectional_2 (Bidirectional)	(None, 3, 16)	10368
bidirectional_3 (Bidirectional)	(None, 3, 16)	6272
bidirectional_4 (Bidirectional)	(None, 3, 16)	6272
bidirectional_5 (Bidirectional)	(None, 3, 16)	6272
bidirectional_6 (Bidirectional)	(None, 3, 16)	6272
bidirectional_7 (Bidirectional)	(None, 3, 16)	6272
bidirectional_8 (Bidirectional)	(None, 3, 16)	6272
bidirectional_9 (Bidirectional)	(None, 3, 16)	6272
bidirectional_10 (Bidirectional)	(None, 3, 16)	6272
bidirectional_11 (Bidirectional)	(None, 3, 16)	6272
dropout (Dropout)	(None, 3, 16)	0
bidirectional_12 (Bidirectional)	(None, 16)	6272
dropout_1 (Dropout)	(None, 16)	0
dense (Dense)	(None, 1)	33

Total params: 175,009, Trainable params: 175,009, Non-trainable params: 0

5.5 TESTING PROPOSED/SELECTED MODEL ON UNSEEN/TEST DATA

The proposed model was implemented on the unseen test data set containing Yes Bank's share price for the period from 03-01-2020 to 04-08-2023. The data was transformed to stationary data by applying log transformation and then square root transformation. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1).The proposed model was able to predict the future price of the Yes bank with mean variability of 5.002 across the unseen data. Table 5.5.1 has summarised the first 5 and last 5 actual and predicted values with date. The data was reverse transformed by applying the square-exponential transformation to the transformed data to get the actual value of the data. The Proposed model was also able to predict the trend closely to the real trend(refer to plot 5.5.1).



Plot 5.5.1 Predicted vs. Actual Stock Price (Model implemented on Unseen Data)

Table 5.5.1 Predicted vs. Actual Stock Price (Model implemented on Unseen Data)

Date	Actual_close_price	Predicted_close_price
2020-01-03	46.049999	44.301823
2020-01-06	47.299999	44.389069
2020-01-07	44.750000	45.273369
2020-01-08	42.099998	44.278538
2020-01-09	38.549999	42.562698

Date	Actual_close_price	Predicted_close_price
2023-07-31	17.000000	16.802393
2023-08-01	16.950001	16.825947
2023-08-02	16.900000	16.775229
2023-08-03	16.850000	16.744091
2023-08-04	16.950001	16.697491

5.6 SUMMARY

This section has summarized the result gained from the Chapter 4, analysis. A bi-directional LSTM model was selected from 45 experimental models and 3 baseline models for its low Loss score on validation data of 322.02. Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was selected to implement on the test data set. The proposed model was implemented on the unseen test data set containing Yes Bank's share price for the period from 03-01-2020 to 04-08-2023. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1)(refer to table 5.3.1). The Proposed model was also able to predict the trend closely to the real trend(refer to plot 5.3.1).

CHAPTER 6:
CONCLUSIONS AND RECOMMENDATIONS

6.1 INTRODUCTION

A well-structured literature review laid the foundation of the workflow and research methodology of the data. The study proposed to compare three forms of LSTM architecture, single-layer LSTM, multi-layer LSTM, and bi-directional LSTM. This study projected its analysis to propose a model that can predict the future price of Yes Bank Ltd. The study established that Bi-directional LSTM is more efficient than simple LSTM architecture in predicting the stock price as well as trend in the price moment.

6.2 DISCUSSION AND CONCLUSION

This research incorporated the historical price information of the Yes Bank Ltd data from the period 2005-07-12 to 2023-08-09. There has been a consistent rise in the price from the year 2005 to 2008 then stock price experienced a consistent fall from the first quarter of the year 2008 to mid-2009 due to the effect of sub-prime loan risk. The stock experienced a consistent rise in price from mid-2009 to the first quarter-2019 then the stock experienced a deep fall in price till mid-2020. The stock has been trading range-bound since mid-2019. Bi-variate analysis establishes a strong correlation among features. The existence of the high correlation resulted in the elimination of highly correlated variables. The date and Close price were selected for further analysis. Open price, High price and Low price were eliminated from the data set. The Dickey-Fuller test established the presence of non-stationary elements in the data. Hence, log-sqrt transformation was successfully applied to remove the non-stationary element. An intuitive approach was adopted to create a new data set by utilising the previous three days' close prices to predict the close price of the next day. The literature review laid the foundation to research the ability of the LSTM as an efficient architecture to predict future stock prices. After analysing various scientific literature regarding the application of machine learning, this study was able to understand that various studies have appreciated the ability of the LSTM to predict the stock price more efficiently than other algorithms.

Initially, A bi-directional LSTM model was selected from 45 experimental models and 3 baseline models for its low Loss score on validation data of 322.02. Hyperparameter tuning has improved the Loss by 88.28% from 322.02 to 37.88 (refer to Plot 4.9.1). Hence, a Bi-directional tuned Model was selected to implement on the test data set. The proposed model was implemented on the unseen test data set containing Yes Bank’s share price for the period from

03-01-2020 to 04-08-2023. The proposed model was able to predict the price with a loss of 25.024 and within the average value of 5.002(refer to Plot 4.10.1)(refer to table 5.3.1). The Proposed model was also able to predict the trend closely to the real trend(refer to plot 5.3.1). This paper established that simple LSTM architecture such as single-layer LSTM was less efficient than complex LSTM architecture such as multi-layer LSTM and bi-directional LSTM. The study also established that bi-directional LSTM was able to predict the trend of the moment of the stock efficiently(refer to plot 5.3.1).

6.3 CONTRIBUTION TO KNOWLEDGE

This research was able to establish the efficiency of the LSTM model by comparing three forms of LSTM architecture namely, single-layer LSTM, multi-layer LSTM and Bi-directional LSTM. This Study brought an intuitive approach to feature engineering by using the previous three days' close prices to predict the close price of the next day. The previous three day's price was used as the explanatory variable and every 4^{th}-day's price was treated as the predictor variable. The study was able to keep the feature engineering process simple and derive new features from the close price.

6.4 FUTURE RECOMMENDATIONS

This paper established an LSTM-based stock price prediction framework by comparing the performance of single-layer LSTM, multi-layer LSTM and Bi-directional LSTM. The study proposes to extend its research by combining various forms of LSTM architecture with other models such as CNN and Hidden Markov model etc. to predict the stock price and improve the current model performance. Kabbani, Id and Usta (2022) found that including technical indicators, macroeconomic data and financial news sentiment scores improved the prediction power of the LSTM model. Hence, The study proposes to incorporate the macro-economic data, and technical indicators to improve the prediction strength of the proposed model. The paper also inspires to work on automated stock price prediction using LSTM.

REFERENCES

1. Bhandari, H.N., Rimal, B., Pokhrel, N.R., Rimal, R., Dahal, K.R. and Khatri, R.K.C. (2022).Predicting stock market index using LSTM. Machine Learning with Applications, p.100320.
   Available at: https://doi.org/10.1016/j.mlwa.2022.100320. [Accessed 25th May 2023].

2. Yang, J., Wang, Y. and Li, X. (2022). Prediction of stock price direction using the LASSO-LSTM model combines technical indicators and financial sentiment analysis. PeerJ Computer Science, 8, p.e1148.
   Available at: https://doi.org/10.7717/peerj-cs.1148. [Accessed 26th May 2023].

3. Usmani, S. and Shamsi, J.A. (2023). LSTM-based stock prediction using weighted and categorized financial news, PLOS ONE, 18(3), p. e0282234.
   Available at: https://doi.org/10.1371/journal.pone.0282234. [Accessed 30th May 2023].

4. Perry, J. (2023). Predictive AI for the S&P 500 Index Predictive AI for the S&P 500 Index. Available at:https://digitalcommons.dartmouth.edu/cgi/viewcontent.cgi?article=1023&context=cs_senior_theses [Accessed 1st June 2023].

5. Wei, X., Ouyang, H. and Liu, M. (2022). Stock index trend prediction based on TabNet feature selection and long short-term memory. PLOS ONE, 17(12), p.e0269195. Available at: https://doi.org/10.1371/journal.pone.0269195. [Accessed 5th June 2023].

6. Biswas, A., Uday, I.A., Rahat, K.M., Akter, Mst.S. and Mahdy, M.R.C. (2023). Forecasting the United States Dollar(USD)/Bangladeshi Taka (BDT) exchange rate with deep learning models: Inclusion of macroeconomic factors influencing the currency exchange rates. PLOS ONE, [online] 18(2), p.e0279602. Available at: https://doi.org/10.1371/journal.pone.0279602. [Accessed 6th June 2023].

7. Orsel, O. and Yamada, S. (2022). Comparative study of machine learning models for stock price prediction. Available at: https://arxiv.org/pdf/2202.03156.pdf [Accessed 12th June 2023].

8. Kabbani, T., Id, E. and Usta (2022). Predicting The Stock Trend Using News Sentiment Analysis and Technical Indicators in Spark. [online] Available at: https://arxiv.org/pdf/2201.12283.pdf. [Accessed 25th June 2023].

9. Rather, A.M. (2021). LSTM-based Deep Learning Model for Stock Prediction and Predictive Optimization Model. EURO Journal on Decision Processes, 9, p.100001. Available at: https://doi.org/10.1016/j.ejdp.2021.100001. [Accessed 1st July 2023].

10. Shah, J., Jain, R., Jolly, V. and Godbole, A. (2021). Stock Market Prediction using Bi-Directional LSTM. [online] IEEE Xplore. Available at: https://doi.org/10.1109/ICCICT50803.2021.9510147. [Accessed 10th July 2023].

11. Shahi, T.B., Shrestha, A., Neupane, A. and Guo, W. (2020). Stock Price Forecasting with Deep Learning: A Comparative Study. Mathematics, 8(9), p.1441. Available at: https://doi.org/10.3390/math8091441. [Accessed 20th July 2023].

12. Althelaya, K.A., Mohammed, S.A. and El-Alfy, E.-S.M. (2021). Combining Deep Learning and Multiresolution Analysis for Stock Market Forecasting. IEEE Access, 9, pp.13099–13111. Available at: https://doi.org/10.1109/access.2021.3051872. [Accessed 22nd July 2023].

13. Saud, A.S. and Shakya, S. (2020). Analysis of lookback period for stock price prediction with RNN variants: A case study on banking sector of NEPSE. Procedia Computer Science, 167, pp.788–798. Available at: https://doi.org/10.1016/j.procs.2020.03.419. [Accessed 23rd July 2023].

14. Wu, S., Liu, Y., Zou, Z. and Weng, T.-H. (2021). S_I_LSTM: stock price prediction based on multiple data sources and sentiment analysis. Connection Science, pp.1–19. Available at: https://doi.org/10.1080/09540091.2021.1940101. [Accessed 25th July 2023].

15. Ma, R., Zheng, X., Wang, P., Liu, H. and Zhang, C. (2021). The prediction and analysis of COVID-19 epidemic trend by combining LSTM and Markov method. Scientific Reports, [online] 11(1), p.17421. Available at: https://doi.org/10.1038/s41598-021-97037-5. [Accessed 26th July 2023].

16. Shah, J., Vaidya, D. and Shah, M. (2022). A comprehensive review on multiple hybrid deep learning approaches for stock prediction. Intelligent Systems with Applications, 16, p.200111. Available at: https://doi.org/10.1016/j.iswa.2022.200111. [Accessed 27th July 2023].

17. Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780. Available at: https://doi.org/10.1162/neco.1997.9.8.1735. [Accessed 28th July 2023].

18. Bao, W., Yue, J. and Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7), p.e0180944. Available at: https://doi.org/10.1371/journal.pone.0180944. [Accessed 28th July 2023].

19. Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2), pp.654–669. Available at: https://doi.org/10.1016/j.ejor.2017.11.054. [Accessed 28th July 2023].

20. Qiu, J., Wang, B. and Zhou, C. (2020). Forecasting stock prices with long-short term memory neural network based on attention mechanism. PLOS ONE, 15(1), p.e0227222. Available at: https://doi.org/10.1371/journal.pone.0227222. [Accessed 29th July 2023].

21. Lanbouri, Z. and Achchab, S. (2020). Stock Market prediction on High frequency data using Long-Short Term Memory. Procedia Computer Science, 175, pp.603–608. Available at: https://doi.org/10.1016/j.procs.2020.07.087. [Accessed 29th July 2023].

22. Yadav, A., Jha, C.K. and Sharan, A. (2020). Optimizing LSTM for time series prediction in the Indian stock market. Procedia Computer Science, 167, pp.2091–2100. Available at: https://doi.org/10.1016/j.procs.2020.03.257. [Accessed 29th July 2023].

23. Zou, Z. and Qu, Z. (2022). Using LSTM in Stock prediction and Quantitative Trading. [online] Available at: http://cs230.stanford.edu/projects_winter_2020/reports/32066186.pdf#:~:text=In%20this%20research%2C%20we%20have%20constructed%20and%20applied [Accessed 30th July 2023].

24. Roondiwala, M,. Patel, H,. Varma, S.(2017). ResearchGate. (n.d.). (PDF) Predicting Stock Prices Using LSTM.[online]Available at: https://www.researchgate.net/publication/327967988_Predicting_Stock_Prices_Using_LSTM[Accessed 1st August 2023].

25. Rana, M., Uddin, Md.M. and Hoque, Md.M. (2019) "Effects of Activation Functions and Optimizers on Stock Price Prediction using LSTM Recurrent Networks," Conference: CSAI2019: 2019 3rd International Conference on Computer Science and Artificial Intelligence. Available at: https://doi.org/10.1145/3374587.3374622.[1st August 2023]

APPENDIX A

RESEARCH PROPOSAL

"A Comparative Study on Different LSTM architectures to predict the stock price".

CHARLIE THOMAS

STUDENT ID: 1080424

MSC IN MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

LIVERPOOL JOHN MOORES UNIVERSITY

Research Proposal

May 2023

Abstract

Stock price prediction has welcomed the development of advanced capabilities due to the speedy development of machine learning techniques and Artificial Intelligence, Improved computational speed, and ever-increasing data Availability. The stock market has become more volatile and complex due to access to investment opportunities is easier in current times. This research compares three different types of LSTM architecture(single-layer, multi-layer, and bi-directional) to predict the next-day closing price of banking stock, Yes Bank Ltd. The study uses predictors such as Historical prices(open, close, high, and low) and traded volume of the day. The fundamental data will be extracted from the Yahoo finance website for the period year 2005 to 2023. The Research will perform feature selection and scaling using appropriate measures.

The experiments will be conducted on three different types of LSTM architecture(single-layer, multi-layer, and bi-directional) with different combinations of hyperparameters to find the best model. Performance metrics such as RMSE and MSE(Loss) will be utilized to compare the model. The model with a relatively lower MSE(Loss) on validation data will be considered the best pick. The hyperparameter tuning will be performed to get the most optimized model to predict the stock price.

1. Introduction/background

The Fluctuation in the stock price is uncertain, and may interlinked reasons are proposed by various researchers for such uncertainty in price fluctuations(Karmiani, Kazi, Nambisan, Shah, & Kamble, 2019). Researchers have outlined various reasons for the uncertainty in the stock price's fluctuations such as fiscal and monetary policies, unemployment rate, economic policies, disasters(natural and manmade), health and education indicators, policies pertaining to immigration, global business and economic happenings, and several other economic and financial factors. The Stakeholders Participate in the stock market to generate maximum profits and utilise the market evaluation to reduce the risk to their investments. The major challenge in accurate prediction is to create a reliable and consistent model by extracting valuable information through different sources at higher speeds.

Some of the characteristics of stock markets such as noise in the price and market information, non-linear price movements and a chaotic system that is deemed to be deterministic(Lanbouri & Achchab , 2020). These Characteristics impede an efficient analysis to predict the price of the underlying asset in the stock market. Even though, researchers have pointed out various predictors of the stock price, still feature selection is a cumbersome and difficult process to manage efficiently. Hence, Selecting a correct feature selection technique is a complicated task. Various research has been using only historical data or technical data and others have incorporated multi-facet data such as economic data, financial data, historical price data, technical data and any other information considered to influence the fluctuation in the price of underlying assets(Gao, Zhang, & Yang, 2020). Utilizing all the available predictors can make the model complex but selecting a few features can reduce the performance of the model. The existence of high co-relation among various predictors is another challenge that may influence the model's performance. Many researchers have clearly highlighted that LSTM architecture has been outperforming other machine learning algorithms in predicting stock prices (Fischer & Krauss, 2020). This research tries to propose an LSTM model by comparing three different types of LSTM architecture(single-layer, multi-layer, and bi-directional) to predict the stock price of banking stock, Yes Bank Ltd. The study uses predictors such as Historical prices(open, close, high, and low) and traded volume of the day. The fundamental data will be extracted from the Yahoo finance website for the period year 2006 to 2023. The Research will perform feature selection and transformation using appropriate measures.

The experiments will be conducted on three different types of LSTM architecture(single-layer, multi-layer, and bi-directional) with various amalgamations of hyperparameters to find the model that is deemed to be best. Performance metrics such as RMSE and MSE will be utilized to compare the model. The model with a relatively lower MSE(Loss) on validation data will be considered the best pick. The hyperparameter tuning will be performed to get the most optimized model to predict the stock price. This research's contribution is to propose a model to predict the stock price that is less complex and will utilize fundamental data of the stock.

2. Related Study/Problem Statement

In this study, we will explore the idea that future stock prices can be successfully predicted by machine learning techniques such as LSTM. The research will utilize appropriate algorithms for time series data such as LSTM(single-layer), LSTM(Multi-layer), and LSTM(bi-directional) to predict the next day's close price.

A sequence of 40 days encompassing 10 features related to learning was derived from historical data (Lanbouri & Achchab, 2020). This transformation helped the model to improve the accuracy from 15% to 28% in comparison to the prediction model considered to be random. Rao et al. learn the features of the data by implementing the autoencoders that are stacked. The authors applied a similar transformation to financial data and predicted the opening price of the FTSE 100 index by using LSTM architecture(Deepika & Bhat, 2021). They used R-scores(average) as performance metrics and predicted the index price with an average R-score higher than 88%.

The LSTM model was used by Roondiwala et al. to predict the nifty 50's closing price(Roondiwala, Patel, & Varma, 2017). They used the NIFTY 50 data from 2011 to 2016. The study used fundamental trading features such as historical price(open, close, low, and high) and excluded indicators that are technical and macroeconomic to predict the NIFTY 50's price.

Fischer and Krauss(2020) conducted a study to predict the movement's direction for the stocks of FTSE 100 by implementing the LSTM architecture on financial data from 1990 to 2019.

The study highlighted that the LSTM architecture was able to derive reasonable information and predicted the stocks of FTSE 100's direction of movement. Based on various performance parameters such as the accuracy of the prediction of the model and returns calculated daily minus any cost related to the transaction, the LSTM(single-layer) model performs better than the LSTM(multi-layer).

A similar study was conducted by Qiu et al.(2019). Authors used various LSTM architectures on historical prices of S & P 500, Historical prices of the Hang Seng Index, and Scores of the Dow Jones Industrial Average(DJIA) (Fischer & Krauss, 2020). An attention mechanism was used by them to extract the relevant information from current dated sources to analyse the movement in the price. The noise in the data was removed by using wavelet transformation. After applying the wavelet transformation, they implemented the attention-based LSTM architecture on fundamental market data to predict the opening price of the index.

A study by Lanbouri and Achchab(2020) on high-frequency trading on historical prices of different constituent stocks of the S&P 500 was conducted to price the future stock price. An 85% and above accuracy was noted in predicting the stock price in the different minute intervals from 1 to 15. A single-layer LSTM was able to outperform the multi-layer LSTM while predicting the stock price for 1-minute intervals but the multi-layer LSTM outperforms the single layer for predicting the stock price for 5,10 and 15 minutes.

A similar study was conducted by Yadav, Jha, & Sharan(2020) on Indian stock market data. They used the LSTM architecture with numerous amalgamations of layers (hidden) to predict the stock's opening price. The scope has kept current technical trends and market movement components out of study to predict the opening price of the stocks. Their study demonstrated that the multi-layer LSTM model was outperformed by the single-layer Model based on the accuracy score.

Karmble et al.(2019) compared single-layer LSTM, bi-directional LSTM, and Multi-layer LSTM to efficiently forecast the share price. The authors selected 9 organisations and implemented different LSTM architectures on historical stock prices to predict the next day's opening price. A bi-directional LSTM outperforms other LSTM architectures by having a better R-score with a relatively low RMSE.

Yu and Yan(2019) implemented five different combinations of LSTM models with a construction method that is time-paced on auto-correlated data to predict the index price. Both models were implemented on the ChiNext index, Hang Seng Index, DJIA, Nikkei 225, China Securities Index, and S & P 500 were selected for the study. The historical price was only considered to build both prediction models. The outcomes of single-layer LSTM and Multi-layer LSTM were compared. Single-layer LSTM outperformed multi-layer LSTM for the China Securities index, Nikkei 225, and ChiNext index but failed to have better accuracy than multi-layer LSTM for S & P 500, DJIA, and Hang Seng Index. Hence, the Study was inconclusive and recommended to add market-sensitive data in further studies.

A comparative study was conducted by Gao, Zhang, & Yang(2020) on the historical prices of the three different indexes to represent three different stages of the market. Authors implemented LSTM, CNN-LSTM, CNN, and Multilayer Perceptron to predict the closing price of the indexes. The study considered predictors such as historical fundamental data(prices-open, close, high and low), the volume traded on the day, Score of MACD, True Range(Average), interest rate and rate of the exchange to index's prices.

The outcome of the research suggested that the LSTM model performed better than CNN, CNN-LSTM and Multi-layer perceptron. When variables such as the rate of unemployment and volatility index for the analysed period were selected as additional predictors, LSTM model accuracy increased by 5%. A good understanding has been built based on related works to propose an LSTM model that would be less complex and require only fundamental indicators to predict the share price.

3. Research Questions (If any)

There is no research question suggested.

4. Objectives and Aim

The research's aim is to propose an efficient LSTM model with low variability by comparing the various LSTM architectures such as Single-layer LSTM, bi-directional LSTM and Multi-

layer LSTM to predict the stock price of Yes bank Ltd. The study will incorporate predictors such as the daily historical prices(open, high, low, close) and volume of the stock traded on the given day.

The objectives of the research are as follows:

- To analyze the relationship and Pattern between the various predictors such as Historical fundamental indicators – historical prices(open, high, low, close) and volume of the stock traded on the given day.
- To compare the predictive models such as single-layer LSTM, bi-directional LSTM, and multi-layer LSTM.
- To evaluate the performance of the various LSTM models.

5. Significance of the Study

The prediction of the stock price has drawn the attention of various stakeholders to utilize various machine learning and deep learning algorithms for precise and consistent prediction of the stock or asset's price. Extensive scholarly articles have highlighted that various LSTM models have been outperforming other machine learning algorithms. The stock price has a non-linear behaviour and is affected its behaviour by various financial and economic variables. The presence of noise in the time series data and auto-correlation makes the prediction even more inconsistent.

This research proposed to develop an LSTM architecture to predict Yes Bank Limited's future price by analysing the historical price(open, close, high, low). Single-layer LSTM, multilayer LSTM, and bi-directional LSTM architectures will be implemented and their performances will be analysed by using different performance metrics such as MSE(Loss), and RMSE to identify the best model. This research's contribution is to propose a model to predict the stock price that is less complex and will utilize fundamental data of the stock.

6. Scope of the Study

The study will analyse the historical price(open, close, high and low) of the stock -Yes Bank Ltd and propose a model to predict the yes bank stock price. The study will not utilize any economic data(Macro and Micro) and indicators such as technical of the underlying stocks. Hence, the research has been limited to the utilization of trading fundamental data. The study

will incorporate only LSTM architectures. Hence, the research will not be able to compare LSTM models with other machine learning algorithms.

7. Research Methodology

The ability to learn long-term dependencies is a major characteristic of the LSTM architectures. LSTM architectures have a credible advantage over other algorithms as they can eliminate long-term biases by allowing its memory structure to select the duration of the learning procedure.

7.1 Dataset preparation

In this research, Yes Bank Limited, a popular Indian bank stock is used for the analysis and model prediction. The data will be extracted from Yahoo finance data using the yfinance library in Python. The process of feature selection has been kept very simple and features such as historical prices(open, close, low, and high) and volume traded will be utilized. A complete 20 years of data have been collected from 2005 to 2023. The time period selected incorporates the 2018 Yes Bank financial crisis, pandemics such as Covid-19 in 2019, and financial crunch in 2007-08. The model will encompass the effect of both bull and bear markets and may provide a better prediction. The price will be predicted based on the fundamental trading data.

7.2 Feature selection strategy

We will incorporate an appropriate feature selection strategy based on the importance of the level of contribution to the model to predict the stock price.

7.3 Data denoising, normalization, and input preparation

The presence of noise in stock price data is quite common. The noise in the data would be managed by appropriate transformation methods. Data will be split into train and test datasets at an 80%-20% ratio. Training data will be further divided into train data and validation data at a 70%-30% split.

7.4 Experiment and results, Model performance metrics

The research will implement the Multilayer, single-layer and bidirectional LSTM to predict the next day's closing price of the stock. Experiments will be conducted on different LSTM models with different numbers of neurons and results will be compared to find the efficient model. The stochastic behaviour of the models will be managed by executing the models independently

multiple times. Performance metrics such as R, MAPE, and RMSE will be utilized to compare the different models. A model with the highest R score and lowest MAPE and RMSE score will be selected as the best model.

7.5 Hyperparameter tuning

Hyperparameter tuning will be performed to improve the prediction ability of the model. Hyperparameter tuning will be performed in all three architectures of the LSTM namely, Single-layer LSTM, Multi-layer LSTM, and Bi-directional LSTM. Single-layer LSTM will have one LSTM layer, Multi-layer LSTM will have more than one layer of LSTM and Bi-directional LSTM will have two LSTM layers for taking input in forward and backward directions. After tuning the hyperparameters, the model will be trained with optimized hyperparameters. The values of the hyperparameters such as batch size, optimizer, and initial learning rate will help in producing different combinations of hyperparameters for each model. The lowest average MSE score on the validation data will be used as a base for selecting the best possible combination of the hyperparameters for the given model.

7.6 Comparison: single, multilayer, bi-directional LSTM models

The research will incorporate comparative boxplots of performance metrics to analyse the performance of these three LSTM architectures. We would choose LSTM architecture with a smaller MSE(Loss) score on validation data.

8. Requirements Resources

Resource	Particular/ total	Comments
Researcher	1	There is only one person involved in research
Equipment	MSI Laptop	Equipment will use to create the model
Source	Website, Books and University library	Everything accessed online
Location	Researcher's Home	
Cost	Nil	There is no cost incurred.

Time	8 weeks	Research will be conducted in two months.

9. Research Plan

- Duration is in weeks

ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Literature search	1	1	1	1	**25%**
Literature review	1	3	1	1	**25%**
Investigate & evaluate LSTM	2	4	2	4	**0%**
Develop & Test LSTM	4	2	4	2	**0%**
GET stock market data	4	Immediate	4	Immediate	**0%**
Train the LSTM	4	3	4	3	**0%**
Experiment the LSTM model	5	3	5	3	**0%**
Hyper tune the selected model	5	2	5	5	**0%**

Analyse and Evaluate the final model	5	2	5	2	**0%**
Complete the report	6	2	6	2	**0%**

References

1. Garcia C., Goldstein J., and Kestenbaum D.(2021) Planet money summer school 2: Index funds & the bet. URL https://www.npr.org/2021/07/29/1022440582/ planet-money-summer-school-2-index-funds-the-bet.
2. Deepika.N and Bhat M.N.(2021) An efficient stock market prediction method based on kalman filter. Journal of The Institution of Engineers (India): Series B, 102(4):629–644, Aug 2021. ISSN 2250-2114. doi:10.1007/s40031- 021-00583-9. URL https://doi.org/10.1007/s40031-021-00583-9.
3. Karmiani D., Kazi R., Nambisan A., Aastha Shah A., and Kamble V.(2019) Comparison of predictive algorithms: Backpropagation, svm, lstm and kalman filter for stock market. In 2019 Amity International Conference on Artificial Intelligence (AICAI), pages 228–234, 2019. doi:10.1109/AICAI.2019.8701258.
4. Kavinnilaa J., Hemalatha E., Jacob.M.N., and Dhanalakshmi R.(2021) Stock price prediction based on LSTM deep learning model. In 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), pages 1–4, 2021. doi:10.1109/ICSCAN53069.2021.9526491.
5. Emerson, S., Kennedy, R., O'Shea, L., & O'Brien, J. (2019, May). Trends and Applications of Machine Learning in Quantitative Finance. In 8th International Conference on Economics and Finance Research (ICEFR 2019).
6. Bao W., Yue J., Rao Y.(2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory, PLoS One, pp. 1-24
7. Fischer T., Krauss C.(2018) Deep learning with long short-term memory networks for financial market predictions, European Journal of Operational Research, pp. 654-669

8. Gao P., Zhang R., Yang X.(2020) The application of stock index price prediction with neural network, Mathematical and Computational Applications, p. 53
9. Karmiani D., Kazi R., Nambisan A., Shah A., Kamble V.(2019), Comparison of predictive algorithms: Backpropagation, SVM, LSTM and Kalman filter for the stock market, 2019 amity international conference on artificial intelligence (AICAI), pp. 228-234
10. Lanbouri Z., Achchab S. (2020). Stock market prediction on high-frequency data using long-short term memory, Procedia Computer Science, 175, pp. 603-608

The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 15th International Conference on Future Networks and Communications (FNC), The 10th International Conference on Sustainable Energy Information Technology.

11. Roondiwala M., Patel H., Varma S. (2017). Predicting stock prices using LSTM, International Journal of Science and Research (IJSR), 6 (4) pp. 1754-175

APPENDIX B: Code to Extract the Yes bank Historical price data using yfinance

In [4]:
```
# in order to specify start date and
# end date we need datetime package
import datetime

# startDate , as per our convenience we can modify
Date_start = datetime.datetime(2005, 7, 12)

# endDate , as per our convenience we can modify
Date_end = datetime.datetime(2023, 8, 10)
Data_YesBank = yahoo_Finance_data.Ticker("YESBANK.NS")

# pass the parameters as the taken dates for start and end
stock_data=Data_YesBank.history(start=Date_start,end=Date_end).iloc[:,0:4]
stock_data.head()
```

Out[4]:

Date	Open	High	Low	Close
2005-07-12	11.748900	12.479863	10.697026	10.848567
2005-07-13	10.875311	11.231879	10.242404	10.465259
2005-07-14	10.697026	10.697026	10.162174	10.206745
2005-07-15	10.028463	11.490390	10.028463	11.026852
2005-07-18	11.178391	11.713243	11.142734	11.427988

In [5]:
```
stock_data.tail()
```

Out[5]:

Date	Open	High	Low	Close
2023-08-03	17.000000	17.10	16.900000	17.000000
2023-08-04	17.049999	17.15	16.900000	16.950001
2023-08-07	17.000000	17.15	16.750000	16.900000
2023-08-08	16.900000	17.00	16.750000	16.850000
2023-08-09	16.900000	17.00	16.799999	16.950001

Appendix C: Code for Generating Data Report using Pandas Profile report.

In [13]:
```
#Use pandas profiling to generate the data profile report and display as widgets
Data_report=pp.ProfileReport(stock_data)
Data_report.to_widgets()
```

Summarize dataset: 100% 34/34 [00:03<00:00, 10.45it/s, Completed]

Generate report structure: 100% 1/1 [00:01<00:00, 1.75s/it]

Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview | Alerts (17) | Reproduction

Number of variables	5	DateTime	1
Number of observations	4461	Numeric	4
Missing cells	0		
Missing cells (%)	0.0%		
Duplicate rows	0		
Duplicate rows (%)	0.0%		
Total size in memory	174.4 KiB		
Average record size in memory	40.0 B		

Report generated with pandas-profiling.

In [14]:
```
#Save the report to local machine as html file
```

Appendix D: Code for Feature Engineering and Creating new data set

In [7]:

```
def create_input_target(df):

    df.rename(columns = {'Close':'Input_1'}, inplace = True)
    df1=df.iloc[1:,:]
    df1.rename(columns = {'Input_1':'Input_2'}, inplace = True)
    df2=df1.iloc[1:,:]
    df2.rename(columns = {'Input_2':'Input_3'}, inplace = True)
    df3=df2.iloc[1:,:]
    df3.rename(columns = {'Input_3':'Target'}, inplace = True)

    df_actual=pd.DataFrame(pd.concat([df,df1,df2,df3],axis=1))
    df_actual['Input_2']=df_actual['Input_2'].shift(-1)
    df_actual['Input_3']=df_actual['Input_3'].shift(-2)
    df_actual['Target']=df_actual['Target'].shift(-3)

    df_actual=pd.DataFrame(df_actual.iloc[:-3,:])
    df_actual=df_actual.reset_index()
    return df_actual
df_actual= create_input_target(Data_YesBank_ClosePrice)
```

```
C:\Users\44775\anaconda3\lib\site-packages\pandas\core\frame.py:5039: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

Appendix E: Code for Hyper Parameter Tunning

Hyperparameter Tunning

In [35]:

```
#This is in contrast to the tuner approach where options for hyper parameters "hp" are specified and passed to the model

def build_model(hp):
    model = Sequential()

    for i in range(hp.Int('num_layers', 2, 20)):
        model.add(Bidirectional(LSTM(hp.Int('input_unit'+ str(i),min_value=16,max_value=128,step=16),activation='relu',return_sec
    model.add(Dropout(hp.Float('Dropout_rate',min_value=0,max_value=0.5,step=0.1)))
    model.add(Bidirectional(LSTM(hp.Int('input_unit'+ str(i),min_value=16,max_value=128,step=16),activation='relu', return_sequer
    model.add(Dropout(hp.Float('Dropout_rate',min_value=0,max_value=0.5,step=0.1)))
    model.add(Dense(1))
    learning_rate = hp.Float("lr", min_value=1e-4, max_value=1e-2, sampling="log")
    import keras
    model.compile(loss='mean_squared_error', optimizer=keras.optimizers.Adam(learning_rate=learning_rate), metrics=[tf.keras.metr
    return model

tuner= kt.RandomSearch(
    hypermodel=build_model,
    objective=kt.Objective("val_loss", direction="min"),
    max_trials=1,
    executions_per_trial=3,
    overwrite=True,
    directory="LSTM_search",
    project_name="Search_result"
    )
```

```
epoch=10
batch_size=64
stop_early = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)

file_path = 'best_model_lstm_selected.h5'

mc = tf.keras.callbacks.ModelCheckpoint(file_path, monitor='val_loss', mode='min', verbose=1, save_best_only=True)
tuner.search(
        x=X_train,
        y=y_train,
        epochs=epoch,
        batch_size=batch_size,
        validation_data=(X_train,y_train),validation_split=0.3, callbacks=[stop_early,mc]
)
```

Appendix F: Code for Dickey fuller test to check whether data is stationary or not.

In [5]:
```
def test_stationarity(timeseries):
    '''
    Input: timeseries (dataframe): timeseries for which we want to study the stationarity
    '''

    #Determing rolling statistics
    rolmean = timeseries.rolling(20).mean()
    rolstd = timeseries.rolling(20).std()

    #Plot rolling statistics:
    plt.plot(timeseries, color='blue',label='Original')
    plt.plot(rolmean, color='red', label='Rolling Mean')
    plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend()
    plt.figure(figsize = (30,10))

    plt.title('Rolling Mean & Standard Deviation')
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value',\
                                             '#Lags Used','Number of Observations Used'])
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
```

In [6]:
```
#df_actual.set_index('Date',inplace=True)
test_stationarity(Data_YesBank_ClosePrice)
```