National College of Ireland

Prediction Stacking for long-term forecasting of the Bombay Stock Exchange Index by leveraging Machine and Deep Learning Models

MSc Research Project
Data Analytics

Sumanth Bijadi Sridhar Rao
Student ID: x18140181

School of Computing
National College of Ireland

Supervisor: Dr. Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Sumanth Bijadi Sridhar Rao
Student ID:	x18140181
Programme:	Data Analytics
Year:	2018
Module:	MSc Research Project
Supervisor:	Dr. Catherine Mulwa
Submission Due Date:	23/04/2020
Project Title:	Prediction Stacking for long-term forecasting of the Bombay Stock Exchange Index by leveraging Machine and Deep Learning Models
Word Count:	3972
Page Count:	30

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	23rd April 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Prediction Stacking for long-term forecasting of the Bombay Stock Exchange Index by leveraging Machine and Deep Learning Models

Sumanth Bijadi Sridhar Rao
x18140181

Abstract

Stock markets are considered to be very volatile and for the most part unpredictable. It is certainly challenging to predict stock indices on a short-term basis let alone forecast it over a long period of time. In this research project, the main objective is to find reasonable answers to the research questions considered and to do so, the prediction model stack approach is proposed in order to combine the predictive capabilities of the individual machine and deep learning models to further optimize the predictions made, while, significantly reducing the margin of error. Results obtained showed that the BSE stock index can be predicted to a great extent over a long period of time and that the LSTM RNN model predicts the future stock index prices quite accurately and it achieved an R^2 score of 0.9492. The RF-DLSTM prediction model stack, implemented using the proposed prediction model stacking approach, outperformed the LSTM RNN model with an R^2 score of 0.966, RMSE score of 0.0503 and MAE score of 0.0408.

Keywords: *Financial Forecasting, Regression, Stock Index, Custom Loss Function, Deep Neural Network, Long-Short Term Memory network, Decision Tree, Gradient Boosting, Random Forest, Prediction Stacking, Forecasting, Machine Learning, Deep Learning.*

1 Introduction

India is known to be the fifth-largest economy by nominal GDP and is regarded as a developing market economy. It is generally believed that the stock market reflects the economic conditions of a nation. And hence, it is important to be able to predict where the market is headed to a day, as far in the future as possible. But, forecasting over a long period of time has been a key challenge for quite a while now. The following research question focuses on addressing this setback and comb through suitable solutions to deal with the issue.

Economic transaction of stocks that represent ownership of claim on businesses are considered to be a loose network. Stock market trading works as a unique investment tool for investors for the very reason that it offers liquidity which in turn allows them to either hold or sell those stocks with ease in order to gain profit. The efficient market hypothesis or commonly known as EMH states that the market is, as the name suggests, efficient

and that there is no space for predictive analysis but researchers have been adamant on proving that movement of stocks, shares and indices within a financial market is predictable. Previously many techniques have been employed to predict stock market behaviour but have failed due to the volatile nature of the market which is quite the issue to solve. The two most widely incorporated approaches to predict movement of the market are fundamental and technical analysis. The drawback of using these techniques alone, is that their forecasting capabilities only enable short-term predictions and even so, not to a great extent (Kumar and Ningombam; 2018).

It is important to understand what the stock index is and why it is of importance. A stock market index, basically is a statistical measure which is capable of indicating how a particular portfolio of stocks changes over time. This portfolio of stocks is only a portion of the overall market. The stock market index is taken as the total value accrued when different stocks are combined together. This total value is with respect to a base value at a particular period of time. Forecasting index values allows investors to evaluate returns based on changes in market values over longer time periods. Due to the stock markets' chaotic and nonlinear nature, there is need of a solution that can identify the chaotic aspects and make valid and trustworthy predictions. Artificial neural network, commonly referred to as ANN is considered to be one such solution (Ravi et al.; 2017).

This research project proposes a prediction model stacking approach to combine multiple machine learning models with deep learning models to optimize and improve upon the individual predictive capabilities of the machine and deep learning models. The proposed approach is used to implement RF-DNN, RF-LSTM, Deep-LSTM and RF-DLSTM prediction stacks, which are developed upon the machine learning and deep learning models such as Random Forest Regressor, Deep Neural Network and LSTM Recurrent Neural Network. The following Research Questions are to be answered and this report documents the findings with respect to the machine learning and deep models implemented in order to find answers to the research questions specified in section 1.1 below.

1.1 Research Question

RQ: *"To what extent can the BSE Index be accurately predicted over a long period of time by implementing prediction stacks of machine learning and deep learning models?"*

Sub-RQ: *"Can a custom formulated loss function minimize the margin of error for a deep learning model?"*

In order to find reasonable answers to the above mentioned research questions, the following research objectives are being set to accomplish.

1.2 Objectives

Table 1: Research Objectives

	Milestone	Significance	Objective Description
1	Feature Engineering	Engineering both Technical and Fundamental Indicators using the Original Data	1. Simple Difference between High and Low Closing Prices 2. Simple Difference between Open and Close Prices 3. 30 day Future Closing Price 4. 7 day Future Closing % Change 5. 14 day Future Closing % Change 6. 7 day Future Closing Difference 7. Closing Price with lag of 1 and 2 based on ACF and PACF plots 9. Moving Average (MA) 10. Relative Strength Index (RSI) 11. 7, 14 and 30 day Rolling Mean
2	Modeling	Training multiple models to forecast Adjusted Closing Price over a long period of time	1. Linear Regressor 2. Decision Tree Regressor 3. Random Forest Regressor (RFR) 4. Gradient Boosted RFR (GBRFR) 5. Deep Neural Network (DNN) 6. LSTM Network
3	Custom Loss Function (CLF)	Manually defined loss function to assign penalty for every incorrect prediction made	1. DNN with CLF 2. LSTM with CLF
4	Stacking	Prediction Stacking method combines the prediction capabilities of two or more predictive models and helps reduce error even further	1. LR-DNN and LR-LSTM 2. DT-DNN and DT-LSTM 3. RF-DNN and RF-LSTM 4. GBR-DNN and GBR-LSTM 5. DLSTM 6. RF-DLSTM
5	Evaluating	Model evaluation enables the assessment of their predictive capabilities	1. R^2 Score 2. RMSE Score 3. MAE Score

The structure of this research project report henceforth is as follows. Following the Introduction section, is the Related Work chapter. Chapters Methodology, Implementation, Evaluation and Results come right after. Discussion, Conclusion and Future Work chapters are located at the end of this report just before the References section.

2 Related Work

This section goes through brief descriptions of related research work. The following two subsections of related work go through the three major techniques used in stock price

forecasting, namely, Fundamental Analysis, Technical Analysis and Machine Learning.

2.1 Fundamental and Technical Analysis

Published work from the year 2001, as part of the Journal of Financial Economics discusses the significance of fundamental analysis, returns on stocks and short-sellers. Dechow et al. (2001) provide evidence for how short-sellers take a position such that fundamental to price ratios are low. The research done here, calculates features such as ratios of cash flow to price, earnings to price, book to market, value to market and fundamental to price as these features had predictive capabilities with respect to future stock returns. The results of this research has two interpretations. One, being that the temporary overpricing which is exploited by short-sellers is associated with low fundamental to price ratios. The other interpretation is that the ratios of fundamental to price, which are low, are somewhat in association with risk characteristics.

Author Chen, in his work from the year 2010, makes use of Support Vector Machine to forecast financial time series. The objective of this study is to apply SVM to pattern recognition techniques employed in financial engineering domain. Here, the forecasting is done, not by using the original time series data but by engineering technical indicators from the that original time series data. Author Chen claims that in finance engineering, transforming the input data space of SVM brings about good quality performance. The technical indicators developed here are, the Moving Average (MA) of a security's close price, Stochastic %K, Stochastic %D, Larry William's %R which is a momentum indicator that measures overbought or oversold levels, On Balance Volume (OBV), and Relative Strength Index (RSI).

Research work published by Deng et al. (2011), deals with combining technical analysis techniques with sentimental analysis such that future stock prices can be predicted. As observed and reported by Deng et al., stock price movements can be modeled as a function of input features so that this function can be solved as a regression problem. The authors of this work have used measures such as Root Mean Squared Error, Mean Absolute Error, and Mean Absolute Percentage Error to evaluate experimental results. The data considered for this work were historical data of three Japanese companies in US stock market. The model proposed here, has four major components, of which, the last two are of utmost importance. First of the two being, Feature Extraction Component involving the extracting of features needed from two different sources, time series data and web sources. Technical indicators of elements such as price and volume from time series source and numerical features from web source. The other major component deals with prediction and evaluation of the model proposed. It was observed that features different from those extracted from stock prices themselves improved the performance greatly. The average RMSE, MAE and MAPE scores are observed to be 0.5108, 0.4014, and 1.3769 respectively.

Another IEEE publication, authored by Hargreaves and Hao, in the year 2012, examines the use of technical and fundamental analysis techniques and whether or not they improve the choosing of stocks. This work employs application of data mining to the Australian market. The stock selection process involved four important strategies, namely, personal trading, price trading, growth trading, growth and value trading. This

classification problem took into consideration two classes, one to indicate that there will be an increase in the stock price while the other indicates that there will be a decrease in the stock price. In order to achieve this, they first identified the sector that performed the best and to do so, sector price trends are compared against the ASX index. Using the above mentioned stock selection strategies, five variables are considered such that the input variables stay consistent (Hargreaves and Hao; 2012). These five variables are Return on Equity, Return on Assets, Analyst Opinion, Growth this Year, and Price. This work by Hargreaves and Hao shows how data mining techniques can be made use of with respect to stock data and the results observed indicate that the metrics ROE, ROA, AO, GTY and Price are good predictors and these findings justify making use of data mining techniques for prediction purposes.

Udagawa, Faculty of Engineering at Tokyo Polytechnic University, published his work on prediction of stock price trends using the Candlestick Chart Blending Technique, in the year 2018. His work mainly deals with technical analysis technique for the predictions and suggests a blending algorithm wherein candlesticks are combined while sharing a certain price range so as to eliminate the noisy ones. The technique proposed here is applied to the Nikkei-225 stock daily average. The results of the blended candlesticks show that when compared to the original chart, the final chart obtained by blending two candlestick charts reveal trends of price movements by eliminating those candlesticks generated at the time or period of markets' indecisiveness. Although, the experimental results also show that this method performs best and is effective only in the case of stock-term price predictions (Udagawa; 2018).

Beyaz et al. (2018) documents the comparison of experimental results obtained using indicators based on technical and fundamental analysis. The experiments are run against a hundred and forty companies from the S&P 500, and these experiments show that models using the indicators based on fundamental analysis outperform those based on technical analysis. This publication also documents that in more than 95% of the cases during experimentation, results obtained by combining the above mentioned indicators has a lower RMSE score when compared to those results obtained by using the indicators individually. Some of the more significant technical and fundamental indicators considered are Relative Strength Index (RSI), Moving Average Convergence Divergence (MACD), Average True Range (ATR), Money Flow Index (MFI), Rate of Change (ROC), and S&P 500 Future Prices. The aim of the research was to forecast stock price change in 126 and 252 days. The RMSE scores when the models used fundamental and technical indicators for 126 and 252 days are 0.1685, 0.2110 and 0.1464, 0.1693 respectively.

2.2 Machine Learning

The research work documented by Lee at Sungshin Women's University, adopts reinforcement learning for stock price forecasting by considering the stock price changes as a Markov process. Here, the Markov process is modeled by considering reinforcement learning elements such as state, action, reward, policy and so on. Lee (2001) mentions the use of TD(0) algorithm and claims that the stock price prediction considered as Markov process can be optimized by the same. The experimental result in this work is based on the Korean stock market. The data for training and testing are chosen as 100 stocks for 2 years and 50 stocks for 1 year respectively. The evaluation metric chosen in this work

is the Root Mean Squared Error and the experimental results are evaluated as the error between value grades and return grades by considering a certain range. For each of the three ranges considered, the average of the RMS error is calculated. The average returns after 5 days is noted to have the least RMSE scores across all three grade ranges at 3.07, 2.62 and 3.38 while the average returns after 20 days is noted to have the highest RMSE scores across all three grade ranges at 4.65, 3.46 and 4.99 respectively.

Another research, Tsai and Wang (2009) takes the approach of combining two models, namely, Artificial Neural Network and Decision Tree to forecast stock prices. As mentioned in this research work, combining models as such, enhances the rate of prediction accuracy in stock price forecasting. Authors Tsai and Wang have documented their observations on the average accuracy of multiple variations of both the ANN and DT models. The DT models have an average accuracy of 65.415% and the ANN models have 59.016%. By combining the Decision Tree models with the Artificial Neural Network models, an average accuracy of 77.1985% is achieved.

Zhao et al. (2009) documents the set up of a two-phase forecast approach wherein, the first phase makes use of Gray Correlation Analysis to choose a set of variables that can give the description of characteristics of the stock market. This research work involves making use of traditional BP neural network and the combination of SOFM neural network and the traditional BP neural network. Evaluation metrics used to interpret experimental results are Mean Absolute Error (MAE) and Correct Sign Ratio (CSR). The forecasting results of both the traditional model and combined model are compared and it is noted that the Mean Absolute Error in both cases is 0.0016 and hence, the deciding factor, being the Correct Sign Ratio, indicates that the combination of the traditional model and the SOFM neural network performs better with a CSR score of 0.6891 compared to 0.5407 achieved by the traditional BP neural network model.

Dai et al. (2011) adopts a Support Vector Machine method so that the stock index futures can be predicted for the next five days. The authors of this research work document the use of an information granulation methodology in order to transform the data of three stock index futures into a series of fuzzy granules and then minimum, median and maximum values of future opening price are extracted in each granule. This work adopts a three step process, first of which is the preprocessing of data to normalize it within a range of 100 and 500. Secondly, optimal parameters are searched and selected by the SVMcgRorRegress function and as part of the last and final step, the SVM model and trained and predictions are made accordingly. The evaluation method here is the variation range. The three stock indexes considered for this research work are CSI 300, IF 100 and IF 1009. The original variation range for CSI 300 stock index are noted to be 2961.4, 3162.9, and 3323.8 while the predicted variation range for the same are observed to be 3071.3, 3166.7, and 3270.3. For the IF 100 stock index, the original variation range is noted to be 3299, 3348, and 3381.9 while the predicted range for the same is observed to be 2929.6, 3011.4, and 3036.2. Similarly, for the IF 1009 stock index, original variation range is noted to be 2851.8, 2916.4, and 2934 while the prediction range for the same is observed to be 2929.6, 3011.4, and 3036.2.

Iuhasz et al. (2012) documents the use of a hybrid system based on a Multi-Agent Architecture. And this hybrid system investigates the evolution of some of the neural

networks alongside the fundamental and technical analysis techniques. The above mentioned models and techniques are applied on stock market indexes in order to improve the stock market behaviour. The models proposed here are Artificial Neural Network (ANN), Elman and Jordan Recurrent Neural Network and a Neural Network evolved with Neuro-Evolution of Augmenting Topologies (NEAT). The main feature taken into consideration is the closing price of stock. As for implementing the ANN, the 5 day trading close values of OLT from the Bucharest Stock Exchange are passed as inputs. The validation data is taken to be 30% smaller when compared to the training data. The above mentioned models are evaluated in this work based on the training error and validation error. The lowest training and validation error achieved are 5.18% and 5.84% with the NEAT model.

Authors Di Persio and Honchar, in their published work from the year 2016, present an Artificial Neural Network approach to make predictions with respect to trend movements of the stock market indexes. The Multilayer Perceptron, Convolutional Neural Network and Recurrent Neural Network models are chosen to be implemented based on the existing literature reviewed (Di Persio and Honchar; 2016). Along with the above mentioned models, ensembles are implemented and evaluated using the Mean Squared Error and Accuracy. The experimental results show that CNN made predictions with higher accuracy and the least error score compared to the MLP and RNN models with MSE and accuracy of scores of 0.2491 and 0.536 respectively. Similarly, among the ensembles built, the Hand-Weighted Ensemble and Average Ensemble observed MSE and accuracy scores of 0.23, 0.56 and 0.23, 0.558 respectively and by blending the two ensembles to form the Blended ensemble resulted in an MSE and accuracy score of 0.226 and 0.569 respectively.

Gao et al. (2016) is another research focused on leveraging deep learning models to achieve high precision while forecasting the stock market movement. The deep learning model implemented for this purpose is the Deep Belief Network (DBN), along with stock technical indicators (STIs) and a two dimensional principal component analysis. The aim of this work is to predict closing price of the stock market. The technical indicators engineered are Relative Strength Index (RSI) and Moving Average Convergence Divergence (MACD). The DBN model is implemented as three different variations named Model A, Model B and Model C, where Model A is fed with inputs comprising only the basic variables, Model B is fed with inputs comprising of both basic variables and stock technical indicators and Model C is fed with inputs comprising of basic inputs, stock technical indicators and two dimensional principal component analysis. The three models are evaluated with RMSE and MAE score metrics. As such, the experimental results imply that the Model C is the best performing model with RMSE and MAE scores of 12.918 and 10.192 respectively, while Model A and B achieve RMSE and MAE scores of 22.867, 17.934 and 15.548, 12.046 respectively.

Sharma and Juneja (2017) present a research work emphasizing on the stock market index future prediction based on the historical data. In this work, the authors combine the predictions from Random Forest model with the Least Square Boost (LSBoost), which is a strategy applied as a training loss function. The proposed model's performance is compared with that of the Support Vector Regressor using the evaluation metrics such as Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), and relative Root Mean Squared Error (rRMSE). The experimental results show that at 10 days ahead

of time, MAPE, rRMSE, and MSE scores for both LS-RF (LSBoost-Random Forest) and SVR are noted to be 0.8130, 0.0099, 5736.013 and 1.1524, 0.0126, 9414.83 respectively. And at 30 days ahead of time, the MAPE, rRMSE, and MSE scores calculated for the LS-RF and SVR models are observed to be 1.011, 0.0181, 16565.96 and 1.1252, 0.0126, 8987.99 respectively.

Khare et al. (2017) documents about the experimentation done with deep learning models for short-term prediction of stock prices. This work implements two predictive models namely, Feed Forward Multilayer Perceptron and LSTM Recurrent Neural Network. These two models are evaluated using the Root Mean Squared Error (RMSE) and the experimental results are based on the predictions made for the next 60 minutes which is quite close to the present rather than the far future. The experimental results imply that the MLP model performs better than LSTM RNN in case of a short term prediction. The average case and best case RMSE scores are observed to be 0.048 and 0.0188 for the LSTM model and, 0.0025 and 0.000937 for the MLP model respectively.

Kumar and Ningombam (2018) aim to evaluate the LSTM Recurrent Neural Network's effectiveness with respect to implementation of technical analysis for forecasting AAPL ticker stock prices derived from the NASDAQ Stock Exchange. The above mentioned Recurrent Neural Network model is implemented with three different optimization algorithms, namely, linear, Sigmoid and tanh. The evaluation metric being, Root Mean Squared Error, it is observed that the model with linear output activation function scaled within the range (0, 1) recorded an average RMSE score of 12.483, while the same model with Sigmoid output activation function scaled within the range (0, 1) recorded an average RMSE score of 21.769 and 21.738 for the same model with tanh output activation function.

Zarkias et al. (2019) documents the development of a novel price trailing method that is beyond the traditional forecasting system. The proposed model is implemented by reformulating trading as a control problem in order to deal with the volatile nature of the financial market. A deep reinforcement learning model called Deep Q-learning model is implemented. The neural network model is employed with specifications such as 3 hidden layers with 64 neurons each and an output layer with 3 units with the PReLU activation function for the hidden layers while output layer is initialized with no activation function. The optimizing algorithm used while training the model is Adam optimizer. This proposed model is trained using the Euro-Dollar exchange rate data and the experimental result and evaluation of the proposed model indicates that this proposed model regardless of the commission factor used, the proposed model outperforms the existing "Trading RL" approach.

Pasupulety et al. (2019) proposes an ensemble model of the SVM and Random Forest based ExtRa models so as to predict future stock prices of companies listed in the National Stock Exchange (NSE) of India. The above mentioned models' predictive capabilities are assessed based on two input types, first of which is the technical indicators, and the second being technical indicators with positive or negative tweet counts. Models are evaluated based on the R^2 score and the RMSE score. With respect to the first input type, the experimental results show that the meta-regressor set to either of the two models performs best with a significantly higher R^2 score and low RMSE score on the 5 month

data. Similarly, with respect to the second input type, it is observed that the meta-regressor set to either the SVM or ExtRa tree model in the ensemble helped achieve best performance, again, on the 5 month data. Although, the R^2 score is observed to be higher for both 5 and 6 month data, the RMSE is the determining factor and is significantly lower for the 5 month data.

3 Methodology

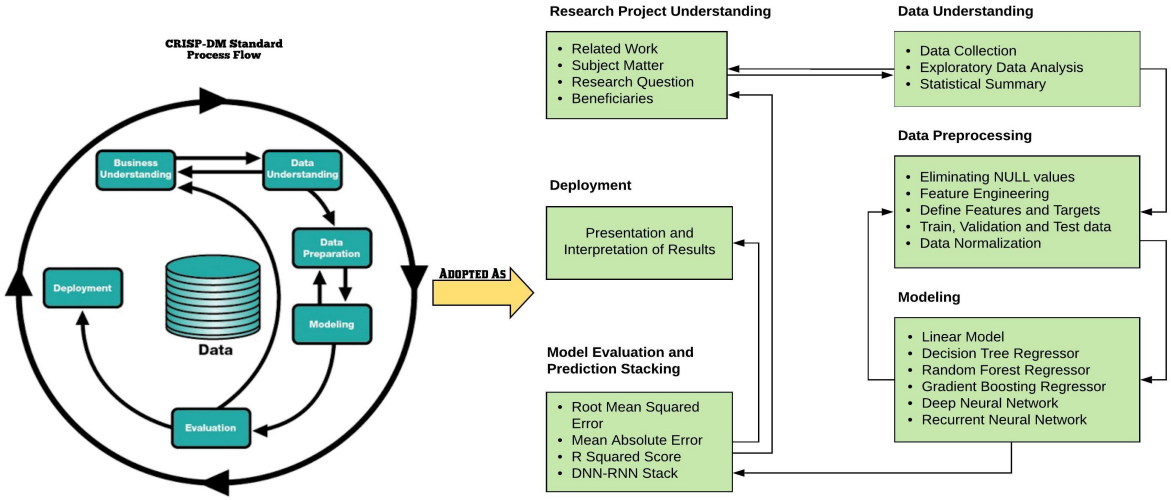

Figure 1: Adopted Process Design of CRISP-DM

The methodology incorporated to implement this project is CRISP-DM. The CRISP-DM process model, developed by Daimler Chrysler, SPSS and NCR is believed to provide a uniform framework and guidelines for those involved in mining and managing data (Azevedo and Santos; 2008). Based on the findings of Azevedo and Santos (2008), CRISP-DM is overall, more complete when compared to SEMMA. According to Wirth and Hipp (2000), the chosen process model is independent of the industry sector as well as the technology being used. Also as proposed by Wirth and Hipp (2000), the generic model of CRISP-DM is highly useful for planning the stages of a project, documentation and communication regarding the same.

The stages of CRISP-DM process model with respect to this project are as follows:

3.1 Data Understanding

The data used for this research project is obtained from the Bombay Stock Exchange official website, dating back to 1997 from the year 2020. The data holds record of 5709 days of historical prices. Each instance in the dataset has 6 recorded attributes namely, Date, Open, High, Low, Close and Adjusted Close.
The following describes what each of the above mentioned attributes represent:

1. Date: Contains all the recorded historical dates in YYYY-MM-DD format.

2. Open: Holds the Opening Stock price for that day.
3. High: Denotes the highest value of price observed that day.
4. Low: Denotes the lowest value of price observed that day.
5. Close: Holds the Closing Stock price for the day.
6. Adjusted Close: Represents a value determined by taking into account dividends, stock splits and new stock offerings with Close price as the starting point.
7. Volume: Indicates the number of stock that was moved in a day.

The historical data is available at:

`https://finance.yahoo.com/quote/%5EBSESN/history?p=%5EBSESN`

3.1.1 Exploratory Data Analysis

As part of the exploratory data analysis with respect to time series, it is important to understand how the data is varying over time, check for its seasonality, sensitivity and trend. To do this, a line graph is plotted and to check for how the data is distributed, histograms are plotted.

Figure 2 shows a line graph plot of the BSE SENSEX stock index's Adjusted Closing Price attribute from the beginning till the end date, i.e 6^{th} of March, 2020. From figure 2 it can be observed that the adjusted closing price data follows an upward trend and is most definitely not seasonal. It is also clear that the price started out quite low and increased in value rapidly after the year 2004. The histogram of historical adjusted closing price as seen in figure 3, indicates that the data is not normally distribute which is common for stock prices due to high volatility of the stock market. But this was normalized during the pre-processing stage so that the data was within a certain range, generally 0 and 1. It is evident that the lower end of price value recur more often when compared to the higher end of price value.

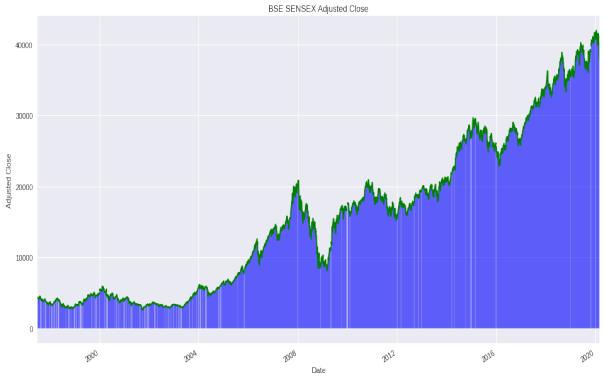

Figure 2: Adjusted Close Price

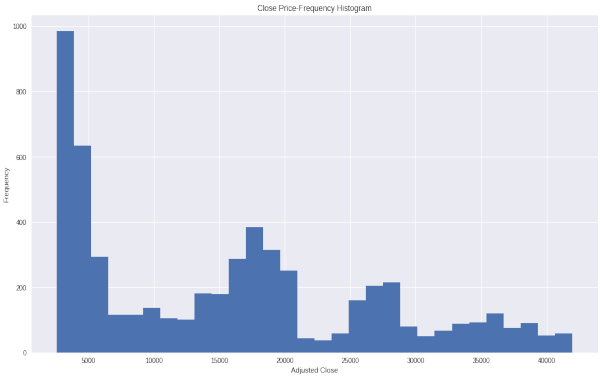

Figure 3: Histogram of Adjusted Close Price

Figure 4 represents the boxplot of the Adjusted Closing Price. This plot is used to check for outliers. Accordingly, it can be observed that there are no data points that appear past the minimum and maximum extremes of the boxplot. Majority of the values under this attribute are closer to the minimum value and not the maximum value as

indicated by the interquartile range. Figure 5 is another boxplot similar to figure 4 and this figure indicates the distribution of the Volume data. Evidently, there are many outliers, meaning the Volume data is not distributed well within a certain range and this may cause problems while training the models. One of the solutions for this problem is Normalization, the process of which is described in sub section 3.2.6 under chapter 3.2 of this report.

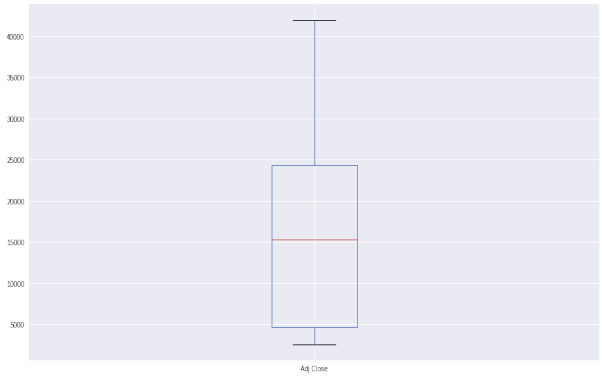

Figure 4: Closing Price Outlier Check

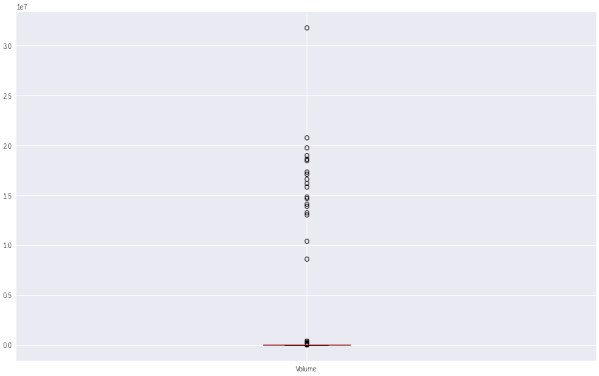

Figure 5: Index Volume Outlier Check

Figure 6 as the label suggests, is a heat map based correlation matrix of the original BSE SENSEX Index historical data. It can be observed that the daily Opening, High and Low Closing Prices are highly correlated to the Adjusted Closing Price while the Volume attribute is poorly correlated to all the other features in the dataset with a correlation score of 0.11.

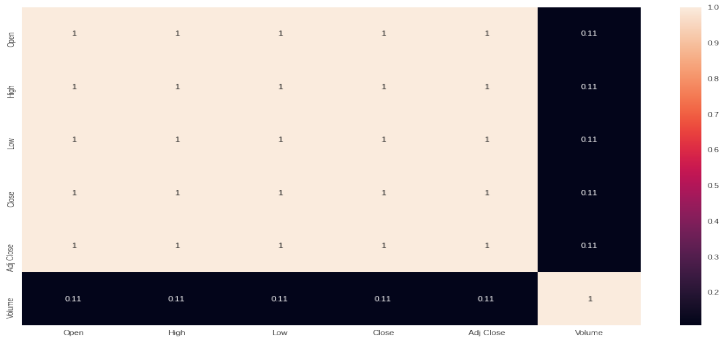

Figure 6: Correlation Matrix of BSE SENSEX Index Historical Data

3.2 Data Preprocessing

3.2.1 Eliminating Null Values

The first and foremost task before the process of engineering features was to eliminate all the instances with missing values. As there are multiple ways to deal with N/A values, the choice was made to eliminate them because the data accumulation in the case of stock index is purely based on the daily stock prices of the top 100, 300 or 500 companies which are part of that particular stock market and the stock market remains shut down or inactive on certain days that have been decided beforehand, most of these days being considered national holidays. In this case, the Bombay Stock Exchange remained closed

on a varying number of days depending on which segment of trading is being considered. It varies from 12 days to 19 days for any given trading segment within a year. Hence, it was deemed better to eliminate the N/A values as there seemed to be no significance in replacing a value that is not missing due to an erroneous data collection.

3.2.2 Feature Engineering

The insights gained from chapter 2 of this report, implies that the technical indicators are paramount to achieving accurate forecasts of any stock market related closing price. Also, from other research done before, it was observed that combining both the fundamental and technical indicators will result in forecasts that are much more precise. Based on this insight from previous research, this project involved engineering a number of features or indicators, both fundamental and technical as listed in section 1.2, under chapter 1 of this report and as such, this section is structured to provide a walk-through of the process involved in engineering those features.

First of the many features engineered, were the ratio of daily high and low closing prices and the ratio of daily open and close prices, which are the first two objectives under feature engineering listed in the research objective section 1.2. Followed by this, the adjusted closing price percent change features were calculated using the percentage function attribute of the pandas dataframe. Listed 3, 4 and 5 under the objective description column in table 1 are the 30 day Future Closing Price, 7 day Future Closing Price % Change and 14 day Future Closing Price % Change respectively. Also listed number 6 under the Feature Engineering milestone is the 7 day Future Closing Difference and this objective was accomplished by calculating the simple difference between the 7 day Future Closing Price and the 14 day Future Closing Price respectively. As for engineering the 30 day future closing price, 7 and 14 day future closing price % change, the shift and percentage change function attributes of the pandas dataframe were used.

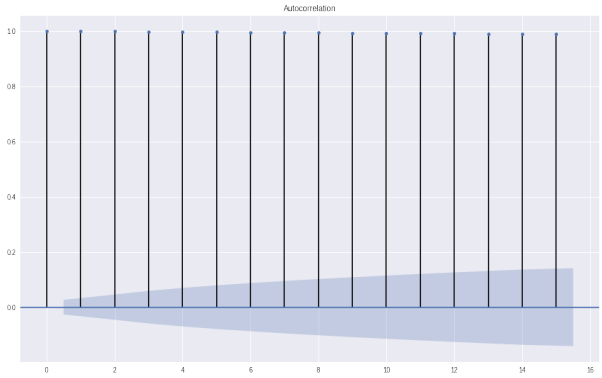

Figure 7: ACF Plot

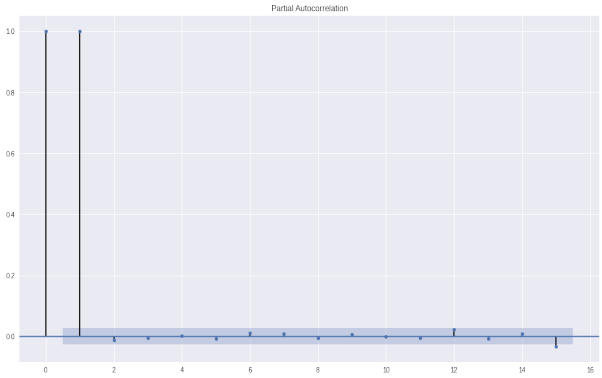

Figure 8: PACF Plot

The figures 7 and 8, respectively, are the commonly used plots to implement new features based on the number of lags that can be introduced to a particular set of data and determine how each of those lags are correlated to that set of data. The plot that is considered more significant of the two is the Partial AutoCorrelation Function plot as it indicates the number of lags with the highest correlation. As such, it can be observed the PACF plot, lags 1 and 2 are highly correlated to the Adjusted Closing Price data. Accordingly, features lag1 and lag2 are formed as listed under the feature engineering

subsection in table 1.

Further, as listed in the Research Objectives subsection of chapter 1, specifically 9, 10 and 11 under the feature engineering objectives in table 1, technical indicators such as the Relative Strength Index (RSI), Simple Moving Averages (SMA) and Rolling Mean are engineered. The Relative Strength Index and Simple Moving Averages were engineered with respect to 14, 30, 90, 180 and 360 days respectively and the Ta-Lib module was used to achieve this task. As for the Rolling Mean features, they were created using the rolling function attribute of the pandas dataframe with an argument of window equal to 7, 14 and 30 indicating rolling values for 7, 14 and 30 days respectively and the mean of this was calculated using the mean function attribute. Thus, all the objectives set under the Feature Engineering milestone in table 1 under Section 1.2 named Research Objectives were accomplished and set the path forward to the following milestones to be achieved as listed within table 1.

3.2.3 Feature Correlation

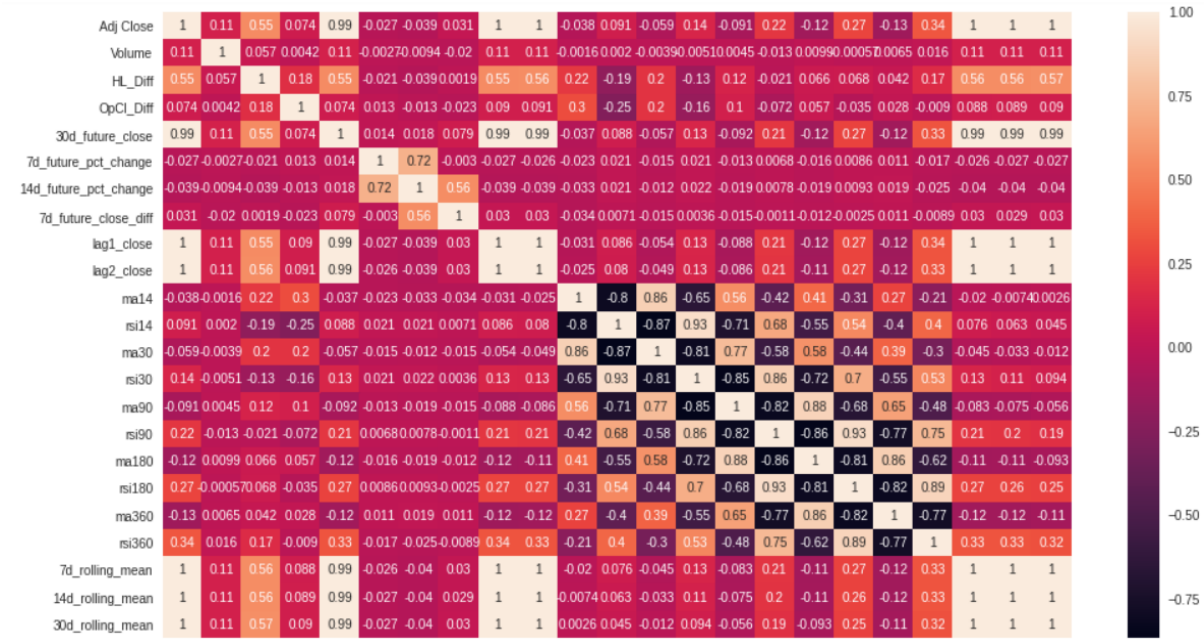

Figure 9: Complete Feature Correlation Matrix

The correlation between all the features, post feature engineering, were visualized as a heatmap to follow the correlation scores by just looking at the shade of colour on the correlation matrix as shown in figure 9. The heatmap works best when assessing statistical significance such as the correlation, especially when those correlations are to be calculated for a large number of features. By observing closely, it is evident that the relative strength index (RSI) technical indicator is positively correlated to the Adjusted Closing Price while the simple moving average (SMA) indicator is negatively correlated. The rolling mean features along with the lag features engineered based on the ACF and PACF plots can be observed to be highly correlated to the Adjusted Closing Price as well. Although not extremely negative, the remaining features seem to be slightly negative in

terms of correlation to the Adjusted Closing Price. The 30 day future closing price feature also observes a similar trend in correlations with other features as that of the Adjusted Closing Price. Hence, the features thus engineered, can be considered to be well balanced and a good representation of the volatile nature of the stock market.

3.2.4 Features and Targets

Since, the feature engineering process was concluded with 23 features in total as indicated by figure 14 in section 3.2.3, it was crucial to form features and targets to proceed further. The target feature was chosen to be the 30 day future closing price that was previously engineered during the feature engineering process, while the other 22 features were set aside as the feature set. These feature and target sets were then used to form the train, validation and test datasets.

3.2.5 Train, Validation and Test sets

The train, validation and test datasets as mentioned in the features and targets subsection, was created from the target and feature sets respectively. The training size to form the train data was 30% of the features and target sets. The train data formed as such is ranged between 5^{th} of November, 1997 and 30^{th} of December, 2011. The validation data was formed from the initally formed train data holding data till 18^{th} of April, 2013. As such, the validation data set is ranged between the beginning of year 2012 and 18^{th} of April, 2013. The remaining 70% of the features and target sets form the test data set ranging between 22^{nd} of April, 2013 and 23^{rd} of January, 2020.

3.2.6 Normalization

Normalization is a process commonly employed in data pre-processing stage to scale and fit the data within a certain range, generally between 0 and 1, such that there are no extreme values that might end up disrupting the model training process. This research project made use of the MinMaxScaler library from the scikit-learn's preprocessing module. Even though the scaling function sets the range between 0 and 1, to be absolutely certain that the data is scaled and fit between 0 and 1, the feature range was set as 0 and 1 while initializing the scaling function. Once the scaling function was initialized, fit and transform function attributes were used to normalize the data. The data was reshaped to usable format and a plot of the normalized feature proved that the data was indeed ranged between 0 and 1.

3.3 Modeling

This research project leveraged multiple machine learning models and deep learning models to forecast the adjusted closing price 30 days ahead of time for a 7 year period and the models that were used in this project for the same are as follows:

1. Linear Model: The linear model was used as the base model to assess the performance of the other, more advanced models and to keep track of the error scores. The scikit-learn's linear model module was imported and the linear model was trained by feeding to it the training data and validated using the validation data set.

2. Decision Tree Regressor: This is a predictive model that initiates a decision which goes from observations to conclusions with respect to a target value. Decision trees are considered to mirror the decision making capabilities of humans quite closely when compared to other models. One of the drawbacks is that the model is non-robust, meaning that a small variation in the training data will result in huge errors while making predictions.

3. Random Forest Regressor: This model is a type of bootstrap aggregation as it is an ensemble of many decision trees built by re-sampling the training data quite frequently and those decision trees are voted on to form a consensus with respect to the predictions. Random forests are also capable of correcting the decision tree's tendency to overfit. Random forests average the deep decision trees reducing the high variance factor of the decision tree.

4. Gradient Boosting Random Forest Regressor: Gradient Boosting for tree learning models helps improve the quality of fit of each base learner. It is a technique that results in a prediction model as an ensemble of multiple weak prediction models. It generalises at each stage of boosting and allows for optimization of an arbitrary differentiable loss function. The significance of this model, apart from the gradient boosting technique remains the same as the previously discussed Random Forest Regressor model.

5. Deep Neural Network (DNN): DNN is basically an artificial neural network but with more hidden layers between the input and output layers. The one major advantage of DNN is that it is equipped with the correct mathematical manipulations to deal with inputs of both linear and non-linear nature. Deep Neural Nets are known for modeling complex non-linear relationships. The model creates a map of virtual neurons while assigning numerical values that are random called weights. These weights are multiplied with the inputs and returned as an output between the range of 0 and 1. Also, until the model can determine the correct mathematical manipulation, the algorithm continues to adjust weights to the point where it can recognize the patterns accurately.

6. Long-Short Term Memory Recurrent Neural Network: LSTM is a deep learning model following an artificial Recurrent Neural Network architecture. LSTM has feedback connections unlike Deep Neural Networks. A single LSTM unit is made up of one cell, an input and output gate and a forget gate. It is called the Long-Short Term Memory because the cell remembers values over a short interval period. This model was developed to deal with the vanishing gradient problem, a problem that is commonly encountered while training traditional recurrent neural networks.

3.4 Evaluation

The evaluation metrics considered for this project so as to interpret the performance of the models trained and predictions obtained after feeding the training and test sets to the various models are the R^2 Score, Root Mean Squared Error and Mean Absolute Error.

The metrics mentioned here in this section are calculated as such:

$$R^2Score = 1 - \frac{\sum(RegressionError)^2}{\sum(TotalError)^2} \tag{1}$$

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2} \tag{2}$$

$$MAE = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i| \tag{3}$$

where,

n is the total number of instances.
y_i is the Predicted Value.
x_i is the Actual Value.

4 Design Specification

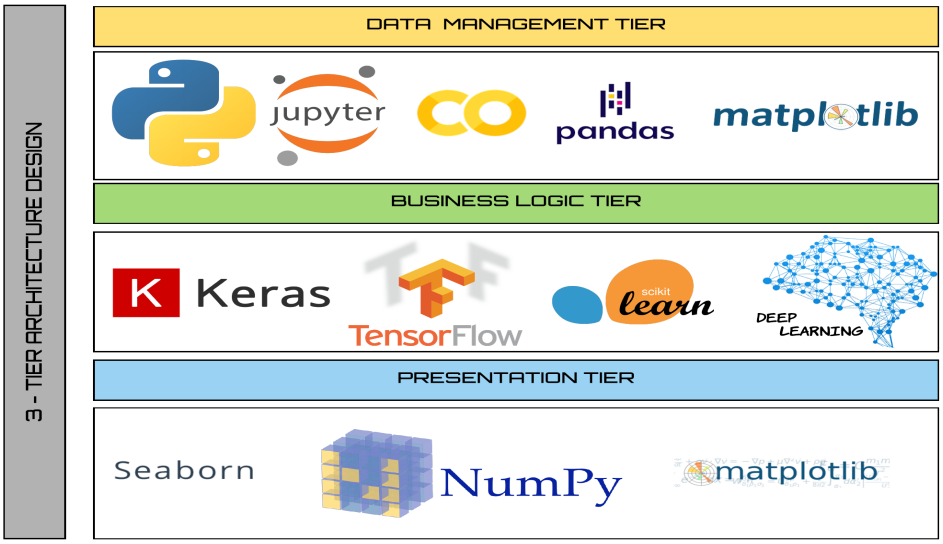

Figure 10: 3-Tier Architecture Design

Figure 10 above, represents the 3-tier architecture design that was employed to implement this research project. The following provides a brief overview of each of the tier that constitutes the 3-tier architecture.

The data management tier, as the name suggests, dealt with managing and manipulating data to make sure that the data is prepared according to the requirements of

the Business Logic Tier while the Business Logic Tier is the one meant to deal with the stakeholder related part of the research project, wherein, the correct environment was set up to design and develop the solution that could be of great value to the stakeholders. The Presentation tier constitutes the final stage of the architecture wherein findings of this research project are visualized and presented.

5 Implementation, Evaluation and Results

In this chapter, the focus is on the objectives 2, 3, 4 and 5 as listed in table 1. The aforementioned objectives are Modeling, Custom Loss Function, Prediction Stacking and Evaluation respectively. This chapter of the report shall provide in detail, information about how the model was implemented, how experimental results were evaluated using the chosen metrics, and what the results observed indicate and how they are of value to the research work. Also, the results obtained by implementing each of the models below, help answer the research question which is the aim of this research project.

The evaluation metrics that were used to assess the implemented models and the experimental results produced by those models were R^2, RMSE and MAE scores as mentioned in section 3.4. The R^2 score is a statistic that signifies the goodness of fit of a model and an R^2 score close to 1 is preferred because a score of 1 indicates that predictions fit the data perfectly. RMSE gives the root value of the mean of squared differences between predicted and actual data points, while MAE gives the average value of the absolute difference between predicted data points and the actual data points. It is preferred to observe a value as close to zero as possible for both RMSE and MAE evaluation metrics.

5.1 Implementation, Evaluation and Results of Linear Model

The linear regressor model was chosen as the base model and was implemented using the sklearn module's linear library. Prior to its implementation, the seed was set to 42 using the numpy library's random function attribute. The linear model was initialized and trained using the training data created during the data preprocessing stage and validated using the validation data. The feature testing data was fed to the trained linear model to predict stock index prices based on its training. The linear model's predictions are evaluated and the metrics are calculated. Accordingly, the base model was able to achieve an R^2 score of 0.8220, validation score of 0.5953, RMSE score of 0.1062 and an MAE score of 0.0952. Plotting the predicted stock index closing prices against the actual index closing prices, it was observed that the base model was able to mimic the trend of the stock index closing price movement accurately as depicted by figure 11 below.

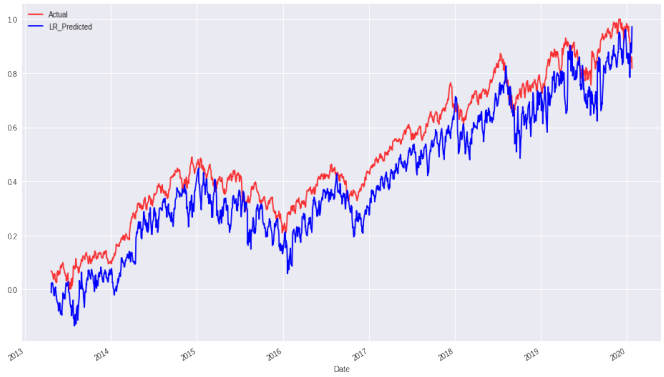

Figure 11: Predicted Future Index Price v/s Actual Index Price

5.2 Implementation, Evaluation and Results of Decision Tree Regressor

The decision tree regressor model was implemented and initialized using the tree module of the sklearn machine learning module. The model was trained and validated using the train and validation datasets prepared earlier. In order to determine the best depth value for optimal performance of the model, a list of depth values were iterated over while keeping track of the training and validation scores for each of them. The max depth attribute set to 5 was observed to have the highest validation score, hence, the max depth attribute was set to 5 and model was initialized again. Predictions were made by the model which were evaluated based on the chosen metrics. The R^2, RMSE and MAE scores were observed to be 0.9104, 0.0753 and 0.0596 respectively. Figure 12 represents the predicted stock index closing prices plotted alongside the actual prices and it can be observed that the model was not very successful at accurately predicting long-term future stock index prices.

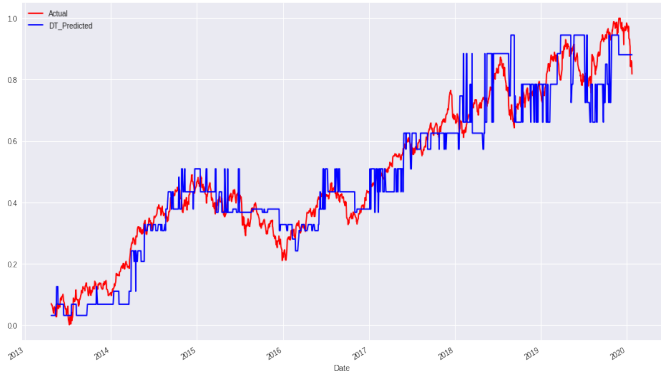

Figure 12: Predicted Future Index Price v/s Actual Index Price

5.3 Implementation, Evaluation and Results of Random Forest Regressor

Random forest regressor model was also implemented using the tree module under the sklearn machine learning module. The model was initially initialized with 200 estimators

upon which it was trained and validated to get an estimate on how well the model was doing during the training phase. With 200 estimators, the model achieved a training and validation score of 0.99 and 0.5406 respectively. A parameter grid search was implemented to determine the best possible parameters to initialize the model with. The parameter grid search recommended setting random state to 42, number of estimators to 150, max features to 15 and max depth to 5. The model was rebuilt and deployed with these parameters and the feature test set was fed to this trained model. On evaluating the predictions produced by this model, the R^2, RMSE and MAE score were observed to be 0.9369, 0.0632 and 0.0508 respectively. The feature importances attribute of the random forest regressor model provided a good understanding of how important each of the 15 features were in forecasting the future stock index prices as depicted in figure 13. The predicted future prices when compared to the actual prices were observed to be quite accurate as seen in figure 14.

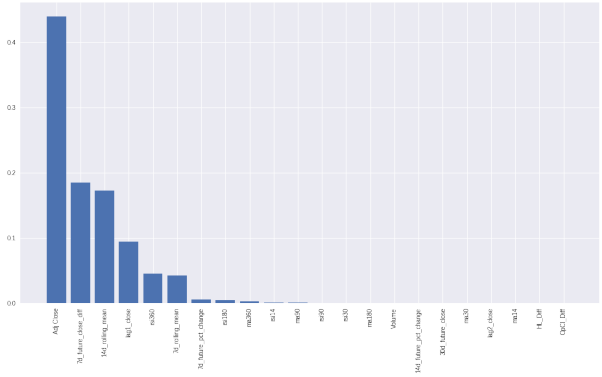

Figure 13: RFR Feature Importance Plot

Figure 14: Predicted Future Price v/s Actual

5.4 Implementation, Evaluation and Results of Gradient Boosting Random Forest Regressor (GBR)

Similar to Random forest regressor model, the GBR was implemented using the tree module under the sklearn machine learning module. The model was initialized with the similar parameters as defined for the random forest model in addition to learning rate of 0.01 and subsample of 0.50. The GBR model validation score was noted to be 0.5655 while the R^2, RMSE and MAE scores were observed to be 0.8722, 0.090 and 0.0698 respectively. This was a clear indication that gradient boosting applied to random forest regressor did not perform nearly as well as the random forest model, even though it was implemented by specifying the same parameters. The feature importances attribute of the GBR model was used to determine how important each of the 15 features were in forecasting the future stock index prices as depicted in figure 15. The predicted future prices when compared to the actual prices were observed to be quite accurate initially but, by the end the predictions made seemed nowhere near the actual prices, as depicted by figure 16.

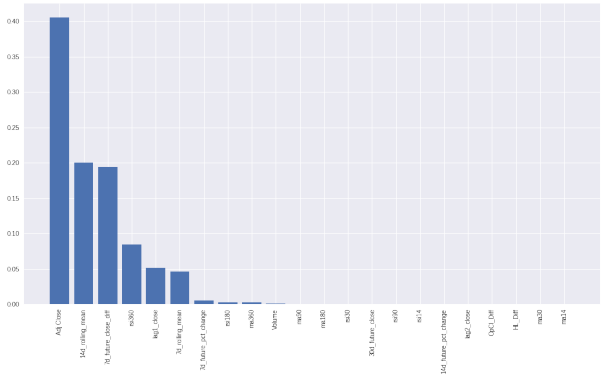

Figure 15: GBR Feature Importance Plot

Figure 16: Predicted Future Price v/s Actual

5.5 Implementation, Evaluation and Results of Deep Neural Network (DNN)

The Deep Neural Network was implemented as a sequential model using the Keras deep learning backend environment. The input, hidden and output layers were all initialized as Dense layers. The input dense layer was initialized with 16 neurons, he_uniform kernel initializer and the activation function used was the Rectified Linear Unit (relu). The two hidden dense layers that followed, were initialized with 8 neurons and 4 neurons respectively and the activation function used was rectified linear unit while the output dense layer was initialized with a single neuron and with the linear activation function. The optimizer chosen to compile the model was Adam and the loss function incorporated was the mean squared error. The compiled model was then trained with a batch size of 1100 and 100 epochs. The trained model was assessed using the training and validation loss parameters, based on which the batch size and epochs count were specified. Change in both training and validation loss for each epoch were kept track of with the help of a train versus validation loss graph as seen in figure 17.

Evaluation of the model was carried out using the metrics specified in section 3.4. But first, it was an important task to assess the trained model by observing the train versus validation loss graph. Based on the observations made from figure 17 below, it was noted that the training and validation loss, both, started off high, but, as the model went through further training, at 100 epochs, the training loss and validation loss were observed to be 0.0028 and 0.0032 respectively. As for the goodness of fit of the deep neural network model, the R^2 score was found to be 0.9488 while the RMS and MA error scores were observed to be 0.0569 and 0.0459 respectively.

The predictions of future stock index prices made by the deep neural network model was visualized alongside the real data points as seen in figure 18. Even though slight deviations can be observed in the forecasted stock index prices, this model can be considered to have performed really well in terms of learning from the trend in the training data.

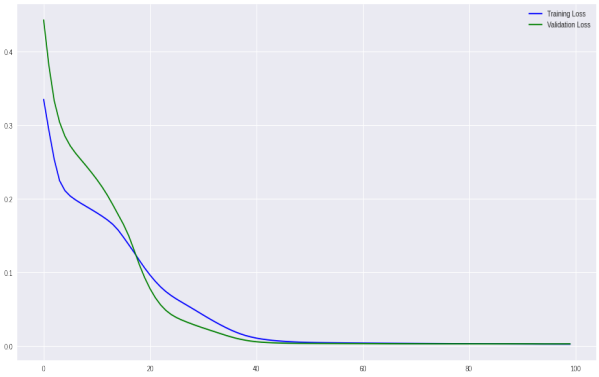

Figure 17: Train-Validation Loss

Figure 18: Predicted Future Price v/s Actual

5.5.1 Deep Neural Network with Dropout

This Deep Neural Network was implemented as a sequential model similar to the previous model using the Keras deep learning backend environment. The input, hidden and output layers were all initialized as Dense layers. Along with these layers, a dropout layer was introduced between the second and third hidden layers. The input dense layer was initialized with 16 neurons, he_uniform kernel initializer and the activation function used was the Rectified Linear Unit (relu). The two hidden dense layers were initialized with 8 neurons and 4 neurons respectively, while the dropout layer was initialized with a value of 0.1 and the activation function used was rectified linear unit while the output dense layer was initialized with a single neuron and with the linear activation function. The optimizer chosen to compile the model was Adam and the loss function incorporated was the mean squared error. The compiled model was then trained with a batch size of 1100 and 100 epochs similar to the previously implemented DNN model. The trained model was assessed using the training and validation loss parameters, based on which the batch size and epochs count were specified. Change in both training and validation loss for each epoch were kept track of with the help of a train versus validation loss graph as seen in figure 19.

Evaluation of the model was carried out using the metrics specified in section 3.4, before which, the trained model's effectiveness was assessed by observing the train versus validation loss graph. Based on the observations made from figure 19 below, it was noted that the training loss and validation loss were high at nearly the same values and as the model went through further training, at 100 epochs, the training loss and validation loss were observed to be 0.0059 and 0.0183 respectively. While the training loss was observed to have decreased smoothly to a lower error score, the validation loss was observed to increase and decrease frequently resulting in an unstable descent as seen in figure 19. As for the goodness of fit of the deep neural network model with dropout, the R^2 score was found to be 0.7121 while the RMS and MA error scores were observed to be 0.1351 and 0.1193 respectively.

The predicted future stock index prices obtained by training the deep neural network model with dropout was visualized alongside the real data points as seen in figure 20. As the R^2 score indicated, the graph of predicted stock index prices is very poor when compared to the graph seen in figure 18. As the model made an effort to predict prices farther in time, it was observed that the predictions were far off from the real data points. Hence, it was clear that the introduction of a dropout layer does not improve the performance of the deep neural network model.

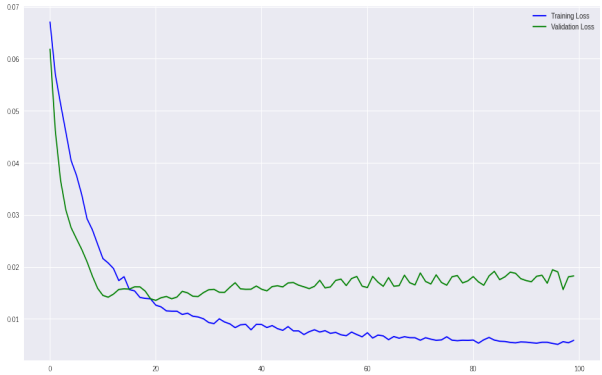

Figure 19: Train-Validation Loss

Figure 20: Predicted Future Price v/s Actual

5.5.2 DNN with Custom Loss Function

The Custom Loss Function (CLF), as the name suggests, is a custom defined loss function specific to this problem scenario. The CLF was defined to experiment with the implemented models in order to observe whether or not the deep learning models' performance would improve. Algorithm of logic for the Custom Loss Function was defined as follows:

$penalty \leftarrow 180$
if $(y\_true * y\_pred) \leq 0$ **then**
  $loss \leftarrow penalty * (y\_true * y\_pred)^2$
else
  $loss \leftarrow (y\_true * y\_pred)^2$
end if

The Deep Neural Network was implemented as a sequential model. All the were initialized as Dense layers. The input dense layer was initialized with specifications similar to deep neural network as mentioned in 5.5. The two hidden dense layers following the input layer were initialized with 8 neurons and 4 neurons respectively and the activation function used was rectified linear unit while the output dense layer was initialized with a single neuron and with the linear activation function. The optimizer chosen to compile the model was Adam and the loss function incorporated was the **Custom Loss Function**. The compiled model was then trained with a batch size of 1024 and 96 epochs. The trained model was assessed using the training and validation loss parameters, based on which the batch size and epochs count were specified. Change in both training and validation loss for each epoch were kept track of with the help of a train versus validation loss graph as seen in figure 21.

Evaluation of this model, the same as other models was carried out using the metrics specified in section 3.4. The assessment of the trained model by observing the train versus validation loss graph closely. Based on the observations made from figure 21, it was clear that the training and validation loss, both see a good amount of decrease in the loss incurred, and at 96 epochs, the recorded training loss and validation loss were 0.0019 and 0.0062 respectively. Goodness of fit of the deep neural network model was assessed by taking note of metrics calculated and as such, the R^2 score was found to be 0.9257 while the RMS and MA error scores were observed to be 0.0686 and 0.0566 respectively. Although, the loss incurred during model training was lower than the DNN model initialized with MSE loss function, the validation loss was comparatively higher.

All things considered, the DNN model implemented with CLF loss function performed better than the base model and almost as good as the DNN model.

The future stock index prices predicted by this model was visualized against the actual prices as seen in figure 22 and from this plot, it can be noted that the model fits close to actual prices initially and deviates later on.

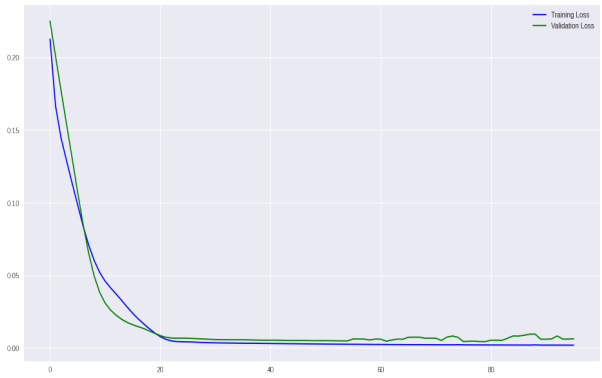

Figure 21: Train-Validation Loss

Figure 22: Predicted Future Price v/s Actual

5.6 Implementation, Evaluation and Results of LSTM Recurrent Neural Network

The LSTM Recurrent Neural Network was also implemented as a sequential model using the Keras deep learning backend environment. The model was initialized with an LSTM layer first, with 32 LSTM units, activation function set to rectified linear unit (relu), kernel intializer chosen to be 'he_uniform' while specifying the input shape reflecting the features and targets. The hidden and output layers were all initialized as Dense layers. The first dense layer was initialized with 16 neurons, and the activation function used was the Rectified Linear Unit (relu). Similarly, the two hidden dense layers that followed, were initialized with 8 neurons and 4 neurons respectively and the activation function used was rectified linear unit while the output dense layer was initialized with a single neuron and with the linear activation function. The optimizer chosen to compile the model was Adam and the loss function incorporated was the mean squared error function. The compiled model was then trained with a batch size of 850 and 68 epochs. The trained model was assessed using the training and validation loss parameters, based on which the batch size and epochs were decided on. Change in both training and validation loss for each epoch were kept track of with the help of a train versus validation loss graph as seen in figure 23.

Evaluation metrics specified in section 3.4 were crucial in assessing the predictive capability of the trained model. By observing the train versus validation loss graph, from the figure 23 below, it was noted that the training and validation loss, both, started off high, similar to the training and validation losses as observed in the case of deep neural network, but, as the model finished training, at 68 epochs, the training loss and validation loss were observed to be 0.0021 and 0.0032 respectively. Comparing this to the best performing model so far, the deep neural network model, training loss was evidently lesser in LSTM RNN model's case while validation loss remained the same, clearly indicating that the LSTM model was able to achieve better results than the DNN model with a batch size of 850 at 68 epochs. As for the goodness of fit of this LSTM

Recurrent Neural Network model, the R^2 score was found to be 0.9492 while the RMS and MA error scores were observed to be 0.0567 and 0.0451 respectively.

The plot of predicted future stock index prices versus actual stock index prices provided by the figure 24 shows that the trained LSTM RNN model fits the actual data much better than the models previously mentioned in this chapter of the report. Despite the goodness of fit achieved through this model, it can be observed that there are sudden spikes in predicted values throughout the timeline, more so with respect to the earlier years. Nevertheless, based on the RMSE and MAE scores, it was evident that the error in predictions made with respect to the actual data, was much lesser for the LSTM model as compared to the error scores observed for the rest of the models before, making the LSTM RNN model, the best performing deep learning model so far.

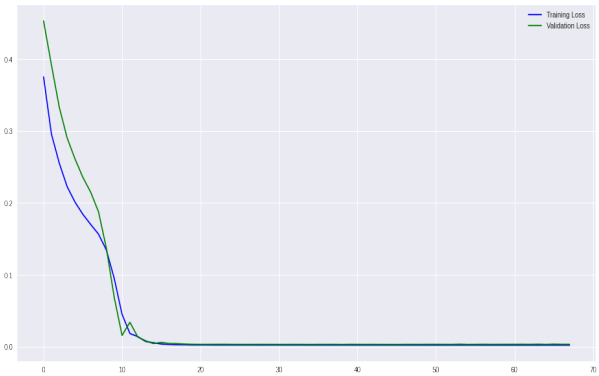

Figure 23: Train-Validation Loss

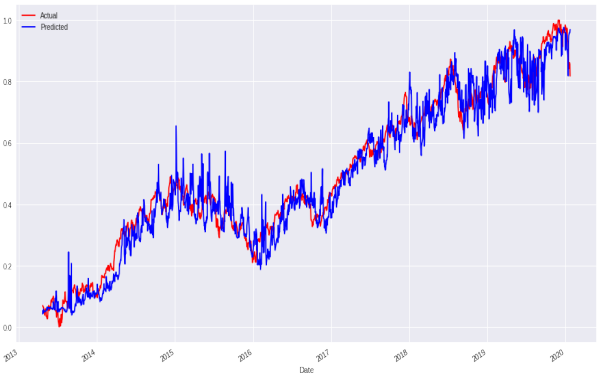

Figure 24: Predicted Future Price v/s Actual

5.6.1 LSTM RNN with Custom Loss Function

The Custom Loss Function (CLF) used for the LSTM RNN model is the same as the CLF used for the DNN model. The algorithm of logic defined for CLF is described in subsection 5.5.2. The LSTM model in this case was implemented as a sequential model with 16 LSTM units for the input layer with relu activation function. The hidden layers and the output layer were implemented as Dense layers with 8,4 and 1 neurons respectively. Activation function attribute of the output layer was set to linear. Adam optimizer and CLF loss function were used to compile the model. To train the model, batch size and epochs were set to 1100 and 100 respectively. The training and validation losses incurred during model training is depicted in figure 25.

Evaluation of the model showed that training loss from the start was quite less and did not improve further while the validation loss increased by the end of model training. Thus, the model trained with CLF loss function was observed to be quite poor with respect to performance. The plot of predicted stock index prices and the actual prices as seen in figure 26 reflects the same. The predicted prices were nowhere near the actual prices. The R^2, RMSE and MAE scores were found to be -4.8552, 0.6095 and 0.5441. A negative R^2 value indicates that the predictions made by the model were below the mean value of the actual data points. Thus, it was safe to conclude that the CLF, despite doing well with the DNN model, was not compatible with the LSTM RNN model.

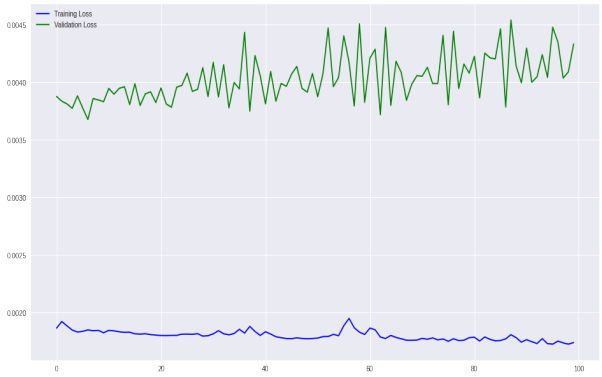

Figure 25: Train-Validation Loss

Figure 26: Predicted Future Price v/s Actual

Table 2 lists the results obtained for all the implemented models with respect to each of the evaluation metric highlighting the best performing model.

Table 2: Model Evaluation Scores

Model	R^2 Score	RMSE	MAE
Linear	0.82201	0.1062	0.0952
Decision Tree Regressor	0.9104	0.07538	0.0596
Random Forest Regressor	0.9369	0.0632	0.0508
Gradient Boosted RFR	0.87225	0.09	0.069
DNN	0.9488	0.0569	0.0459
DNN-Dropout	0.7121	0.1351	0.1193
CLF-DNN	0.9257	0.06861	0.0566
LSTM	0.9492	0.0567	0.0451
CLF-LSTM	-4.885	0.6095	0.5441

5.7 Prediction Stacking

5.7.1 Machine and Deep Learning Model Stacking

Machine learning models such as Linear Regressor, Random Forest, Gradient Boosting Random Forest were combined with deep learning models such as the Deep Neural Network and LSTM Recurrent Neural Network by using the stacking method to assess the ability of prediction model stacks to improve the stock index price predictions made while reducing the error rate further. In this project, prediction stacks, namely, LR-DNN, LR-LSTM, RF-DNN, RF-LSTM, GBR-DNN and GRB-LSTM were implemented and their performance based on the evaluation metrics, R^2, RMSE and MAE score are listed in table 3 and further discussed upon in chapter 6 of the report.

5.7.2 DLSTM (DNN-LSTM) Stack

DLSTM Stack or the DNN-LSTM stack was implemented as a combination of the Deep Neural Network model and the LSTM RNN model by stacking the predictive capabilities of the two models together to optimize the predictions made by each of the above mentioned models individually. Evaluation of the DLSTM model developed as such, showed

that the predicted stock index prices when compared with the actual prices resulted in R^2, RMSE and MAE scores of 0.9582, 0.0514 and 0.0417 respectively.

5.7.3 RF-DLSTM Stack

The RF-DLSTM stack was implemented as a combination of the Random Forest Regressor, Deep Neural Network and the LSTM RNN models by using the stacking approach to optimize the predictions made by RF-DNN, RF-LSTM and DLSTM stacks respectively. Evaluation of the RF-DLSTM model provided evidence of optimization of the models' individual performance while significantly bringing down the error scores observed prior to the application of the stacking approach. RF-DLSTM prediction stack was deemed the best performing model stack in terms of long-term forecasting with R^2, RMSE and MAE scores of 0.9660, 0.0503 and 0.0408 respectively. Accordingly, figure 27 below shows the improvement in fitting of predicted prices curve against the actual prices curve. From figure 27, it is evident that the line representing the predicted stock index prices is a lot smoother with respect to the line representing the actual prices, more so when compared to the DNN and LSTM models individually.

Figure 27: Histogram of Index Closing Price Percentage Change Features

Table 3 below, lists the results obtained with respect to each of the evaluation metric chosen, for each of the model stack developed while highlighting the best among the prediction model stacks.

Table 3: Prediction Stack Evaluation Scores

Stack	R^2 Score	RMSE	MAE
LR-DNN	0.9175	0.0723	0.0625
LR-LSTM	0.9181	0.0720	0.0611
RF-DNN	0.9563	0.0540	0.0438
RF-LSTM	0.9567	0.0523	0.0422
GBR-DNN	0.9318	0.0657	0.0520
GBR-LSTM	0.9385	0.0624	0.0494
DLSTM	0.9582	0.0514	0.0417
RF-DLSTM	0.9660	0.0503	0.0408

6 Discussion

The evaluation of machine learning and deep learning models and the results thus obtained showed that the LSTM RNN model was the best performing model among the nine models implemented, in terms of making predictions, while, the LSTM RNN model trained with the custom loss function was observed to be the worst among all the models implemented with respect to performance. The CLF-LSTM model's performance was far worse than the base model, which is the linear model. The Random Forest regressor and the DNN model were observed to have performed quite well as the calculated metric score showed the above mentioned models' predictive capabilities to be quite close to that of the LSTM RNN model. Furthermore, it was observed that the custom loss function does not optimize training accuracy of the LSTM RNN model while, it was observed to be quite effective in the case of DNN model. The results obtained from the CLF-DNN model indicated that the model, although performed quite as well as the DNN model implemented without dropout and trained with the MSE loss function, the CLF-DNN model was not able to improve upon the predictive capabilities of the DNN model in turn being unable to minimize the margin of error any further. The findings of this research project concerning the prediction model stacks indicated that the prediction stack of random forest, RF-DNN and RF-LSTM models, named RF-DLSTM, was the best performing model stack implemented with an R^2 score of 0.966, RMSE score of 0.0503 and 0.0408 as compared to the best individual model implemented, i.e, LSTM RNN model which achieved an R^2 score of 0.9492, 0.0567, 0.0451. When compared with related work done in the past, the proposed prediction model stacking approach was observed to have helped accomplish splendid results as there are very less research work done on long-term forecasting, and also, previous work as published by Sharma and Juneja, document the evaluation metrics for their chosen models with respect to forecasting stock prices 30 days ahead of time, which were observed to have achieved an MSE score of 16565.96 which would translate to an RMSE score of 128.708 while the RF-DLSTM model stack implemented using proposed model stacking approach, observed an RMSE score 0.0503.

7 Conclusion and Future Work

This research project focused on answering the research questions as mentioned in subsection 1.1, and as such the results observed implied the following regarding the main research question, which was to determine to what extent the BSE SENSEX stock index can be predicted over a long time period by using machine and deep learning models. Results obtained by evaluating the machine and deep learning model that were implemented, showed that the BSE stock index can be predicted to a great extent over a long period of time. Accordingly, the LSTM RNN model predicts the future stock index prices quite accurately and it was observed to have achieved an R^2 score of 0.9492 with lesser error score. The RF-DLSTM prediction model stack, implemented using the proposed prediction model stacking approach, optimized the predicted future stock index prices even further and was observed to have outperformed the LSTM RNN model in terms of predictive capabilities, with an R^2 score of 0.966, RMSE score of 0.0503 and MAE score of 0.0408. As for the sub-research question which was aimed at determining whether or not a custom defined loss function can minimize the margin of error for deep learning models, observations were made that indicated the compatibility of the CLF function with DNN model and complete incompatibility with the LSTM RNN model. Results

showed that the CLF-DNN model performed quite well and performance was close to that of the initial DNN model, but failed to improve upon this performance any further. Meanwhile, the CLF-LSTM model was observed to have performed worse than the base model, with a negative R^2 score of -4.885. With this, it was evident that a custom loss function, unlike the standard loss functions need to be rewritten for every single deep learning model specifically to deal with incompatibility issues.

In the future, this research project can be further improved upon by predicting stock prices over a long period of time, of companies that fall under the BSE SENSEX Stock Index, forecasted over the same period of time as the BSE stock index to determine the potential returns on those stocks using the Modern Portfolio Theory (MPT). Also, many more complex features can be engineered to optimize the model's training process. User applications can be developed for stakeholders such as investors, traders etc. to interactively select the time period during which they would like to invest. Furthermore, cloud TPUs can be employed to train and combine the predictive capabilities of more complex deep learning models to make the models more precise with respect to future stock index price forecasting.

References

Azevedo, A. I. R. L. and Santos, M. F. (2008). Kdd, semma and crisp-dm: a parallel overview, *IADS-DM* .

Beyaz, E., Tekiner, F., Zeng, X.-j. and Keane, J. (2018). Comparing technical and fundamental indicators in stock price forecasting, *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, IEEE, pp. 1607–1613.

Chen, J. (2010). Svm application of financial time series forecasting using empirical technical indicators, *2010 International Conference on Information, Networking and Automation (ICINA)*, Vol. 1, IEEE, pp. V1–77.

Dai, H., Zhang, Y. and Wang, D. (2011). Price prediction of stock index futures based on svm, *2011 Fourth International Conference on Business Intelligence and Financial Engineering*, IEEE, pp. 54–57.

Dechow, P. M., Hutton, A. P., Meulbroek, L. and Sloan, R. G. (2001). Short-sellers, fundamental analysis, and stock returns, *Journal of financial Economics* **61**(1): 77–106.

Deng, S., Mitsubuchi, T., Shioda, K., Shimada, T. and Sakurai, A. (2011). Combining technical analysis with sentiment analysis for stock price prediction, *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, IEEE, pp. 800–807.

Di Persio, L. and Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications, *International journal of circuits, systems and signal processing* **10**(2016): 403–413.

Gao, T., Li, X., Chai, Y. and Tang, Y. (2016). Deep learning with stock indicators and two-dimensional principal component analysis for closing price prediction system, *2016 7th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, IEEE, pp. 166–169.

Hargreaves, C. and Hao, Y. (2012). Does the use of technical & fundamental analysis improve stock choice?: A data mining approach applied to the australian stock market, *2012 International Conference on Statistics in Science, Business and Engineering (ICSSBE)*, IEEE, pp. 1–6.

Iuhasz, G., Tirea, M. and Negru, V. (2012). Neural network predictions of stock price fluctuations, *2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, IEEE, pp. 505–512.

Khare, K., Darekar, O., Gupta, P. and Attar, V. (2017). Short term stock price prediction using deep learning, *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, IEEE, pp. 482–486.

Kumar, S. and Ningombam, D. (2018). Short-term forecasting of stock prices using long short term memory, *2018 International Conference on Information Technology (ICIT)*, IEEE, pp. 182–186.

Lee, J. W. (2001). Stock price prediction using reinforcement learning, *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, Vol. 1, IEEE, pp. 690–695.

Pasupulety, U., Anees, A. A., Anmol, S. and Mohan, B. R. (2019). Predicting stock prices using ensemble learning and sentiment analysis, *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, IEEE, pp. 215–222.

Ravi, V., Pradeepkumar, D. and Deb, K. (2017). Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms, *Swarm and Evolutionary Computation* **36**: 136–149.

Sharma, N. and Juneja, A. (2017). Combining of random forest estimates using lsboost for stock market index prediction, *2017 2nd International Conference for Convergence in Technology (I2CT)*, IEEE, pp. 1199–1202.

Tsai, C. and Wang, S. (2009). Stock price forecasting by hybrid machine learning techniques, *Proceedings of the international multiconference of engineers and computer scientists*, Vol. 1, p. 60.

Udagawa, Y. (2018). Predicting stock price trend using candlestick chart blending technique, *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, pp. 4162–4168.

Wirth, R. and Hipp, J. (2000). Crisp-dm: Towards a standard process model for data mining, *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*, Citeseer, pp. 29–39.

Zarkias, K. S., Passalis, N., Tsantekidis, A. and Tefas, A. (2019). Deep reinforcement learning for financial trading using price trailing, *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, pp. 3067–3071.

Zhao, Q.-y., Zhao, X. and Duan, F. (2009). Prediction model of stock prices based on correlative analysis and neural networks, *2009 Second International Conference on Information and Computing Science*, Vol. 3, IEEE, pp. 189–192.