Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Investigation of Forecasting Model Designed to Evaluate Financial Trends

Master's Final Degree Project

Mykolas Benjaminas Meškas

Project author

Prof. Dr. Vidas Raudonis

Supervisor

Kaunas, 2025

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Investigation of Forecasting Model Designed to Evaluate Financial Trends

Master's Final Degree Project

Control Technologies (6211EX014)

Mykolas Benjaminas Meškas
Project author

Prof. Dr. Vidas Raudonis
Supervisor

Asist. Dr. Vygandas Vaitkus
Reviewer

Kaunas, 2025

Kaunas University of Technology

Faculty of Electrical and Electronics Engineering

Mykolas Benjaminas Meškas

Investigation of Forecasting Model Designed to Evaluate Financial Trends

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Mykolas Benjaminas Meškas

Confirmed electronically

Meškas Mykolas Benjaminas. Investigation of Forecasting Model Designed to Evaluate Financial Trends. Master's Final Degree Project / supervisor Prof. Dr. Vidas Raudonis; Faculty of Electrical and Electronics Engineering, Kaunas University of Technology.

Study field and area (study field group): Electronics engineering (Engineering sciences).

Keywords: Stock market, chart patterns, neural network, GA-LSTM-CNN model.

Kaunas, 2025. p. 53.

Summary

In this thesis, chart pattern influence on the GA-LSTM-CNN model is investigated. Chart patterns are a branch of technical analysis that uses stock price movement shapes or candlesticks to predict future price movement. The GA-LSTM-CNN model was selected due to its combination of several neural network algorithms that achieve self-structuring, temporal and spatial pattern recognition. As it is not clear which features should be chosen for stock trading due to numerous trading strategies, a genetic algorithm provides an automatic search for features that best fit the use case. LSTM and CNN combinations provide the model with temporal and spatial pattern recognition capabilities, respectively. That is needed in stock trading, as patterns exist between features and in time for the same feature. Before the preliminary tests were conducted, chart patterns were selected from financial data using analytical rules outlined in the analysed research paper. Searching of chart patterns was done backwards to not give the model any knowledge of financial events in the future. Preliminary tests showed that the model selected chart patterns as one of the features. The results were similar to those of the model without chart patterns. Final tests revealed that chart patterns do not have a significant impact on individual model performance, but could be useful in an ensemble scenario. This thesis is separated into seven sections. The first section is a brief introduction to the problem, object of experiments and experimental steps taken. The second section is state-of-the-art research, where 29 sources are analysed to get a comprehensive overview of automated stock market trading. The third section contains explanations of used methods, models and metrics. The fourth section has descriptions of the training and testing environments, data used for experiments. The fifth section has comparison of model testing results. The sixth section is a discussion on problems encountered in the experiments, successes and future work. Finally, conclusion is given on model performances in the last section. In total, this work contains 41 figures and 53 pages.

Meškas Mykolas Benjaminas. Finansinių tendencijų vertinimui skirto prognozavimo modelio tyrimas. Magistro baigiamasis projektas / prof. dr. Vidas Raudonis; Kauno technologijos universitetas, Elektros ir elektronikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Elektronikos inžinerija (Inžinerijos mokslai).

Reikšminiai žodžiai: Akcijų birža, kainų formuotės, neuroniniai tinklai, GA-LSTM-CNN modelis.

Kaunas, 2025. 53 p.

Santrauka

Šiame tyrime atskleidžiama kainų formuočių įtaka pasirinktam GA-LSTM-CNN modeliui. Kainų formuotės yra techninės analizės šaka bandanti prognozuoti akcijų kainas ateityje pagal akcijų kainos pokyčių sudaromas formas arba žvakes. GA-LSTM-CNN modelis buvo pasirinktas, nes jis yra kelių NT algoritmų kombinacija, kuris geba persistruktūrizuoti bei atpažinti laikines ir erdvines tendencijas. Nors nėra aišku, kurias savybes reikėtų pasirinkti akcijų biržai dėl didelio kiekio prekiavimo akcijomis strategijų, genetinis algoritmas atlieka paiešką savybių, kurios labiausiai atitinka situaciją. LSTM ir CNN kombinacija atitinkamai suteikia laiko ir erdvės tendencijų atpažinimo galimybes ir yra naudinga akcijų biržos kontekste, nes tendencijos egzistuoja ne tik tarp bruožų, bet ir tarp to paties bruožo laike. Prieš pradedant preliminarius bandymus, turėjo būti surinkti kainų formuotės pavyzdžiai iš finansinės duomenų bazės, naudojant analitines taisykles, kurios yra nurodytos analizuotame tyrime. Kainų formuočių paieška buvo atliekama atbuline tvarka, kad modelis neturėtų jokios informacijos apie būsimus finansinius įvykius. Preliminarių bandymų rezultatai parodė, kad šis modelis, iš tikrųjų, pasirenka kainų formuotes kaip vieną iš savo bruožų, kurį naudoja kaip įvestį likusiam modeliui ir šio modelio rezultatai buvo panašūs į modelio nenaudojančio kainų formuočių rezultatus. Galutiniai testai parodė, kad kainų formuotės neturi žymios įtakos individualaus modelio darbui, tačiau gali padėti sujungus abu modelius į ansamblį. Šis tyrimas yra išskaidytas į septynias pagrindines dalis. Pirma dalis yra įžanga, kurioje aprašoma problematika, tyrimo objektas ir atlikti žingsniai tikslui pasiekti. Antra dalis yra literatūros apžvalga, kurioje išanalizuoti 29 šaltiniai, kad gauti bendrą naujausių tyrimų supratimą. Trečioje dalyje yra aprašomi naudoti metodai, modeliai ir kokybės metrikos. Ketvirtoje dalyje aprašomos eksperimentinės sąlygos, naudoti duomenys. Penktoje dalyje aprašomi modelio eksperimentiniai rezultatai. Šeštoje dalyje aptariamos problemos, pavykę ir ateities darbai. Septintoje dalyje daromos išvados apie modelio darbą. Viso šiame darbe yra 41 paveikslai, diagramos ir 53 lapai.

Table of contents

List of figures

Introduction

Stock markets all around the world provide a platform for people to trade shares or stocks of publicly traded companies. The initial price of shares is dictated by initial public offering but later is controlled by market forces of supply and demand. Each share represents a stake in the company and its profits. If the company is successful and maintains good public perception, the stock price increases. New York stock exchange alone reported market capitalization of 28.33 trillion United States dollars in the year 2024. For the experiments, financial information is downloaded from "Yahoo Finance" website about United States market as it is the most popular and has the most information. In this thesis, a combined GA-LSTM-CNN model was used to determine the impact of chart patterns on the quality of stock price predictions. Such automated systems are created in order to make market analysis and trading easier, more accurate, faster when compared to experts. This is necessary research as current state of the art does not have any data on chart patterns. Chart patterns are created by price movement, and they could be useful, easier tool in analysing market sentiment. There is always a portion of stock price movement that is not accounted by fundamental analysis only. That is why investor sentiment is a necessary component in predicting day to day price fluctuations. Chart patterns were found and classified with the use of analytical rules. If the chart pattern is beneficial for stock price prediction, the Genetic Algorithm (GA) passes it to the rest of the model and allows the pattern to influence the result. In case GA recognises the patterns to be harmful or neutral in terms of the final prediction, features are not selected for further analysis by the model. The real change in the output of the model happens when patterns improve prediction accuracy of the model. This way comparison between the output of two models is possible. The big challenges in this approach are analytical rule creation and correct structuring of the model. Analytical rules must be created in such a way that allows imperfect chart patterns to pass but be strict enough to not allow misclassification. There are rare examples of perfect chart patterns in real scenarios, therefore some adjustments to the ideal scenario rules must be made. The model has to be structured in a way to allow effective work for different type of layers. Data transformation inside the model between layers has to be taken into account. One model is trained with all available information and the other is trained without chart patterns. Therefore, the model with chart patterns must select patterns and they have to improve the results of the model in order for this method to be considered a success. To complete this research, the following tasks had to be done:

1. State of the art analysis of the given problem.
2. Pick suitable solutions that can tackle the given problem.
3. Prepare training and testing datasets. One dataset must contain chart patterns and the other must be without them.
4. Train and test created model on the two datasets.
5. Evaluate results with known efficiency metrics.

1. Related research and applications

Chart patterns are a well-known technical analysis tool concerned with the shape of daily price movements. There are examples of research papers attempting to use chart patterns that can be viewed in the form of line patterns, or K-line patterns also known as candlesticks. Most research has used K-line charts as they are easier to detect and utilize. In the paper [1] "Stock Trend Prediction Using Candlestick Charting and Ensemble Machine Learning Techniques With a Novelty Feature Engineering Scheme" took traditional candlestick charting techniques and combined them with the newest machine learning methods. Eight-trigram scheme was constructed based on Feng Shui theory. The scheme was an inter-day stock price movement representation. The model also used stock data and other technical indicators. The trigrams were constructed from high, low and close price data. The trigram had 8 patterns in total. The data also included volume data in the form of rate of change of volume. Finally, 21 technical indicators were selected to be put into the model as inputs. The parameters were 5 days for Moving Average (MA), Double Exponential Moving Average (DEMA), Kaufman Adaptive Moving Average (KAMA) and 10 days for Exponential Moving Average (EMA), Simple Moving Average (SMA), Average Directional Index (ADX), Absolute Price Oscillator (APO), Commodity Channel Index (CCI), Directional Movement Index (DX), Money Flow Index (MFI), Relative Strength Index (RSI), Average True Range (ATR), Normalized Average True Range (NATR). Data was made from 3455 stocks in the Chinese Stock Market in the period of 2000 to 2017. Data was split into 80% training and 20% testing sets. The model always assumed that trading starts at the closing price and ends at the next closing price. The ensemble used Logistic Regression (LR), Support Vector Machine (SVM), k-Nearest Neighbor (KNN), Random Forest (RF), Gradient Boosting Decision Tree (GBDT) and Long-Short Term Memory (LSTM). During model comparison RF and GBDT models had good predictive power, KNN showed only some of the predictive power, SVM took long to process big data and LSTM's abilities were poorly reflected in the experiment. Then, accuracy and F1 score were calculated and both were above 50%. While the model did return profit, after subtracting transaction costs the profits were significantly smaller. In this work analysis on the impact that eight-trigram scheme classes had on ensemble performance was never conducted. Another research paper using candlesticks titled [2] "Improving stock trading decisions based on pattern recognition using machine learning technology" had to predict stock price in the Chinese stock market using candlestick patterns. Different time windows, four machine learning methods and combinations of 11 different feature types were tested. Then a trading strategy was constructed and it was found that two-day candlestick pattern was the best at explaining next day's stock price. The models tested were LR, KNN, RF and Restricted Boltzmann Machine (RBM). Training set consisted of the entire Chinese stock market from 2000 to 2014 and then validation set from 2014 to 2020. The trading strategy involved testing of 1 to 10-day prediction windows with assumptions that when buying the stock its price will not fluctuate because of it and that the stock will be bought, sold at close price. Finally, annual return of 36.73% was reached with a 0.81 Sharpe ratio and an information ratio of 2.37. In the figure 1.1 is shown a graph that shows prediction accuracy of standard machine learning models and the ones that were trained to recognise candlestick patterns.

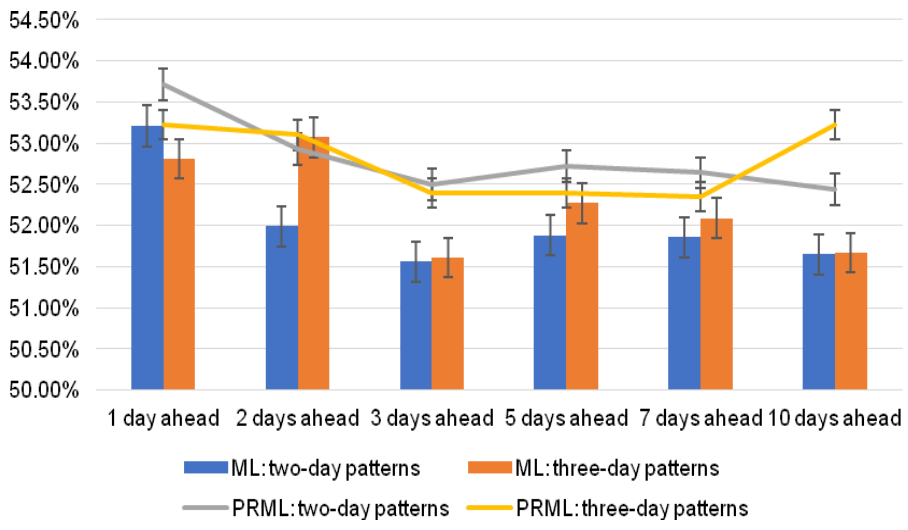


Fig. 1.1 Accuracy overview of PRML and pure machine learning models [2, p. 13]

As shown by the graph the two-day pattern was better in most cases over other methods, but it is not explained why PRLM three-day pattern performs better when predicting 10 days ahead. To use line graph patterns there needs to be a more complex identification algorithm that can remember and recognize shapes of stock price patterns. In the paper [3] titled "Feature extraction for chart pattern classification in financial time series" financial time series shape related features were analytically identified and the impact of the feature maps on the classification of chart patterns was investigated. 14 features were identified that could describe 41 chart patterns in technical analysis domain. Later, feature's effectiveness was evaluated by making them into classifying rules which were tested on financial data of 24 stocks/indices. Shape related features were hierarchical and upper-level features could be constructed using lower-level features. There were four total levels, level one included: slope between points, duration, amplitude and high/low touch; level two included trends; level three included slopes of trend lines and distance between special points and trend lines; Finally, level four included only composition of top trendline and bottom trend line slope magnitudes. During testing it was found that larger search windows resulted in less patterns identified which means that chart patterns that span over long durations (3-6 months) are rare. There were also fewer complex patterns as they require more rules to be satisfied. The system itself could effectively recognise the chart patterns, but the paper does not test the effectiveness of chart patterns in stock price predictions and the system itself was not adaptable to varying number of special points which was stated as a problem for future work.

One of the newer models called "Prophet" that was developed by "Facebook" is a simple model that can notice trends, shows the graphs of the trends it found and finally predicts on given time-series data with confidence intervals. This model takes in date and selected data which is then used by the model to predict future values and confidence interval. "Prophet" model uses sum to represent nonlinear trends that are divided into seasonal influences and holidays. The summation formula is shown in the equation 1.1.

$$y(t) = g(t) + s(t) + h(t) + t \qquad (1.1)$$

If: *y(t)* – model output; *g(t)* – nonperiodic time series variation in values; *s(t)* – periodic time series variation; *h(t)* – impact of holidays on the model's output; *t* – error for which the model cannot account.

As can be seen from the formula, this model does best when time-series data has strong seasonal or periodic influences, however according to "Facebook" their model was also able to tackle with issues regarding missing data or outliers. This was also the reason some researchers try to use "Prophet" model for predicting stock price. One of the use case examples in the stock market can be found in the paper [4] „Stock Index Probability Prediction using the FB Prophet Model". Here the "Prophet" where model was used to predict stock and option price 1-2 weeks into the future. The system had technical indicators and a special calculator made to detect moments of buying or selling. In the beginning user had to type in the stock ticker of their interest so that the model could get its data. The system then downloaded financial data of the chosen stock ticker from "Yahoo finance" page. The data used for the model was 3-year stock price historical data while the system gathered technical indicators for later use. In the graph 1.2 "Prophet" model's predictions and real stock price values are shown.

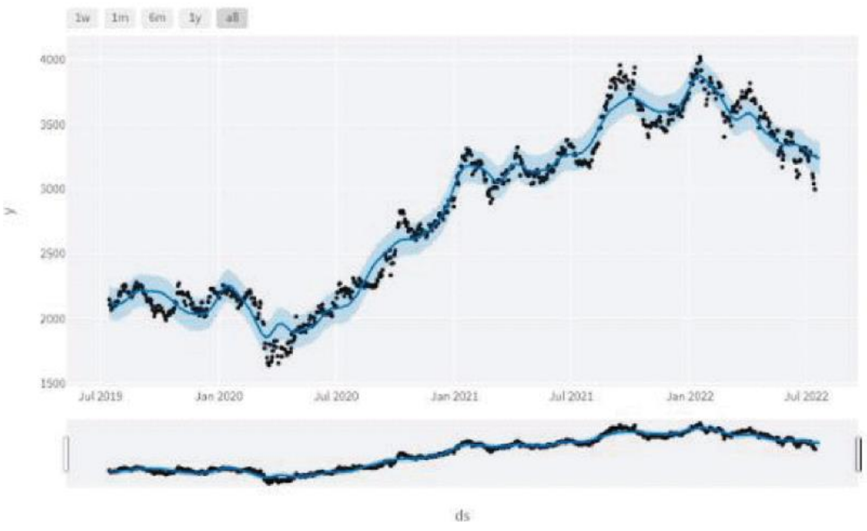
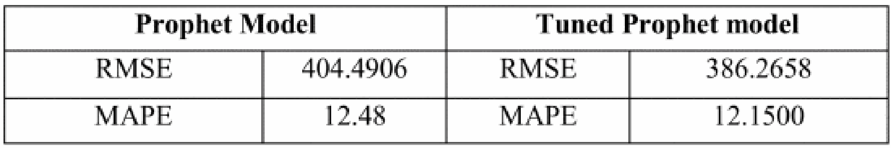


Fig. 1.2 Predicted and real stock price. Black – real stock price; Blue – „Prophet" model's predicted stock price and its confidence intervals [4, p. 4]

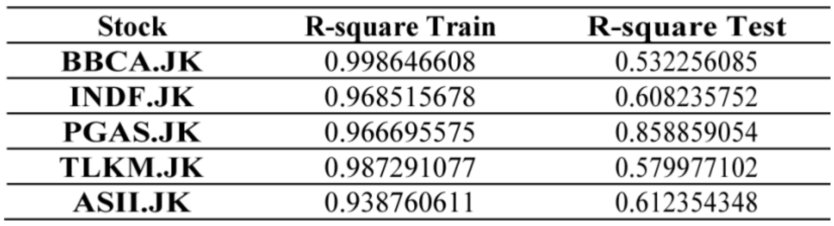
The final prediction was formulated by "PreVision" algorithm design for this experiment. It made buying/selling recommendation calculations which took in model's predictions and technical indicators. For technical indicators RSI, MACD, Stochastic Oscillators (SO) and "Super-trend" indicators were used. "PreVision" calculator algorithm gave an integer value from 0 to 70. When the algorithm gave a rating in between 0 and 30 the chosen action was not recommended, when it was between 30 and 50 the recommendation was to hold the stock, while the rating between 50 and 70 was a recommendation for the chosen action to be taken with the stock. During testing the accuracy of the strategy was 50-60%. The use of only one metric in this paper does not fully explain model performance. In the research article titled [5] „Prediction of Stock Prices using Prophet Model with Hyperparameters tuning" model performance was improved with the use of hyperparameter tuning. Evaluation of the improvements was done with the use of Mean Squared Error (MSE) and Mean Absolute Percent Error (MAPE) values. The two values were used due to them being differently affected by the error's amplitude. MSE is more sensitive to bigger errors as error is squared before the mean is taken. MAPE values show predicted value's absolute percent deviation from the real value on average thus making errors of different amplitudes affect the error with the same strength. For tuning of hyperparameters grid search method was used. Method's goal was to make model's predictions more accurate. The model used “TCS” stock data in order to train and test the accuracy of the model and the entire system was made in “Jupyter labs” environment. For historical data, the “NSE” database was used due to its size compared to other databases found and the database was updated every day. Obtained unchanged “Prophet” model’s and changed model’s prediction evaluation metrics MSE and MAPE are shown in the figure 1.3.



Prophet Model		Tuned Prophet model	
RMSE	404.4906	RMSE	386.2658
MAPE	12.48	MAPE	12.1500

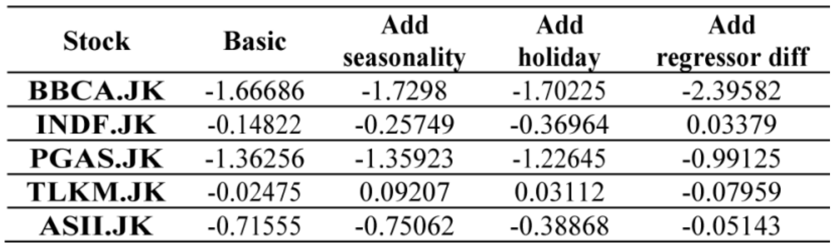
Fig. 1.3 „Prophet“ and tuned „Prophet“ model’s accuracy metrics [5, p. 4]

When comparing the two models, the “Prophet” model’s with tuned hyperparameters accuracy metrics were better than original “Prophet” model’s accuracy metrics in each case. However, to truly know if this model is worth investigating as a standalone solution for stock price prediction there needs to be a comparison of “Prophet” model’s performance with other state of the art solutions to this problem. This was precisely done in the paper [6] titled „Performance Comparison Between Facebook Prophet and Seasonal Autoregressive Integrated Moving Average (SARIMA) on Indonesian Stock”. In the paper comparison was done between “Prophet” and SARIMA model’s performances on Indonesian stock market data. For comparison purposes the “R-square” values were used. This evaluation technique measures the ability of the model to fit the data. This means it measures how well it explains the variance in the dependant value. One problem with this metric is that residuals must be checked as it may be that the model is biased and still produces high “R-square” values. For example, if residuals looked like a sine wave and the model outputs a constant value on its baseline the “R-square” value is going to be very high, but the model is bad at predicting values. Therefore, very high or low “R-square” values do not determine whether the model is good at predicting prices if the bias is strong. SARIMA model is an evolution of Autoregressive Integrated Moving Average (ARIMA) model. The difference between the two models is that SARIMA model uses seasonal or holiday influence on the data like “Prophet” model’s inner workings. Data used to train and test these models was acquired using “Yahoo finance” library from 5 different stocks: “BBCA.JK”, “TLKM.JK”, “INDF.JK”, “ASII.JK”, and “PGAS.JK”. From downloaded financial data the adjusted close price value was used and the period in question was from 2015 to 2018. The data used for the SARIMA model had to be additionally pre-processed to determine whether the data was stationary. For that Dickey Fuller test was used. After such test, the data was split into training and testing sets with testing set comprising of 2018 financial data of stocks in question. Both models after training got their test predictions compared to real test labels and the final “R-square” scores were acquired. The scores can be seen in figures 1.4 for SARIMA model and 1.5 for “Prophet” model. From the results it was clear that SARIMA model had better explanation of the dependant variable. There was no comparison of residuals mentioned in the paper and there was no data on their bias. This means that this study does not paint a full picture of model performance.



Stock	R-square Train	R-square Test
BBCA.JK	0.998646608	0.532256085
INDF.JK	0.968515678	0.608235752
PGAS.JK	0.966695575	0.858859054
TLKM.JK	0.987291077	0.579977102
ASII.JK	0.938760611	0.612354348

Fig. 1.4 „R-square“ values for SARIMA model [6, p. 981]



Stock	Basic	Add seasonality	Add holiday	Add regressor diff
BBCA.JK	-1.66686	-1.7298	-1.70225	-2.39582
INDF.JK	-0.14822	-0.25749	-0.36964	0.03379
PGAS.JK	-1.36256	-1.35923	-1.22645	-0.99125
TLKM.JK	-0.02475	0.09207	0.03112	-0.07959
ASII.JK	-0.71555	-0.75062	-0.38868	-0.05143

Fig. 1.5 „R-square“ values for „Prophet“ model [6, p. 981]

One of the more popular model types used for time-series data is an LSTM model. It has cells that can manage their states by way of gates and can memorize certain information or forget it. The use of such model for stock price predictions was written in the paper [7] titled “An LSTM-Based Model for Stock Price Prediction”. To train an LSTM model one has to take certain steps that are shown in the block diagram 1.6.

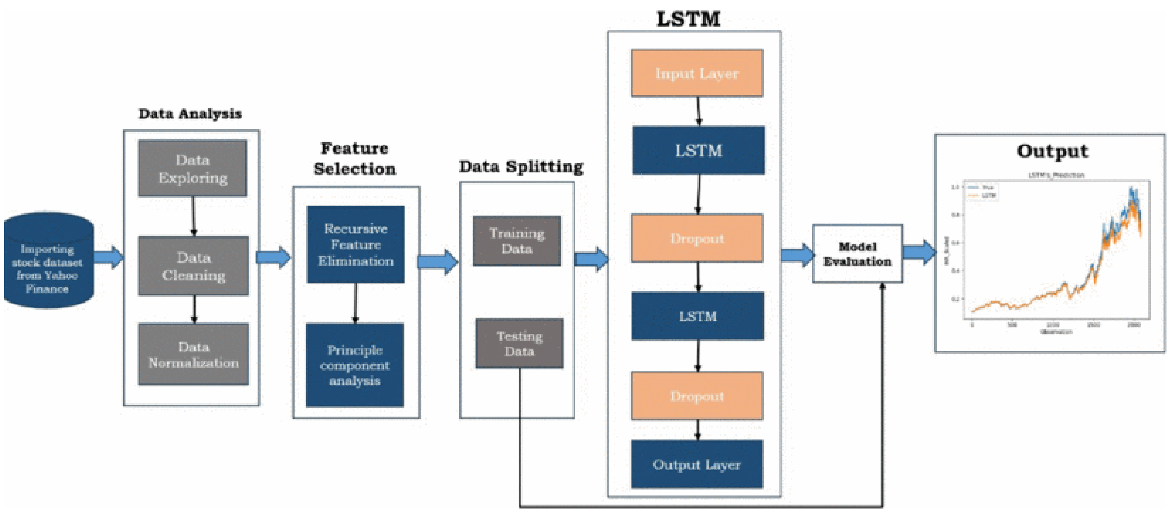
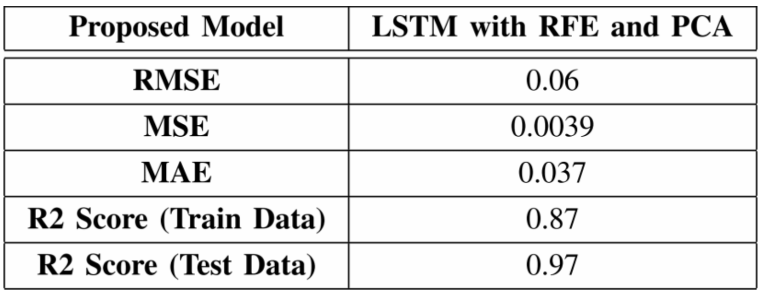


Fig. 1.6 Block diagram of an LSTM model steps required to train and exploit it [7, p. 3]

The financial data of one company, which was not named in the paper, was downloaded using “Yahoo finance” library and spanned from 1980’s to 2021. The dataset had open, close, adjusted close, highest, lowest daily stock price, date and finally trading volume. The values were then normalized and the dataset was split into training and testing sets with a ratio of 80:20. The model had a lookback period of 30 data points which means that the model was predicting next day’s stock price based on the previous 30 days. An algorithm called Recursive Feature Elimination (RFE) was also used. The RFE algorithm removes unnecessary features. Principal Component Analysis (PCA) was also used to systematically reduce number of dimensions. This makes the model more efficient by decreasing the amount of noise generated by large amount of data. The evaluation metrics used in this research were RMSE, MSE, MAE and “R-square” values. The values are listed in the figure 1.7.



Proposed Model	LSTM with RFE and PCA
RMSE	0.06
MSE	0.0039
MAE	0.037
R2 Score (Train Data)	0.87
R2 Score (Test Data)	0.97

Fig. 1.7 LSTM model’s with RFE and PCA evaluation metrics [7, p. 5]

While MSE and “R-square” values do not represent the full view of model performance on their own, the additional metrics confirm paper’s conclusion on the model’s accuracy. In the paper it was stated that the highest accuracy was about 94% and LSTM with RFE and PCA algorithms outperformed an LSTM model without mentioned algorithms. In the paper [8] titled “Stock Price Trend Analysis and Prediction of Closing Price Using LSTM” “Microsoft” stock price was predicted using LSTM model with RFE method only. Before the use of RFE algorithm, the dataset contained 20 years of financial data that included date, open, close, adjusted close, lowest, highest and transaction volume features. After the use of RFE algorithm the only features that were left were date, open, highest, lowest and transaction volume features. The features had to be pre-processed before they were acceptable to use in an LSTM model. The pre-processing steps were normalisation, removal of invalid or missing values and finally splitting data into training and testing sets with a ratio of 80:20 percent. The model achieved 71% prediction accuracy on the testing dataset. While MSE was used in the paper as a loss function, the final MSE value was not mentioned. Finally, paper [9] „Estimation of Market Price through LSTM Model“ used a standalone LSTM model. In this paper the dataset contained data from 2018 to 2024. It included downloaded financial data from „Google“, „Apple“, „Infosys“ and „Amazon“ companies. The only data used for training the model was normalised daily close stock price. It was assumed that if time step was equal to 1 then the current close price was dependent on the previous close price. This dataset was then split with a 65:35 ratio into training and testing sets. After the model was tested the model’s accuracy was reported above 90%. According to previous studies, this should not be possible as this model only used one data type to predict future values with no optimisation methods. If the reported accuracy were real, it would be better if there was a trading strategy tested which used output of the model to identify trading signals.

Single model solutions suffer from accuracy issues as stock price prediction is a multivariate, complex problem that for now no single model can solve it. For that reason, researchers try to combine models to increase their accuracy. Such research can be found in the paper [10] titled “Stock Price Prediction using Combined LSTM-CNN Model” where CNN and LSTM models were combined. Two different CNN models with different input layer sizes were tried. Model input data was composed of date, open, close, adjusted close, lowest and highest stock prices as well as transaction volume which were all normalised. The second model called DIFF-CNN also had calculated difference between daily prices. In tests DIFF-CNN did better than RAW-CNN model and was picked to be combined with LSTM network. The final model was trained for 100 epochs with adam optimiser and had training step of 0.001. The daily price predictions were made using the last 7-day data. The data except for the difference was fed into the LSTM model, however it had to be pre-processed. This LSTM model was trained for 100 epochs with training step equal to 0.001 and used adam optimiser. LSTM layers had "ReLU" activation function. The two models were trained separately with separate datasets. After training was completed, model's weights were saved and the final third model was trained with three neural layers (16,32,16) so that it could combine the two model output predictions. The dataset was comprised of "Apple" stock data that reached from 1980-12-15 to 2021-07-23. This data was gathered from the dataset found on the website called "Kaggle". All models were evaluated using MSE values. DIFF-CNN had MSE value of 5.516, LSTM had MSE value of 5.514 and LSTM-CNN model had MSE value of 4.446. From the results DIFF-CNN had the best MSE value out of the three models. LSTM model predicted stock price better than the CNN model only when it had a larger dataset. The combined model always predicted prices better than the two standalone models. However, the use of only MSE value is not sufficient as it is biased toward larger errors itself. Another research paper [11] titled "Stock Market Prediction Using Genetic Algorithm Assisted LSTM-CNN Hybrid Model" tested performances of 4 different neural networks: CNN, LSTM, LSTM-CNN and finally GA-LSTM-CNN for stock price prediction. The GA-LSTM-CNN model was similar to LSTM-CNN model; however, part of its structure was chosen by a genetic algorithm. The goal of this algorithm was to improve model's accuracy by changing said model's structure. GA optimizes the model for a specific case at the expense of longer model training time. 4 models were trained with "Apple" stock price data from 2010 February 3rd to 2023 for training, testing datasets and it included open, close, highest, lowest stock price data as well as other information such as volume, dividends, and stock splits. This dataset was downloaded using "Yahoo finance" and then was pre-processed. Along normalisation there was an additional pre-processing step taken which allowed GA to pick the "best_window_size" number. This influenced how much data the model takes in as a time-series sequence for stock price prediction of the next day. All of the models were compared using RMSE accuracy metric. From analysing proposed model's structure, it was clear that the LSTM layer was always the first layer. This was due to the fact that CNN layer transforms data dimensions which was not acceptable for LSTM layer. The genetic algorithm controlled the size of the input window and LSTM layer unit count in such a way that the RMSE value had to decrease. Custom function was used to split the dataset into sequences based on the chosen input window. The models were trained for 20 epochs. LSTM, CNN, LSTM-CNN, and GA-LSTM-CNN had RMSE test set values of 0.09, 0.053, 0.04, and 0.03, respectively. LSTM model captured the overall stock price trend, however some of the fluctuations were not captured. The CNN model captured some of the patterns in the stock price data, but had a tendency to overestimate or underestimate actual values, especially during periods of high volatility. The combination of LSTM and CNN models performed better than individual models, but training and validation loss curves were fluctuating which could mean that the model was overfitting. Finally, GA-LSTM-CNN model was able to capture the overall trend of the stock price and was predicting the price accurately. The problem was the use of only MSE metric to compare the models to. Trade signal classification analysis or accuracy would have been good metrics to add as they show model's performance in predicting price movement. Another way to combine constructed models is to join them in an ensemble architecture. Such architecture for stock analysis was written in the paper [12] titled "Candlestick Chart Based Stock Analysis System using Ensemble Learning". First, candlesticks on the found dataset had to be classified as "bullish" or "bearish". The dataset consisted of 11 global indices found on "Yahoo Finance" and on average ranged about 15 years of data. This was split with a ratio of 70:30 to the training, testing datasets, respectively. Data that was pre-processed included volume, opening price, high price, low price, closing price, and adjust closing price. For the model Extreme Gradient Boost (XGBoost) was chosen as it can do classification, regression problems simultaneously. The ensemble was based on the gradient boost family of models and was constructed using decision tree models. This ensemble received all 18 candlestick patterns as Boolean value true or false, 6 financial information features and 2 classified features of "bullish" or "bearish" market that were passed like another pattern alongside candlesticks. Two trading strategies were tested: buy and hold (B&H) strategy where stocks were bought at the beginning of a specified period and sold at the very end disregarding any short-term price fluctuations, proposed approach (PA) where model was allowed to drive buying or selling of stocks based on prediction of future price movement. B&H strategy with this model had 53.8% average accuracy while PA strategy had a 59.42% average accuracy. While other metrics are provided, their averages were not calculated. It was still clear that the selection of a trading strategy has a strong impact on the model accuracy and performance. Paper [13] titled "Stock Market Forecasting Using Hyperparameter-Tuned Ensemble Model" used XGBoost with hyperparameter tuning in order to get better results. Data was taken from "Yahoo Finance" website and consisted of NIFTY, NASDAQ, NYSE indices from 2017 to 2022. This data included volume, open, close, high, low, and adjusted close. As there were some missing values, they were replaced by mean values of the adjacent two days. Technical indicators were calculated to aid the model. The technical indicators were EMA, MACD, CCI, Momentum, Ease of Movement (EMV), and Bollinger Bands (BB). Labels were encoded as 0 if the price of the asset dropped and 1 if the price of the asset had risen since the previous day. Entire dataset was split into training and testing sets with a ratio of 80:20. For training dataset K-Fold Cross Validation was sued as it partitioned training data into k sets at random and each set was used as a validation set once. The other sets were used for the model creation. This way it was hoped to avoid overfitting problems. There were 4 hyperparameter tuning methods used and compared with each other on all indices separately. The methods were Grid Search (GS), Random Search (RS), Bayesian Optimisation (BO), and Particle Swarm Optimisation (PSO). Each group was then used for XGBoost training. As XGBoost is based on decision trees, the hyperparameters that were tuned included: depth of a tree, learning rate, subsample ratio of the training instances, minimum loss reduction needed for further split, subsample ratio of features used for fitting the individual tree, and minimum weights of the instances required in a leaf. The model was then evaluated based on how well it predicted the price movement. Model evaluation accuracies are shown in the figure 1.8.

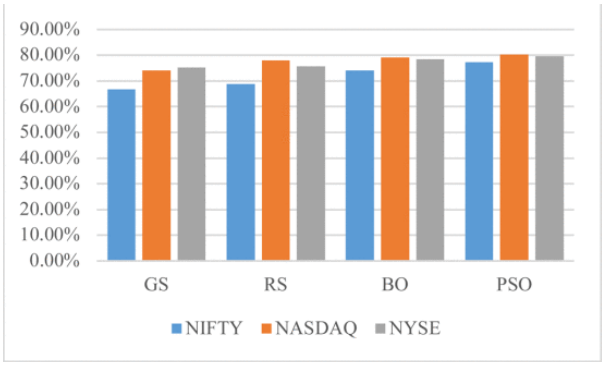


Fig. 1.8 XGBoost's accuracy results based on different hyperparameter tuning algorithms and indices [13, p. 270]

From the results it is clear that two extremes were GS and PSO methods. PSO was more consistent across indices and more accurate than GS method. PSO on average scored 79.08% accuracy while GS scored 72.07% average accuracy. PSO on average had precision of 0.8, 0.84 recall and F1 score of 0.82. Another paper [14] titled "An Ensemble Learning Model Integrating Short-term Trend and

Long-term Trend Used in Stock Price Forecasting" attempted solve a problem with SVR model in the stock market. The problem happens when the model cannot fit multiple price trend characteristics that change through time. According to the paper, this was also why historically SVR algorithms had low accuracy predicting the price. This was attempted to solve using ensemble of two SVR models where one model learned from the entire stock history and the other only from a short segment. The one that learned from the entire history was called the long-term model and the one that learned from recent market events was called the short-term model. The result was a weighted average of both model's outputs where the bigger the weight was the more output of the ensemble resembled the long-term model's output. Dataset was taken from "Yahoo Finance" and included 17 stocks total from American stocks, Hong Kong stocks, Shanghai stocks and Shenzhen stocks ranging from 2004 to 2020. This stock data included daily opening price, closing price, high price, low price and trading volume with all invalid data values removed and an additional column of the next day's price added for the model training. It was the goal to analyse how the look back window of the short-term model affected the overall performance of the ensemble. Then the same experiment was done for the weight of the ensemble. For the main metric RMSE values were used to compare the two models, but no average of this value is given. The only explanation given was that ensemble obviously was better than single model SVR at predicting prices combined with a large table for results. In the results it was found that the shorter the window, the smaller the long-term model needed to be weighted and the better the performance was. However, overfitting could occur with the loss of the weight value. This leads to performance degradation. It was concluded that windows and weights needed to be selected according to stocks separately to realise accurate predictions.

While regression models were often used for time-series data like stock market price prediction, CNN models can also be used for such tasks. While combined models might use CNN for time-series data, it is often used to find patterns between features and not patterns in time. This then requires a transformation of time-series data into images that could be processed by such a model. In the paper [15] "Image-based time series forecasting: A deep convolutional neural network approach" 3 separate CNN model architectures were compared by first transforming time-series data into images that could be processed by CNN models. Datasets chosen were M3 and M4 which store time series, including stock market, data. The data preparation was standard at first: splitting dataset by fixed window size and normalising values. One key feature mentioned in the paper is the ability of the model output to be outside 0 to 1 range as future values could be bigger than all previous values. Transformation of 1D time-series data into 2D images was done by simply creating a black and white image of a line plot from the data inside the windows. According to paper, black and white images were more efficient in terms of space and model complexity required to understand them when compared to the coloured images. All images had the same dimensions of 64 by 64 pixels. The general model architecture that was picked for this task was ForCNN which had an encoder and a regressor. The encoder of the model was changed and the model was named ForCNN-SD. Firstly, the model consisted of stacks which took images and created latent representations of them. Fully Connected (FC) layer was then used to generate predictions of future values. Secondly, each stack consisted of convolutional blocks and then a final convolutional layer for reduction of feature map dimensions. Finally, each convolutional block consisted of convolutional, batch normalisation layers and "ReLU" activation function. The encoder's purpose was to transform input images into a vector that contains latent representation. In the ForCNN-SD architecture each block had a connection to allow said block to map output to the input directly. The regressor was used to produce requested forecasts given embedding vector which was produced at the end of encoding process. The regressor was a simple fully connected neural network with non-linear hidden layers and a linear output layer. To reduce the risk of random model weights heavily effecting the final result same type models were trained as an ensemble of 50 models. The metrics used for model evaluation were RMSE, MAE and Overall Weighted Average (OWA). It was found that ForCNN-SD was better than other models and methods proposed for the problem in OWA metric, especially when it came to yearly data. The model was slightly worse in some metrics depending on situation. One identified drawback of the model was that it was sensitive to the changes in the model's architecture. Other use of image-based predictions with time-series data was written in the paper [16] titled "Market Sentiment Analysis Based on Image Processing With Put-Call Volatility Gap Surface" in which volatility was used to forecast future stock prices. It was done by constructing a gap implied volatility (IV) surface from the derived implied volatility of call and put options, projecting said surface to a 2D image, and then using a Recurrent CNN (RCNN) model to predict future prices. Data was obtained from ICE, including "SPDR" "S&P500" ETF (SPY) and its options trading data which included trading size, price of the option, execution date, etc. Options are priced according to IV. Implied volatility measures how much the price might move in the future, but not in what direction. As options can be exercised in the future, it indirectly shows what investors think about the future price thus extracting market sentiment. IV can be derived in two dimensions which are strike price and time to maturity. Strike price is the price at which the option can be exercised and time to maturity is the expiration date of the options contract. To calculate IV Barone-Adesi and Whaley (BAW) model with bisection was used. The IV calculation was done for call and put options, respectively. Call options imply that the stock price will rise in the future and the opposite is true for put options. Getting IV gap surface required calculation of the difference between call and put IVs. Then one axis of this surface was strike, second axis was time to maturity and third axis was IV gap value. The higher the gap, the more positive movement of the underlying asset is expected and vice versa. This surface was then projected onto the 2D image where IV gap value was encoded as pixel colour. Impact of call, put, gap and fusion surfaces were tested with a CNN model and MAE, MSE, Direction Correctness (DC) metrics. DC measures how well the model predicts asset movement direction which is not often done in other works. While fusion of surface was the most accurate in all metrics, gap surface was faster than fusion surface and better in all metrics when compared to call, put surfaces only. Options have high-frequency data which allows calculation of the velocity, acceleration IV surfaces. The entire framework is demonstrated in the figure 1.9.

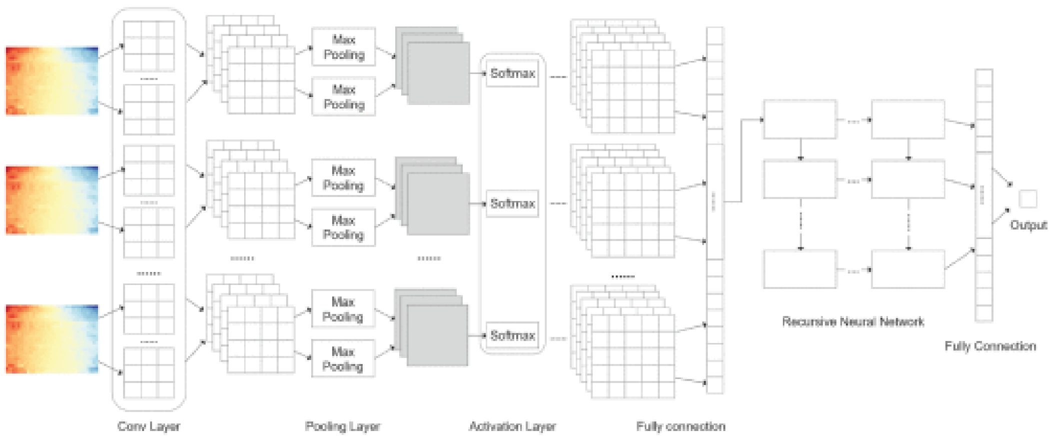


Fig. 1.9 RCNN based model structure for velocity, acceleration IV surface test [16, p. 278]

Three models in total were tested: RCNN with gap surface; CNN with gap, call, put and their velocity surfaces; CNN with gap, call, put, their velocity and acceleration surfaces. RCNN model was slower than the two CNN models. RCNN also had better price prediction accuracy, but was worse at predicting direction as its DC was 2% lower than other CNN models. CNN with velocity and acceleration surfaces was the most accurate model with DC reaching 72.05% and provided fast prediction when compared to CNN with only velocity surface. In the paper [17] “Image Processing Based Implied Volatility Surface Analysis for Asset movement Forecasting” implied volatility surface is also used. First difference was the use of Bjerksund-Stensland (BS) option pricing model with bisection instead of BAW. According to the paper, it was faster on the computer when compared to numerical methods. After that, IV surfaces were constructed like in the paper [16]. When IV was derived for strike price it formed a smile pattern on a line graph. Higher value of a smile is indicative of investor fear towards the future. This is how IV surface holds information of market sentiment and reveals potential trends of the market. Data was collected from SPY or SPDR S&P500 ETF and its options trading data that was then split with a ratio of 80 to 20 for training, testing datasets, respectively. For the experiment only call IV surface was constructed which could be due to the fact that their pricing model was not suited for other option types, but it was not mentioned in the paper. Call IV surface momentum surfaces were also constructed. Momentum surfaces include velocity and acceleration surfaces. LSTM, CNN-C with only call surface, CNN-C-V with velocity added and finally CNN-C-VA with additional acceleration surface models were tested. CNN models between convolutions used SoftMax activation function. This resulted in CNN-C-VA being the most accurate model with directional accuracy of 65.5%, MAE value of 0.04 and MSE value of 0.004 while slightly slower than other CNN models. LSTM was less accurate than other CNN models in all metrics and on average took more time than all other CNN models.

The implied volatility is not used in CNN models only. In the paper [18] titled “The Implied Volatility Surface Analysis Based Trading System” IV surface data was used for support vector regression (SVR) based trading system. SVR used SVM and linear regression to derive final value. SVR analysed IV surfaces based on SPY option’s trading data. BAW pricing model was used in this case. Both call and put surfaces were calculated, but no gap surface was derived. The SVR model had to predict what the underlying price or the implied volatility was going to be in the next 5 minutes. Trading strategy proposed was that if this underlying prediction rises, the underlying stock was bought and vice versa. The system would also buy options when IV was predicted to go down and sell when it was predicted to go up as it negatively correlates with the underlying. To test the system, it was trained on data collected in the first half day while the second half was used for testing. At first MAE and MSE results showed that IV surfaces had better performance than underlying and put options were more suitable for prediction than call options. The results changed when the model was in simulated trading environment that had factors like transaction costs. In such environment call options surpassed profit of put options while underlying had lower returns due to transaction costs. As a result, underlying forecast were not suitable for trading signals and IV call surface better captured the market opportunity with 4-day return of 0.74% with average Sharpe ratio of 1.66. The Sharpe ratio of 1.66 meant that the model made good return when compared to risks. The use of simulated environment to get performance of the model was a great method to use as it explains model performance well even without directional metrics. If financial instrument is sparsely used it can be difficult to find enough data to get a comprehensive IV surface. Paper [19] titled “Creating Synthetic Volatility Surfaces using Generative Adversarial Networks with Static Arbitrage Loss Conditions” aimed to synthesise volatility surface out of data using Generative Adversarial Networks

(GAN). This work used Wasserstein distance between data distribution generated by GAN, Variational Auto Encoder (VAE) as a benchmark and between GAN and true distribution from the historical dataset which was used as a performance metric. Wasserstein distance is a minimal distance between two random variables that marginally satisfy variables describing data and model generated distributions. This on average measures how close the distributions are and was used as a performance metric for the model. The task for GAN was to synthesise such distribution that would produce minimal Wasserstein distances between it and benchmark, real historical data. The generated distribution also had to satisfy conditions for arbitrage so that it was not able to produce risk free profits as such thing is not realistic. To enforce such an arbitrage, modified loss function was used which would satisfy the necessary conditions. Dataset was taken from Johannesburg stock exchange All-Share Index (ALSI) where 144 samples were used for training and 36 samples for testing. inverted Black-Scholes pricing model was used on observed prices as it best fits European options. In the paper GAN and VAE architectures were similar. The results showed that GAN model's with arbitrage conditions Wasserstein distance to real data was the smallest and equal to 0.502 while without arbitrage conditions it was 1.069. Arbitrage conditions for VAE, on the other hand, did not produce significant changes and its Wasserstein distance without arbitrage conditions was the worst at 76.583. There was no explanation given for the vast difference in performance between VAE and GAN.

Any neural network solution to a problem needs to have a dataset which can be comprehended and accurately analysed by said neural network model. This is why data pre-processing is a particularly important field of study as it discovers new ways to generate a dataset out of raw data that is faster, more efficient, produces better results, finds new ways to represent new and old data alike. Paper [20] titled "A Comparative Evaluation of Noise Reduction Versus Data Normalization Techniques in Stock Market Prediction Using Transformer Models" examined the impact of six normalisation techniques and 10 technical indicators on recursive transformer models for stock market prediction. The novel transformer model architecture had multi-head input layer that allowed the model to separately analyse inputs instead of combining them and it was adapted from natural language processing to handle financial data. The multi-head input layer allowed the feature to pass through the model on its own path so its impact could be directly measured. It was important due to the fact that multiple normalisation techniques were used for the same data type that needed to be evaluated. The outputs of each feature analysis were later combined in the model so that the model could learn individual feature patterns and their interactions. For the dataset "NASDAQ", "S&P500", "Johnson&Johnson" and "IBM" stock data were downloaded from "Yahoo Finance" website. This dataset contained data from 1990 to 2024 and had opening and closing daily prices. The dataset was split with a ratio of 90:10 into training and testing sets, respectively. The 10 technical indicators mentioned in the paper were all averages and were designed to reduce noise. As averages use some period across which the data is considered for the calculation it was important to measure what was the best period for each technical indicator. The technical indicators with their best daily period setups were 5-period Triple Exponential Moving Average (TEMA), 5-period DEMA, 11-period Weighted Moving Average (WMA), 5-period EMA, 10-period EMA, 20-period EMA, 5-period SMA, 10-period SMA, 28-period Zero-Lag Exponential Moving Average (ZLEMA), Fractal Adaptive Moving Average (FRAMA). The six normalisation techniques tested were minmax, z-score, max abs scaling, quantile transformation, standard scaler, robust scaler. Two datasets were tested: raw dataset that contained only daily opening and closing prices, dataset that contained raw data and additional 10 technical indicators. In the paper only "S&P500" and "Johnson&Johnson" results were listed while "IBM" and "NASDAQ" results were not found. Evaluation metrics for the models were MAPE, MSE and R-squared. In the experiments, normalisation techniques used with the raw dataset did not have a significant impact on the model output performance. Only quantile transformer had a slight positive impact on “S&P500” and “NASDAQ” indices. While raw data with technical indicators produced worse accuracy for the model, when the right normalisation technique was applied to the inputs the model’s output accuracy was better. In the research “NASDAQ”, “IBM” and “S&P500” had the most positive results with min-max scaler, max abs scaler while “Johnson&Johnson” had best results with standard scaler and robust scaler. This cannot be verified as some results were not listed. Another paper [21] titled “Data Preprocessing for Stock Price Prediction Using LSTM and Sentiment Analysis” described the pre-processing steps for sentiment analysis models. It found out optimal batch size, epoch count for accuracy on Tata Consultancy Services (TCS) data. TCS data had been taken from “Yahoo Finance” and ranged from 2015 to 2021. First task was to pre-process data for regression task which was done using different techniques. Data was scaled using minmax scaler and outliers were replaced with mean values. LSTM network was trained with different epochs and batch sizes. According to the paper, the best results were with default TensorFlow library values which were not mentioned in the paper, but were batch size of 32 and 1 epoch. The most consistent results were of batch size 32 and 32 epochs. Metrics by which this was evaluated were R-squared, max error and Mean Squared Log Error (MSLE). The default model had the highest R-squared value of 0.956, the lowest max error of 4.55 and the second lowest MSLE value of 0.0064. There were no movement metrics listed. The second task was to pre-process textual data taken from twitter which consisted of 50 tweets. Text pre-processing steps involved dropping duplicate entries, removing emojis, translating entries to English, removing hashtags, mentions, retweets, hyperlinks, punctuation, converting words to lowercase, removing stop words, tokenisation, and finally stemming and lemmatization which reduced words to their root. The sentiment analysis was done by TextBlob model that gave an output between -1 and 1 where -1 was negative sentiment and 1 was positive sentiment. After preliminary tests, the dataset of tweets was taken from “Kaggle” and pre-processing was done by encoding the labels. After that, the vectorisation method allowed data to pass to the actual model as input. Then the output from TextBlob was taken as an input to LR model. The final accuracy of predicted price on test dataset was 79.23%. No other metrics for that part were listed which means that the description of the effectiveness of the model is lacking.

With the new popularity of transformers and Large Language Models (LLM) it was inevitable that they were researched for stock trading. They provide a unique challenge as they are language-based models and are not particularly fit for regression tasks. In the paper [22] titled “Stock Price Prediction Using LLM-Based Sentiment Analysis” best suited LLM was picked for news classifications and the classification result was tested for its impact on regression task. The data used for this experiment was taken from “Apple” (2015-2024), “Tencent” (2016-2024) and “Toyota” (2017-2024). Stock information was taken for regression task and news headlines for sentiment analysis. To find the best LLM six different model were tested: GPT 4, Llama 3, Gemma 2, Mistral 7b, FinBERT, VADER. The prompt they were given to classify news headlines were “classify the sentiment of this piece of news headline: [Headline Input]. Sentiment is "positive", "negative" or "neutral". Return only the sentiment of the news headline.” Two datasets were used to test the models: labelled financial news dataset that contained 400 pieces of news labelled by statisticians, Indian stock market analysts, data scientists; Financial Phrasebank which contained 2000 news sentences annotated by 16 people with background in finance. Financial Phrasebank produced better results (most likely due to the higher example count), but Llama 3 and Gemma 2 were both consistently good across both datasets. The best accuracy they got was around 0.85. Due to their insignificant differences, Llama 3 was used in further study. Better performance by fine tuning the LLM models could not be achieved. For stock price prediction, 1 and 16 days into the future, seven different regression and transformer models were used. Technical indicators and macroeconomic indicators were used in all scenarios. Three different scenarios were tested in total: with no sentiment analysis, with polarity score, with predicted sentiment. Polarity score applied a range of -1 to 1 for negative, positive sentiment respectively while predicted sentiment had classes of positive, neutral, and negative sentiment. The seven models used for stock price evaluation were: transformer, informer, LSTM, Temporal Convolutional Network (TCN), SVR, RF, Naive Forest (NF). The stock information was taken from "Yahoo finance" website while news was collected using news feed and news sentiment data API. Technical indicators used were SMA, RSI, On-Balance Volume (OBV) that measures cumulative buying, selling pressure and Directional Movement Index with Average Directional Index (DMI-ADX) to measure strength, direction of a trend. Macroeconomic data used was gold price and USD/JPY exchange rate. Different window sizes for both input and output were also tested. Input window sizes were 16, 32, 96 days while output window sizes were 1 and 16 days. Training and testing sets were split with a ratio of 70:30, respectively. Minmax scaler was also used for both datasets. Two metrics were used to evaluate model predictions and they were MSE and MAE. There were no directional metrics used for model evaluation. With all of this it was found that adding sentiment analysis and especially polarity score yielded better results. With one day close price forecasts NF was performing better second only to informer model, but for 16-day predictions informer was the better choice seconded by transformer models. Informer model for 16-day prediction and polarity could reach MSE value of 0.01 depending on the lookback period used. In general TCN outperformed LSTM models, but was not able to reach the accuracy of the other models. Another paper concerned with transformers in Iranian market is titled [23] "Multi-Task Transformer for Stock Market Trend Prediction". In this paper regression and classification was done simultaneously (multi-tasking) and data paneling method was utilized. First, the dataset was from Iranian stock market data which included open, close, low, and high prices. Paneling refers to a method that makes multiple observations per data point over-time to establish some trend that can be seen between features or even companies. It was not mentioned how paneling is done in the paper itself, but it was hinted that patterns in one stock can emerge in another one, meaning that multiple stocks were tracked during this process. In the paper data was labelled weekly as the model was tasked to predict next week's prices in 5 classes. The labelling was done with 5 levels that depended on the weekly price movement and quantiles. The labelling itself was done by prioritising outer levels over inner levels and when the price crossed a barrier in the outer level it was classified as that class. In the paper, K-fold cross-validation algorithm was used for training the model, but with additional purge and embargo methods. Purging refers to a method that removes samples from training data whose evaluation time is later than the earliest prediction time. Embargo in this case refers to a method that removes some samples right after the test set, so there could be no time pattern bleed over from test set to training set. The transformer model required two loss functions as it was doing classification and regression tasks. Two losses were combined to do backpropagation. Model structure was designed in such a way that regression loss propagates to the classification and vice versa. According to the paper, this meant that each task limits the other task's search space and thus decreased each other's loss. The model had to predict stock trend of the next five days using the past 40 days. The data was passed through a vector representation of time - Time2Vec. This was done so that the transformer could understand time dependencies and how stock prices should be placed based on their temporal order. The model was trained for 200 epochs alongside model that did just classification to check whether or not the model's multi-tasking feature improved overall performance. There was no separate model for regression task comparison. In addition, six other models were examined with quintuple barrier labelling technique and they were CatBoost, Decision Tree, KNN, Multilayer Perceptron (MP), Ridge, Stochastic Gradient Descent (SGD). Overall, the transformer model with multi-tasking got the best results when it came to the classification, but regression performance was not mentioned. Classification achieved 59.62% precision which was 16% higher than transformer's without multi-task, 53.75% recall which was 12% higher, F1 score of 52.79% which was 12% and 55.3% accuracy which was 14% higher. Another body of work investigating performance of Time2Vec vectoral time representation within transformer models when compared to other models used for stock trading is a paper [24] titled "Stock Price Prediction Using Transformer and Time2Vec". Here Time2Vec was used for encoding temporal features, capturing periodic patterns in price movement while transformer model was tasked with extraction of spatial and temporal features from encoded data. The combined model's task was to predict future prices in the form of regressed value which was done by feeding transformer output to a fully connected layer. The model was tested on benchmark datasets designed to test models for stock market trading and which included opening, closing, high, low prices, trading volume. After testing this and LR, Kalman Filter, RNN models the conclusion was that transformer with Time2Vec was the superior model. Looking at the results, it seems like RNN model results were ignored. RNN had higher MSE value, but lower RMSE value. This could mean that RNN model was not making large errors while on average deviating slightly more from the true price compared to transformer model. The model had MSE value of 0.0025, RMSE value of 0.05 and R-squared value of 0.92 while RNN had MSE value of 0.0035, RMSE value of 0.0035 and R-squared value of 0.9. It was still argued that the work highlighted Time2Vec potential for practical applications in financial analysis and algorithmic trading. It was mentioned that transformers were very reliant on large datasets. The work also did not analyse the accuracy of price movement predictions. Finally, paper [25] "Global Stock Market Prediction Using Transformer-Based Deep Learning Techniques" explored transformer model usage in seven different economies: China, Canada, India, Japan, Russia, the United Kingdom and the United States. The work aimed to explore applicability and adaptability of transformer model in different markets and thus included different markets into the training dataset. Raw data needed to be pre-processed before serving as a dataset for the model training. This meant handling of missing values, that in this case were replaced by zeroes, calculation of derivative financial indicators like EMA, normalisation of values by using a method like minmax scaler and finally splitting of data into training, testing sets. The ratio of splitting this dataset into training and testing sets was not mentioned in the paper. The transformer model was structured around an encoder-decoder module and self-attention mechanism. Two other models were tested alongside the transformer model for performance comparison purposes and they were CNN and RNN models. It was found that transformer had lower MAE, MSE and RMSE values compared to other models which did highlight its adaptability and price prediction abilities. MAE had a value of 0.01, MSE had a value of 0.0003 and RMSE had a value of 0.018. There were no price movement metrics analysed.

A more specific use for transformer model architecture is its application as a baseline for generative neural networks. One such network for the use in stock market was explored in the paper [26] "Multi-Step-Ahead Stock Market Prediction Based on Least Squares Generative Adversarial Network". It used Least Squares Generative Adversarial Network (LSGAN) for predicting closing price of stocks. The use of LSGAN instead of just GAN was due to the vanishing gradient problem caused by the standard cost function. GAN generally consists of a generator and a discriminator. Generator tries to generate realistic data and discriminator tries to classify the data as real or generated. Due to this, generator learns to produce realistic data samples that could not be distinguished from real data. In an ideal case the generator pushes the cost function of the model to the maximum limit while discriminator tries to minimise this cost function. LSTM was chosen to be a generator for closing prices of the next 5 trading days. The output of LSTM was given to a fully connected layer that consisted of 15 neurons tasked to generate closing price value. Fully connected layer had leaky ReLU activation function and to prevent overfitting of the model dropout method was used. MLP was chosen as a discriminator which had 3 hidden layers of 128, 64 and 64 neurons, respectively. The hidden layers used leaky ReLU while the output used sigmoid activation functions. For the dataset "S&P500" data collected from "Yahoo Finance" of the last 20 years (about 5000 data points from 2000 to 2020) was used. This data consisted of open, high, low, closing, adjusted closing prices, and trading volume. Model also used 9 technical indicators that were: SMA, MACD, RSI, William %R, SO, Price Rate of Change (PROC), ADX, BB, log return. It was argued that stock market data does not have consistent mean and variance over time. That is a Random Walk process which generates nonstationary data. Differentiation in time domain must be done to transform such time-series set into a stationary time-series set. After that Z-score method was used to remove outliers, Wavelet Transformation was used to smooth out noise in the financial data and all the values were normalized with minmax scaler. The dataset was split into training and testing sets with a ratio of 80:20. The metrics for the model performance were MAE, RMSE, MAPE and Average Return (AR). LSGAN and GAN models were tested to check if Least Squares method really improved the model and the results show that LSGAN was better than GAN in every metric. LSGAN got final values of 0.15 MAE, 0.18 RMSE, 0.02 MAPE and -0.00009 AR. Interestingly the negative number for average return was not commented on in the paper. It could mean that despite better metrics the LSGAN model could not be deployed in real market conditions. Metrics for price movement prediction accuracy were not included in the paper. Another novel approach using GAN model and Wavelet Transformation was found in the paper [27] "A Novel Wavelet Based Generative Model for Time Series Prediction". Wavelet transform was used to realise and decompose the distinct levels of frequences in time series. Market signal was decomposed into frequency layers and coefficients were obtained for each layer. Then Auto-Regressive Moving Average (ARMA) model was constructed for the wavelet coefficients and predicted coefficients were obtained. The predicted data was then reconstructed back from predicted wavelet coefficients. To generate predicted wavelet coefficients Wasserstein GAN with Gradient Penalty (GP) was used. For comparison purposes multiple models were tested on the collected dataset. The models included GRU, LSTM, GAN, and Wavelet-GAN. Not much was mentioned about the dataset itself, only that it was collected from 2023 to 2024 closing price data. This data was then normalised using minmax scaler. For the metrics MSE, RMSE, MAE, R-squared values were used and again no price movement metrics were mentioned in the paper. From the results it was clear that Wavelet-GAN was more accurate and had better explanatory power than other models. GRU did not capture complexities of time-series data, LSTM had better long-term predictive power, but lacked accuracy, and GAN improved accuracy through an adversarial network which was further improved with Wavelet Transformation in Wavelet-GAN model. Final model's metrics were 2.97 for MSE, 1.72 for RMSE, 1.59 for MAE and 0.99 for R-squared. As volatility data can be accepted by GAN models, paper [28] "Research on Stock Price Volatility Prediction Based on Generative Adversarial Network" searched for possible applicability of this data in order to predict "S&P500" daily closing price one day ahead. LSTM and Exponential Generalized Autoregressive Conditional Heteroskedasticity (EGARCH) models were used. Daily closing price data was taken from "S&P500" from 2010 to 2021 and constituted 2840 datapoints. Then this raw data was transformed into a stationary form using Logarithmic Return, its volatility was calculated with a 22-day rolling window and then split into training, testing sets with a ratio of 80:20. For metrics, only

MSE and MAE were used to compare the model performances. EGARCH model was not accurate and its visualised predictions did not align with the shape of real volatility. On the other hand, LSTM did assume the shape, but was not as accurate as GAN model which had particularly good prediction accuracy. In terms of metrics GAN had MSE value of 3.72E-06. That was almost ten times smaller than LSTM's MSE value of 6.23E-05. MAE value for EGARCH model was 1.10E-03 and it was slightly worse than LSTM's MAE value of 4.04E-03. An example of using GAN model as a basis for custom model framework was found in the paper [29] titled "RAGIC: Risk-Aware Generative Framework for Stock Interval Construction" in which Risk-Aware Generative Interval Construction (RAGIC) model was proposed. Firstly, in addition to regressed value of stock price, the model generated an interval which showed where the price could end up. Secondly, Volatility Index (VIX) was used in the model to predict overall market sentiment that informed the width of the intervals. RAGIC model consisted of two phases which were sequence generation and interval construction. In the sequence generation the model was generating a sequence based on multi-scale trends expressed by historical price data and risk attention score derived from VIX. In the interval construction the risk-aware interval was constructed from generated sequences that had different prediction horizons. With multi-head attention module, the long-term global dependencies were modelled while TCN was used to model short term local, and periodic patterns. The discriminator of this GAN used an approximation of Wasserstein distance between real and generated sequences obtained from GRU. An appropriate interval was constructed by assuming it had a normal distribution with unknown mean and variance. The mean and variance were computed from the obtained predictions. The dataset was taken from "Yahoo Finance" and "Investing". This dataset consisted of 5 indices with their corresponding volatility index taken from 2013 to 2021. The indices were: "Dow Jones Industrial Average", "S&P500", "NASDAQ100", "Deutscher Aktienindex" performance index, the Nikkei index, and their corresponding volatility indices VXD, VIX, VXN, VDAX, JNIV. The training data was taken from 2013 to 2015 while testing data was taken from 2016 to 2021. The task was to generate sequences 5 days into the future with the past 30 days as input. To do this 4 historical prices (open, high, low, close), 8 technical indicators (MACD, DIFF, Upper Bollinger band indicator, Lower Bollinger band indicator, 5-day EMA, 13-day SMA, 21-day SMA, 50-day SMA) and 2 volatility indicators (Volatility index, Return of Volatility index) were used. Baseline models were tested along with RAGIC to compare performances. The baseline models included Bollinger Bands, BayesianNN, Multi-Quantile RNN (MQRNN), Conformal RNN (CFRNN), Diffusion, Denoising, and Disentanglement VAE (D3VAE), StockGAN, FactorVAE. For a fair comparison D3VAE, StockGAN and FactorVAE were modified with the proposed interval constructor as they were originally used for one value predictions. In this case the metrics used to evaluate the given interval by Coverage Probability which described how many values fall into the generated interval over the test period, Normalised Mean Width was the mean difference between upper and lower bounds was evaluated, Coverage Width-based Criterion (CWC) which combined two previous metrics to show balance between informativeness and accuracy. In all cases the CWC value of RAGIC was better than CWC of other models. While intervals generated by FactorVAE contained 100% of all points, its width was too wide and was not informative. Other models also had either big interval or a narrow interval, but less points in it when compared to RAGIC. The optimal prediction horizon was found to be 5 days while risk score was found to have enough information to warn about COVID pandemic's financial effects. During the test of point value prediction, the model's predictions were averaged using a weight that exponentially decreased with increasing horizon step. Two more models were also tested and they were AdaRNN, Hierarchical Multi-scale Gaussian Transformer (HMG-TF). In all cases RAGIC performed with highest accuracy when compared to all of the models.

The state of the art was largely focused on testing and comparing different novel models, trading strategies or features. The combination of several models in an ensemble or combination of different model layers into one model was shown to have significant positive effect on the overall performance of the designed systems. Moreover, tests with different trading strategies or features allowed for a better understanding of their effectiveness and predictive power in real trading scenarios. This thesis is focused on chart pattern trading strategies and their impact on the model's predictive ability. The chart patterns in question can be found using analytical rules following methodology outlined in the paper [3]. Furthermore, GA-LSTM-CNN model was selected due to its performance shown in the paper [11]. In this thesis, the model was used to automatically select features that were beneficial to the model, analyse the features to find spatial, temporal patterns and to give the prediction on future price movement. A lot of state-of-the-art research analysed neglected trade signal metrics and they must not be forgotten as they show how well the model detects trend reversals and overall price trend.

2. Proposed method

The main method of trading assessed in this work was chart patterns. Chart patterns were extracted from financial time-series data. The chart patterns were searched for backwards to not give the model any forward knowledge on future financial events by accident (this is impossible in a real scenario). To do this, special points were selected that were points of interest in the financial data which preserve the form of the graph over the last 21 days. Extraction was done by turning point method. For the point to classify as a turning point, it had to be higher or lower than the previous and next data point. Classification of the graphs was done through rules that define graph's shape for a specific chart pattern and were described in [3]. The rules had to be light enough so that the algorithm did not accept only perfect patterns, but strict enough to not allow other patterns into the same category or detect patterns where no patterns were present. For example, head and shoulders downtrend pattern has to have a specific form made from high and low points. Then it has to have two trend lines on top of the pattern, one going up and the other going down. The two trend lines have to be similar which must be defined by the user according to needs. Preferably, the one trend line that is on the underside has to be flat. Chart patterns were also classified as uptrend and downtrend. Uptrend chart patterns signify that the price movement is going to be upwards (the price of the stock will be increasing) and vice versa. When the price trend is going up it is called bullish and when it is going down it is called bearish. According to chart pattern trade strategy, when the uptrend chart pattern is detected, the investor has to buy the stock and when the downtrend chart pattern is detected the investor has to sell the stock. There are two main complications of chart patterns when it comes to this trading strategy succeeding. One complication is that for this strategy to work it has to be used by substantial number of other investors. Second complication is that theoretically it can only predict future price movement based on past price movement and does not take into account other sentiments like news, bad financial reports etc. This means that even if chart patterns work, they will work only during regular trading conditions and they do not protect against other sentiment driven events.

Model that was picked for all experiments was GA-LSTM-CNN model. This was done due to the fact that GA could be used to select features and optimal model structure automatically. That was important as there are many trading strategies and it was not clear what strategy should be used. In the same way it was not clear what features should be passed to the model. GA was specifically tasked to find the lookback period that determines LSTM layer size and how far back the model looked at the values in order to determine the next day's price, search for optimal penultimate dense layer’s size by altering its parameter, find optimal features that determine next day’s price by creating a bit mask. Bit mask was later converted into a Boolean mask for the purpose of selecting features. The GA-LSTM-CNN models themselves in order had one LSTM, two CNN and two dense layers responsible for final closing price prediction generation. CNN layers had ReLU activation function and final dense layer had sigmoid activation function. Model optimiser was adam and loss was calculated using RMSE method. During preliminary testing LSTM layer had 30 units, both CNN layers had 16 units and penultimate dense layer had 20 units. CNN layers had kernel sizes of 5 and 3 to look for bigger and smaller patterns within data. While LSTM layer’s and penultimate dense layer’s unit size depended on GA, GA always picked these numbers for both cases. After preliminary testing, some restrictions on unit size had to be made as the model was overfitting. After some restrictions and downsizing of the model, the lookback period was 60 days, both CNN layers had 3 units and penultimate dense layer had 3 units. After such operation, the model had better training results than initial testing showed. Model's main task was to predict next day's stock closing price based on information gathered from previous trading days.

ReLU and sigmoid were both used in the model as activation functions. Activation functions were used to transform a simple linear unit of the model into a non-linear unit which could solve more complex problems. ReLU was used in hidden layers as it was faster than sigmoid and did not have backpropagation errors like sigmoid did. While the ReLU function did have training instability issues, they were solved by downsizing the model. ReLU function simply takes the maximum between 0 and the input. That means if the input is above zero, the value is unchanged, but if it is below zero the value will be clamped to zero.

Sigmoid was used for the last dense layer despite the fact that it was slower than ReLU. It was chosen due to its stability and the fact that the entire dataset was normalised according to data type. This means that values in the dataset range from 0 to 1 like sigmoid function. This makes training and testing the model easier, but would take extra work to deploy in the real scenario as values above dataset's maximum would need to be accounted for. Previously tanh function was used and the labels consisted of the difference between values of selected and previous day. This function was not used as it produced worse model performances. The sigmoid function is shown in the equation 2.1.

$$y = \frac{1}{1 + e^{-Z}} \tag{2.1}$$

If: y – output of the function; Z – input to the function.

The dataset was normalised using minmax method. The values were normalised according to their data type. For example, daily open price data for the company was taken, then highest and lowest values of said data were found. When the values were found minmax function was applied across the entire data type for that company. The minmax function is shown in the equation 2.2.

$$y = \frac{Z - min}{\max - min} \tag{2.2}$$

If: y – output of the function; Z – input to the function; *min* – lowest value in the datatype found in the dataset; *max* – highest value in the datatype found in the dataset.

As a loss function, RMSE value was used. Loss functions are used as model performance measurement that allows the model to improve its parameters in order to get better performance. To do this, error was calculated between observed and predicted values by taking the difference. This was done for all values between label and prediction vectors. Then the error was squared and the mean of all values was taken to get a single error value. Up to this point the method was MSE. MSE function penalizes larger errors more as the error is squared before the mean is taken. The difference is that the square root of MSE value is taken to bring the value back to the original scale. In this case, large errors were avoided which means that the model did not do big mistakes in its prediction, but could suffer with smaller errors. For this, the prediction simply had to be good enough to spot trend reversals and smaller errors were simply excused.

At the end of the experiments, ensemble method was used to improve model performance. The two models were combined at the output where they were compared. If the two predictions agreed, the trade signal was generated. This prediction was compared with the actual trade signal to measure the accuracy and generate metrics. On the other hand, if two predictions disagreed, no action was taken and it counted as a correct action to take. This way, some trading actions were lost, but the accuracy of the ensemble was improved. The accuracy was improved as theoretically the two models disagree on points where price fluctuations were more random or could not be explained.

Trading signals were generated by comparing current and previous predicted closing price. If current price prediction was higher than previous price prediction, the signal was to buy and vice versa. Buy trade signal was represented with a 1 or positive while sell signal was represented with a -1 or negative. The predicted trade signal was compared with the actual trade signal and the results were counted. To generate confusion matrix True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN) results were counted. TP was counted when the model correctly predicts a buy signal and TN was counted when the model correctly predicts sell signal. In this case, false observation was an incorrect model prediction of positive (buy) or negative (sell) trade signal. The four values were then displayed on the confusion matrix as percentage values where each row must add up to 100 percent.

For models that achieved better signal classification, accuracy metric was used to check how well the model differentiates between buy and sell signals. If accuracy was too low, in the majority of cases with trading models the model simply labelled more signals as buy signals and could not detect sell signals. Accuracy calculation is given in the equation 2.3.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad (2.3)$$

If: *TP* – true positive predictions; *TN* – true negative predictions; *FP* – false positive predictions; *FN* – false negative predictions.

Additionally, for ensemble performance measurement, trade signal loss metric was introduced. Every time two models disagreed on trade signal prediction it was counted as No Action (NA). The percentage of trade signals lost is calculated by equation 2.4.

$$Inaction = 100 * \frac{NA}{TP + FP + TN + FN} \qquad (2.4)$$

If: *Inaction* – percentage of actions not taken; *NA* – no action; *TP* – true positive predictions; *FP* – false positive predictions; *TN* – true negative predictions; *FN* – false negative predictions.

Trading signal graph shows predicted and true trade signals represented by 1 for buy signal and -1 for sell signal. Signal error graph only shows signals that were incorrectly predicted, 1 for incorrectly predicted buy signal and -1 for incorrectly predicted sell signal.

3. Experimental setup

3.1. Fundamental and technical analysis data

Core financial data of chosen stocks was downloaded using yfinance library. This library makes use of “Yahoo Finance” website’s public API to download financial information of the requested ticker. Downloaded financial data was saved in an Excel file to keep it static throughout the experiment. After financial data was downloaded, 3 additional technical indicators were calculated and saved into the Excel file. The 3 technical indicators were RSI, EMA and MACD. The core financial data for “Microsoft” company is shown in the figure 3.1.1, RSI, EMA and MACD technical indicators are shown in figures 3.1.2, 3.1.3 and 3.1.4, respectively. The graphs show information how the traders usually view it. The period of time shown was the entire length of the dataset and was chosen so that there would be close to equal distribution of financial information for any company. Example of how information was saved into the Excel file is shown in the figure 3.1.5.

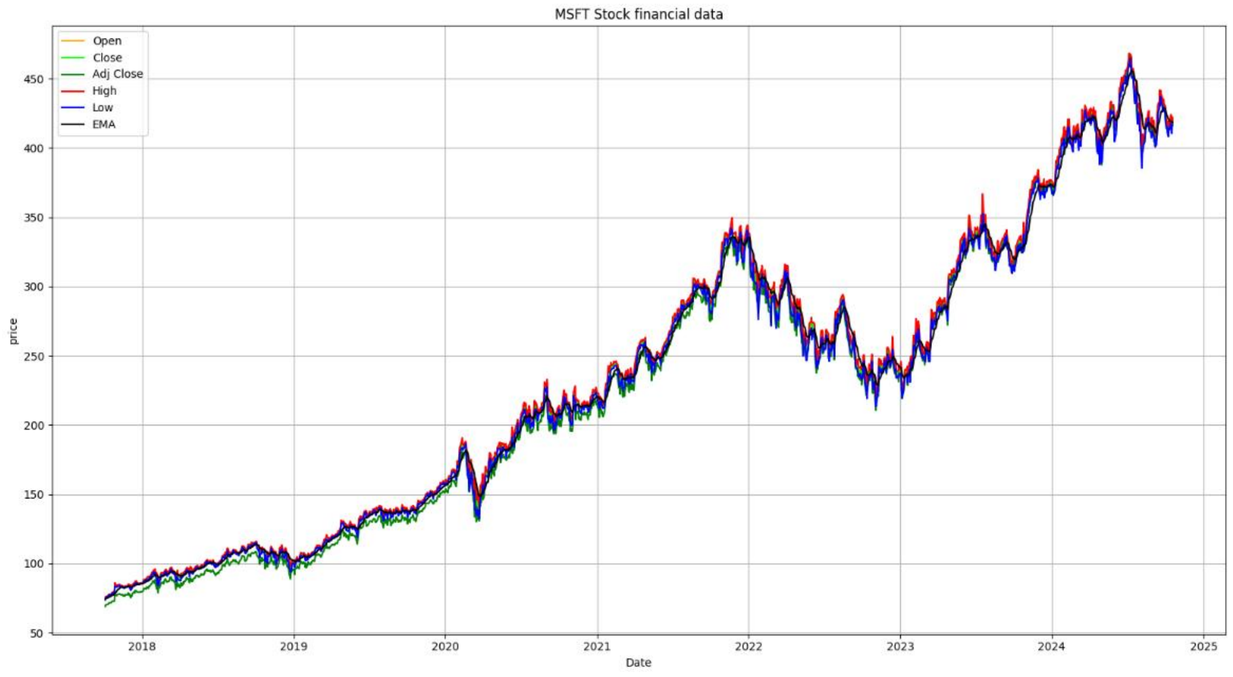


Fig. 3.1.1 „Microsoft“ fundamental financial data. X axis – date, Y axis – stock fundamental indicator value

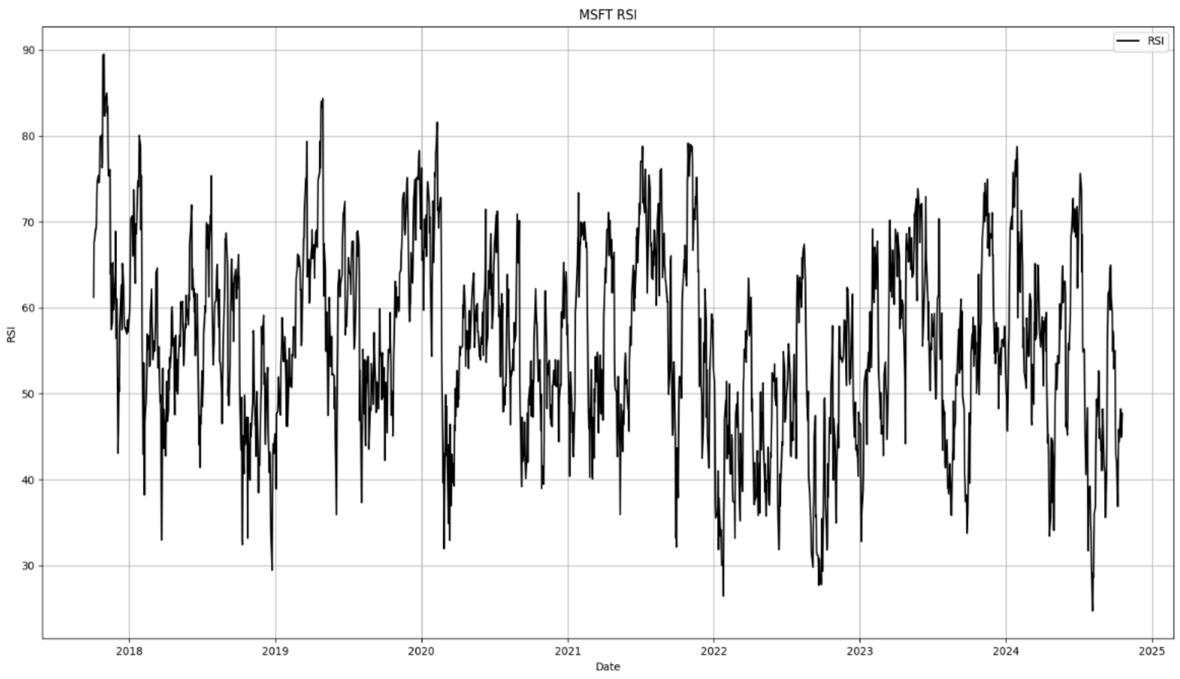


Fig. 3.1.2 „Microsoft“ technical indicator RSI where 70 is overbought and 30 is oversold. . X axis – date, Y axis – RSI value

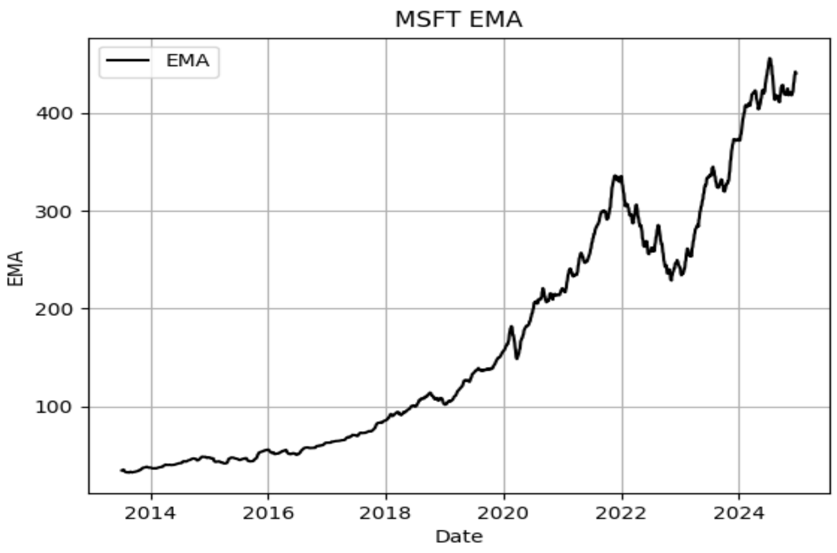


Fig. 3.1.3 „Microsoft“ technical indicator EMA for price trend. X axis – date, Y axis – EMA value

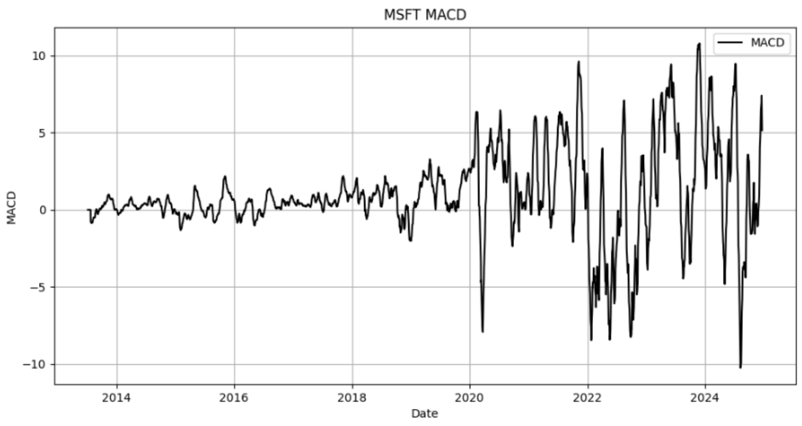
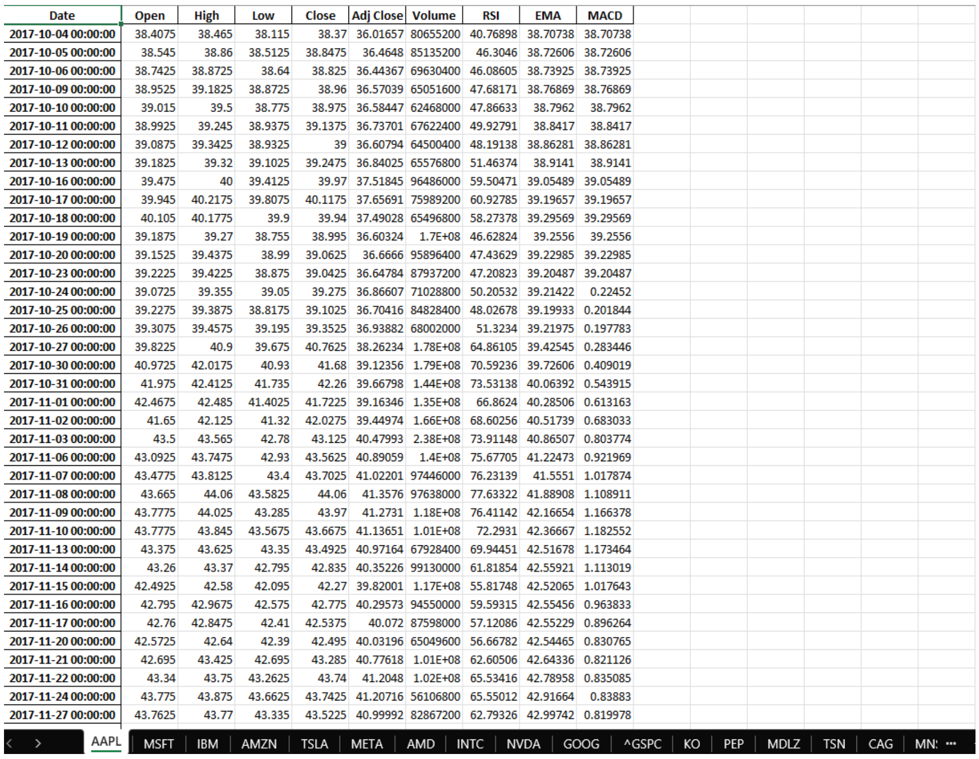


Fig. 3.1.4 „Microsoft“ technical indicator MACD for trend reversal when 0 line is crossed. X axis – date, Y axis – MACD value



Date	Open	High	Low	Close	Adj Close	Volume	RSI	EMA	MACD
2017-10-04 00:00:00	38.4075	38.465	38.115	38.37	36.01657	80655200	40.76898	38.70738	38.70738
2017-10-05 00:00:00	38.545	38.86	38.5125	38.8475	36.4648	85135200	46.3046	38.72606	38.72606
2017-10-06 00:00:00	38.7425	38.8725	38.64	38.825	36.44367	69630400	46.08605	38.73925	38.73925
2017-10-09 00:00:00	38.9525	39.1825	38.8725	38.96	36.57039	65051600	47.68171	38.76869	38.76869
2017-10-10 00:00:00	39.015	39.5	38.775	38.975	36.58447	62468000	47.86633	38.7962	38.7962
2017-10-11 00:00:00	38.9925	39.245	38.9375	39.1375	36.73701	67622400	49.92791	38.8417	38.8417
2017-10-12 00:00:00	39.0875	39.3425	38.9325	39	36.60794	64500400	48.19138	38.86281	38.86281
2017-10-13 00:00:00	39.1825	39.32	39.1025	39.2475	36.84025	65576800	51.46374	38.9141	38.9141
2017-10-16 00:00:00	39.475	40	39.4125	39.97	37.51845	96486000	59.50471	39.05489	39.05489
2017-10-17 00:00:00	39.945	40.2175	39.8075	40.1175	37.65691	75989200	60.92785	39.19657	39.19657
2017-10-18 00:00:00	40.105	40.1775	39.9	39.94	37.49028	65496800	58.27378	39.29569	39.29569
2017-10-19 00:00:00	39.1875	39.27	38.755	38.995	36.60324	1.7E+08	46.62824	39.2556	39.2556
2017-10-20 00:00:00	39.1525	39.4375	38.99	39.0625	36.6666	95896400	47.43629	39.22985	39.22985
2017-10-23 00:00:00	39.2225	39.4225	38.875	39.0425	36.64784	87937200	47.20823	39.20487	39.20487
2017-10-24 00:00:00	39.0725	39.355	39.05	39.275	36.86607	71028800	50.20532	39.21422	0.22452
2017-10-25 00:00:00	39.2275	39.3875	38.8175	39.1025	36.70416	84828400	48.02678	39.19933	0.201844
2017-10-26 00:00:00	39.3075	39.4575	39.195	39.3525	36.93882	68002000	51.3234	39.21975	0.197783
2017-10-27 00:00:00	39.8225	40.9	39.675	40.7625	38.26234	1.78E+08	64.86105	39.42545	0.283446
2017-10-30 00:00:00	40.9725	42.0175	40.93	41.68	39.12356	1.79E+08	70.59236	39.72606	0.409019
2017-10-31 00:00:00	41.975	42.4125	41.735	42.26	39.66798	1.44E+08	73.53138	40.06392	0.543915
2017-11-01 00:00:00	42.4675	42.485	41.4025	41.7225	39.16346	1.35E+08	66.8624	40.28506	0.613163
2017-11-02 00:00:00	41.65	42.125	41.32	42.0275	39.44974	1.66E+08	68.60256	40.51739	0.683033
2017-11-03 00:00:00	43.5	43.565	42.78	43.125	40.47993	2.38E+08	73.91148	40.86507	0.803774
2017-11-06 00:00:00	43.0925	43.7475	42.93	43.5625	40.89059	1.4E+08	75.67705	41.22473	0.921969
2017-11-07 00:00:00	43.4775	43.8125	43.4	43.7025	41.02201	97446000	76.23139	41.5551	1.017874
2017-11-08 00:00:00	43.665	44.06	43.5825	44.06	41.3576	97638000	77.63322	41.88908	1.108911
2017-11-09 00:00:00	43.7775	44.025	43.285	43.97	41.2731	1.18E+08	76.41142	42.16654	1.166378
2017-11-10 00:00:00	43.7775	43.845	43.5675	43.6675	41.13651	1.01E+08	72.2931	42.36667	1.182552
2017-11-13 00:00:00	43.375	43.625	43.35	43.4925	40.97164	67928400	69.94451	42.51678	1.173464
2017-11-14 00:00:00	43.26	43.37	42.795	42.835	40.35226	99130000	61.81854	42.55921	1.113019
2017-11-15 00:00:00	42.4925	42.58	42.095	42.27	39.82001	1.17E+08	55.81748	42.52065	1.017643
2017-11-16 00:00:00	42.795	42.9675	42.575	42.775	40.29573	94550000	59.59315	42.55456	0.963833
2017-11-17 00:00:00	42.76	42.8475	42.41	42.5375	40.072	87598000	57.12086	42.55229	0.896264
2017-11-20 00:00:00	42.5725	42.64	42.39	42.495	40.03196	65049600	56.66782	42.54465	0.830765
2017-11-21 00:00:00	42.695	43.425	42.695	43.285	40.77618	1.01E+08	62.60506	42.64336	0.821126
2017-11-22 00:00:00	43.34	43.75	43.2625	43.74	41.2048	1.02E+08	65.53416	42.78958	0.835085
2017-11-24 00:00:00	43.775	43.875	43.6625	43.7425	41.20716	56106800	65.55012	42.91664	0.83883
2017-11-27 00:00:00	43.7625	43.77	43.335	43.5225	40.99992	82867200	62.79326	42.99742	0.819978

Fig. 3.1.5 Saved ticker fundamental and technical financial data

This dataset was made from financial information of 20 tickers: „Apple“, „Microsoft“, „IBM“, „Amazon“, „Tesla“, „Meta“, „AMD“, „Intel“, „Nvidia“, „Google“, „S&P500“, „Coca-Cola“, „PepsiCo“, „Mondelez“, „Tyson Foods“, „Conagra Brands“, „Monster Beverage“, „Archer-Daniels-Midland“, „Diageo“, „Kellanova“. The financial data was separated into different Excel pages for each ticker. This information was taken from 2013-07-08 to 2024-12-20. In this dataset there were daily open, high, low, adjusted closing and closing prices, trading volume, 14-day RSI, EMA and 14, 21-day MACD features. RSI ranges from 0 to 100 and shows if the stock is overbought or oversold. It counts as overbought when it goes above 70 and oversold when it is below 30. EMA is average which measures current price trend. MACD incorporates 2 EMA's to predict trend reversal. Trend reversal is a point at which two averages cross each other or in this case crossed 0 line.

3.2. Chart Pattern Data

Chart patterns that were classified included Diamond Uptrend (DU), Diamond Downtrend (DD), Head and Shoulders Uptrend (HASU), Head and Shoulders Downtrend (HASD), Triple Uptrend (TU), Triple Downtrend (TD). The graphs for all the patterns in that order are shown in figures 3.2.1, 3.2.2 and 3.2.3.

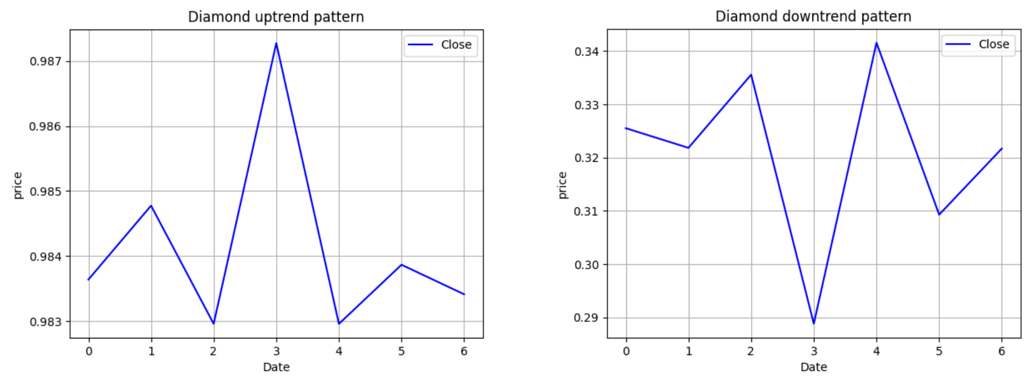


Fig. 3.2.2 Diamond uptrend and downtrend (DU, DD) chart patterns

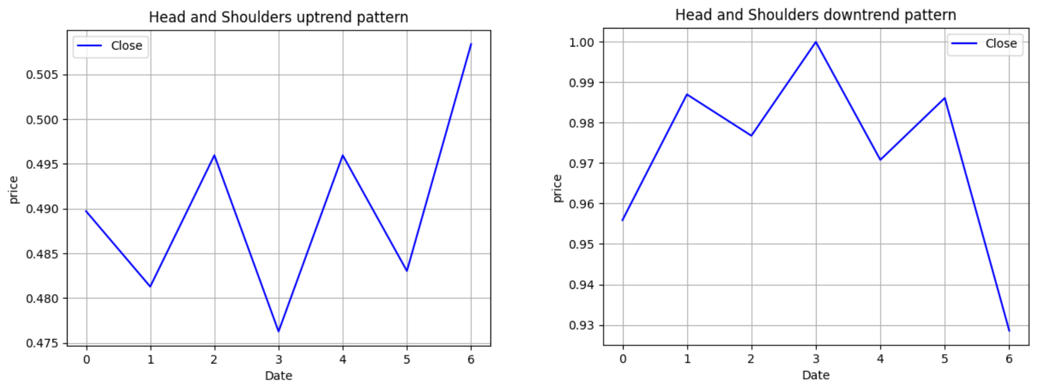


Fig. 3.2.3 Head and Shoulders uptrend and downtrend (HASU, HASD) chart patterns

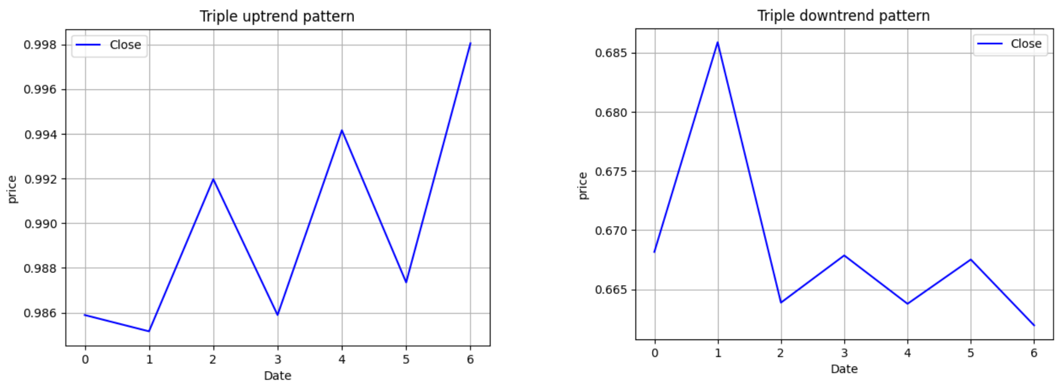
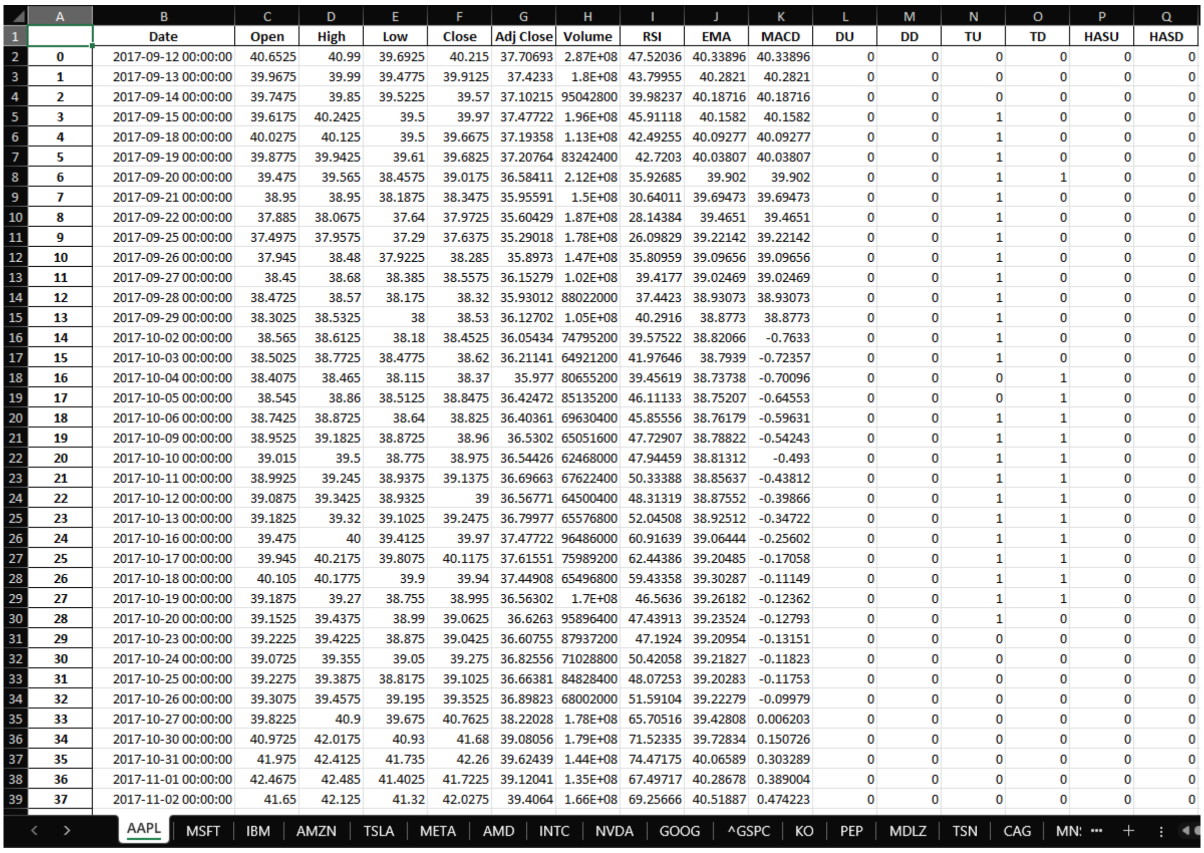


Fig. 3.2.1 Triple uptrend and downtrend (TU, TD) chart patterns

Modified Excel file is shown in the figure 3.2.4. In the final Excel file used for the model training and testing there were fundamental financial data and technical indicators, discussed previously, modified with additional data for chart patterns that the algorithm found. In the dataset, when chart pattern category has a value of 1, that means that the specific chart pattern was found within 21 days from that day and 0 means that there was no pattern withing that time range.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Date	Open	High	Low	Close	Adj Close	Volume	RSI	EMA	MACD	DU	DD	TU	TD	HASU	HASD
2	0	2017-09-12 00:00:00	40.6525	40.99	39.6925	40.215	37.70693	2.87E+08	47.52036	40.33896	40.33896	0	0	0	0	0	0
3	1	2017-09-13 00:00:00	39.9675	39.99	39.4775	39.9125	37.4233	1.8E+08	43.79955	40.2821	40.2821	0	0	0	0	0	0
4	2	2017-09-14 00:00:00	39.7475	39.85	39.5225	39.57	37.10215	95042800	39.98237	40.18716	40.18716	0	0	0	0	0	0
5	3	2017-09-15 00:00:00	39.6175	40.2425	39.5	39.97	37.47722	1.96E+08	45.91118	40.1582	40.1582	0	0	1	0	0	0
6	4	2017-09-18 00:00:00	40.0275	40.125	39.5	39.6675	37.19358	1.13E+08	42.49255	40.09277	40.09277	0	0	1	0	0	0
7	5	2017-09-19 00:00:00	39.8775	39.9425	39.61	39.6825	37.20764	83242400	42.7203	40.03807	40.03807	0	0	1	0	0	0
8	6	2017-09-20 00:00:00	39.475	39.565	38.4575	39.0175	36.58411	2.12E+08	35.92685	39.902	39.902	0	0	1	1	0	0
9	7	2017-09-21 00:00:00	38.95	38.95	38.1875	38.3475	35.95591	1.5E+08	30.64011	39.69473	39.69473	0	0	1	0	0	0
10	8	2017-09-22 00:00:00	37.885	38.0675	37.64	37.9725	35.60429	1.87E+08	28.14384	39.4651	39.4651	0	0	1	0	0	0
11	9	2017-09-25 00:00:00	37.4975	37.9575	37.29	37.6375	35.29018	1.78E+08	26.09829	39.22142	39.22142	0	0	1	0	0	0
12	10	2017-09-26 00:00:00	37.945	38.48	37.9225	38.285	35.8973	1.47E+08	35.80959	39.09656	39.09656	0	0	1	0	0	0
13	11	2017-09-27 00:00:00	38.45	38.68	38.385	38.5575	36.15279	1.02E+08	39.4177	39.02469	39.02469	0	0	1	0	0	0
14	12	2017-09-28 00:00:00	38.4725	38.57	38.175	38.32	35.93012	88022000	37.4423	38.93073	38.93073	0	0	1	0	0	0
15	13	2017-09-29 00:00:00	38.3025	38.5325	38	38.53	36.12702	1.05E+08	40.2916	38.8773	38.8773	0	0	1	0	0	0
16	14	2017-10-02 00:00:00	38.565	38.6125	38.18	38.4525	36.05434	74795200	39.57522	38.82066	-0.7633	0	0	1	0	0	0
17	15	2017-10-03 00:00:00	38.5025	38.7725	38.4775	38.62	36.21141	64921200	41.97646	38.7939	-0.72357	0	0	1	0	0	0
18	16	2017-10-04 00:00:00	38.4075	38.465	38.115	38.37	35.977	80655200	39.45619	38.73738	-0.70096	0	0	0	1	0	0
19	17	2017-10-05 00:00:00	38.545	38.86	38.5125	38.8475	36.42472	85135200	46.11133	38.75207	-0.64553	0	0	0	1	0	0
20	18	2017-10-06 00:00:00	38.7425	38.8725	38.64	38.825	36.40361	69630400	45.85556	38.76179	-0.59631	0	0	1	1	0	0
21	19	2017-10-09 00:00:00	38.9525	39.1825	38.8725	38.96	36.5302	65051600	47.72907	38.78822	-0.54243	0	0	1	1	0	0
22	20	2017-10-10 00:00:00	39.015	39.5	38.775	38.975	36.54426	62468000	47.94459	38.81312	-0.493	0	0	1	1	0	0
23	21	2017-10-11 00:00:00	38.9925	39.245	38.9375	39.1375	36.69663	67622400	50.33388	38.85637	-0.43812	0	0	1	1	0	0
24	22	2017-10-12 00:00:00	39.0875	39.3425	38.9325	39	36.56771	64500400	48.31319	38.87552	-0.39866	0	0	1	1	0	0
25	23	2017-10-13 00:00:00	39.1825	39.32	39.1025	39.2475	36.79977	65576800	52.04508	38.92512	-0.34722	0	0	1	1	0	0
26	24	2017-10-16 00:00:00	39.475	40	39.4125	39.97	37.47722	96486000	60.91639	39.06444	-0.25602	0	0	1	1	0	0
27	25	2017-10-17 00:00:00	39.945	40.2175	39.8075	40.1175	37.61551	75989200	62.44386	39.20485	-0.17058	0	0	1	1	0	0
28	26	2017-10-18 00:00:00	40.105	40.1775	39.9	39.94	37.44908	65496800	59.43358	39.30287	-0.11149	0	0	1	1	0	0
29	27	2017-10-19 00:00:00	39.1875	39.27	38.755	38.995	36.56302	1.7E+08	46.5636	39.26182	-0.12362	0	0	1	1	0	0
30	28	2017-10-20 00:00:00	39.1525	39.4375	38.99	39.0625	36.6263	95896400	47.43913	39.23524	-0.12793	0	0	1	0	0	0
31	29	2017-10-23 00:00:00	39.2225	39.4225	38.875	39.0425	36.60755	87937200	47.1924	39.20954	-0.13151	0	0	0	0	0	0
32	30	2017-10-24 00:00:00	39.0725	39.355	39.05	39.275	36.82556	71028800	50.42058	39.21827	-0.11823	0	0	0	0	0	0
33	31	2017-10-25 00:00:00	39.2275	39.3875	38.8175	39.1025	36.66381	84828400	48.07253	39.20283	-0.11753	0	0	0	0	0	0
34	32	2017-10-26 00:00:00	39.3075	39.4575	39.195	39.3525	36.89823	68002000	51.59104	39.22279	-0.09979	0	0	0	0	0	0
35	33	2017-10-27 00:00:00	39.8225	40.9	39.675	40.7625	38.22028	1.78E+08	65.70516	39.42808	0.006203	0	0	0	0	0	0
36	34	2017-10-30 00:00:00	40.9725	42.0175	40.93	41.68	39.08056	1.79E+08	71.52335	39.72834	0.150726	0	0	0	0	0	0
37	35	2017-10-31 00:00:00	41.975	42.4125	41.735	42.26	39.62439	1.44E+08	74.47175	40.06589	0.303289	0	0	0	0	0	0
38	36	2017-11-01 00:00:00	42.4675	42.485	41.4025	41.7225	39.12041	1.35E+08	67.49717	40.28678	0.389004	0	0	0	0	0	0
39	37	2017-11-02 00:00:00	41.65	42.125	41.32	42.0275	39.4064	1.66E+08	69.25666	40.51887	0.474223	0	0	0	0	0	0

Fig. 3.2.4 Excel file with saved fundamental, technical and chart pattern data

In the end the entire database was composed of 20 ticker fundamental, technical financial and chart pattern data. Fundamental financial information was downloaded from "Yahoo Finance" website, technical indicators were calculated from fundamental data and chart pattern data was found with analytical rules.

3.3. Model training environment

Firstly, data from excel sheet was accessed according to the model that used it. One dataset saved in Excel was kept without chart patterns and its data was fed into a similar model that used chart patterns. Secondly, the dataset data was normalized according to data type (column) values where each value was normalised according to the highest and lowest value of the data type. Then the values were split into training and testing sets with a ratio of 80:20. Training set was split further into training and validation sets with a ratio of 80:20. Training data had a form of [samples, length of sample (days), features]. Due to the data format passed into the model there were no distinctions between different companies. This way a general model was trained whose predictions were based on financial data irrespective of the company. During data splitting, it was made sure that there was no mix between companies in the same sample. Length of sample depended on the lookback period chosen by the model. Features were also dependant on the model’s choice. The output of the model was next day's predicted closing price (from 0 to 1) which could be reconstructed from minmax value to an actual price.

After data pre-processing, the genetic algorithm was set up to find the optimal lookback period. It also searched for optimal penultimate dense layer’s size and picked optimal features that determined next day’s price. The most optimal selection was used to train the final model that was used for predictions on test dataset. The best model was picked comparing all models during the entire training process with hall of fame function. Results of training, testing and the final model were saved separately. One file stored Boolean array of model selected features. Second file stored [y_true, y_pred] array for testing dataset and the third file stored [y_true, y_pred] array for training dataset. The y_true was always the true stock market price and y_pred was model predicted stock price. Models were saved by storing final model’s weights.

The model was trained on the computer with 16 GB random access memory and Intel(R) Core(TM) i7-9750H central processing unit that had 2.6 GHz, 6 cores and 12 logical processors. The main model training was done on an NVIDIA GeForce RTX 2070 graphics processing unit that had 16 GB of combined internal memory.

4. Experimental results

This section shows the performance of trained models. Performance was measured with generated metrics and loss graphs made after training, validation process. Metrics were generated with the use of saved test results in a format discussed in the previous section. Generated metrics included confusion matrix of trading signals, normalized predicted, true closing price graph and trading signal, error diagrams.

4.1. Preliminary testing data

During preliminary testing, models with and without chart patterns were tested in order to set the baseline. The model with chart patterns during testing used High, Close Adjusted, Close, Volume, RSI, EMA, MACD, DD, HASU features. Model’s training and validation loss graph is given in the figure 4.1.1. The final RMSE value of the model was 0.0198.

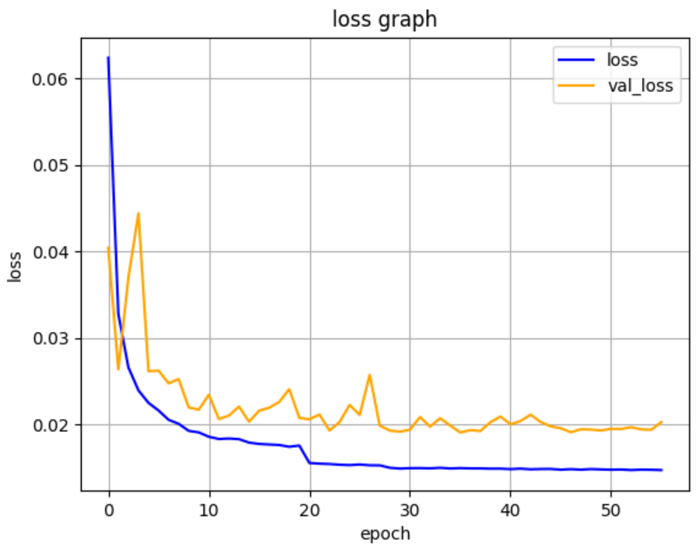


Fig. 4.1.1 Initial model’s with chart patterns loss diagram. Blue – training loss, Yellow – validation loss, X axis – epoch, Y axis – RMSE loss value

Resulting validation loss fluctuation was due to overfitting, but after tests were concluded, the size of the model was reduced to avoid this effect. The test shows that the model can use and train on chart patterns. Confusion matrix of trading signals, normalized predicted and true closing price graphs and trading signal, error diagrams are shown in figures 4.1.2, 4.1.3, 4.1.4 and 4.1.5, respectively.

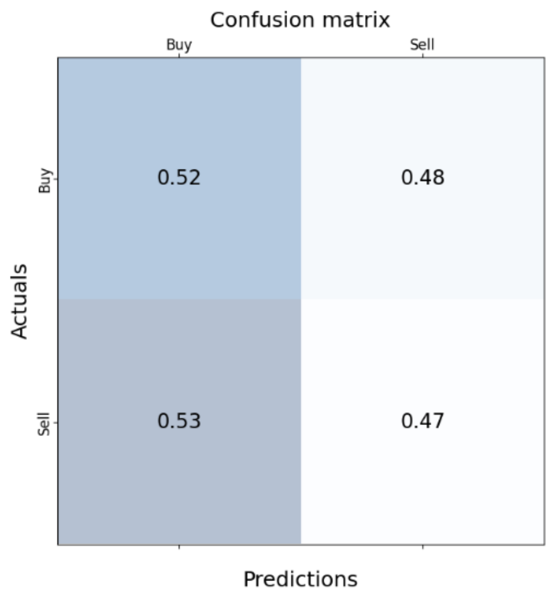


Fig. 4.1.2 Initial model's with chart patterns trading signal confusion matrix. X axis – predicted trade signal, Y axis – actual trade signal

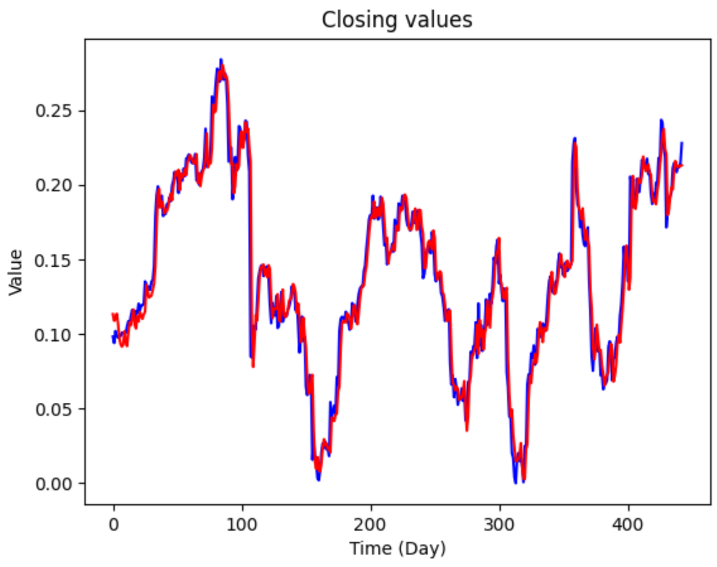


Fig. 4.1.3 Initial model's with chart patterns closing price diagram. Red – predicted price; Blue – real price, X axis – time of sample, Y axis – normalized stock daily close price.

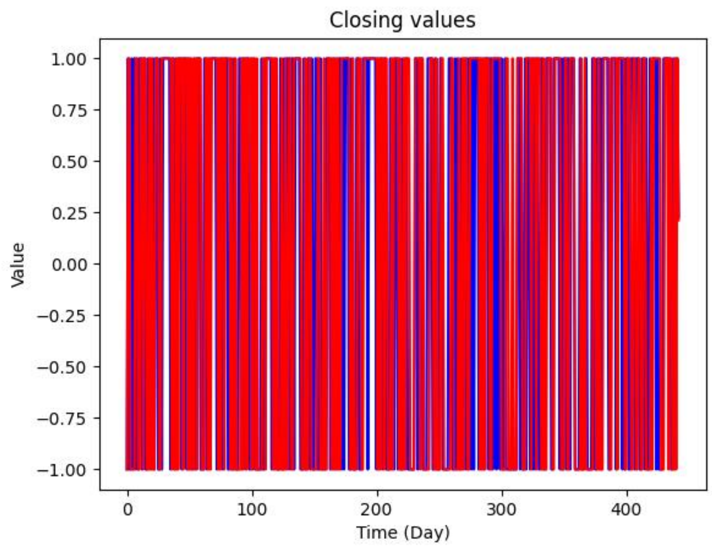


Fig. 4.1.4 Initial model's with chart patterns trading signal diagram. Red – predicted signal; Blue – real signal, X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

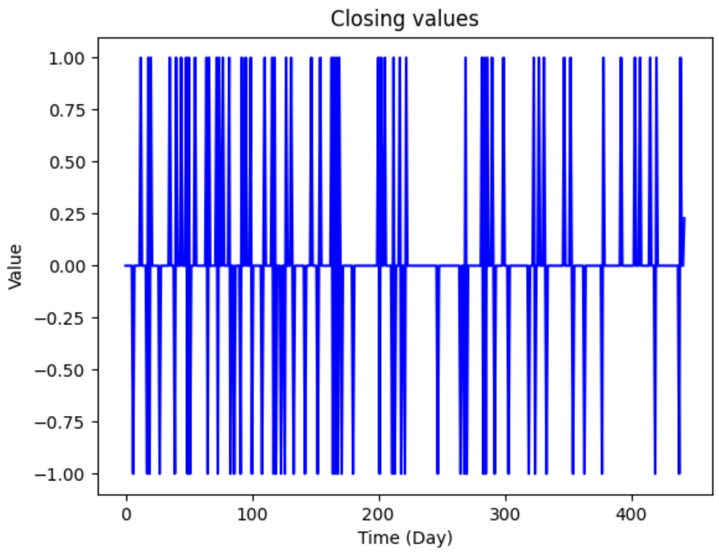


Fig. 4.1.5 Initial model's with chart patterns trade signal error. X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

While this model predicted close price accurately, predicted price movement was with a slight delay and the confusion matrix shows that the model cannot predict whether the price will go up or down. Probability of this model to predict price movement direction accurately is comparable to a coin toss.

During preliminary testing, another model without chart patterns was trained for comparison purposes. Model's architecture was the same. Loss graph is shown in the figure 4.1.6. The final RMSE value for this model was 0.344.

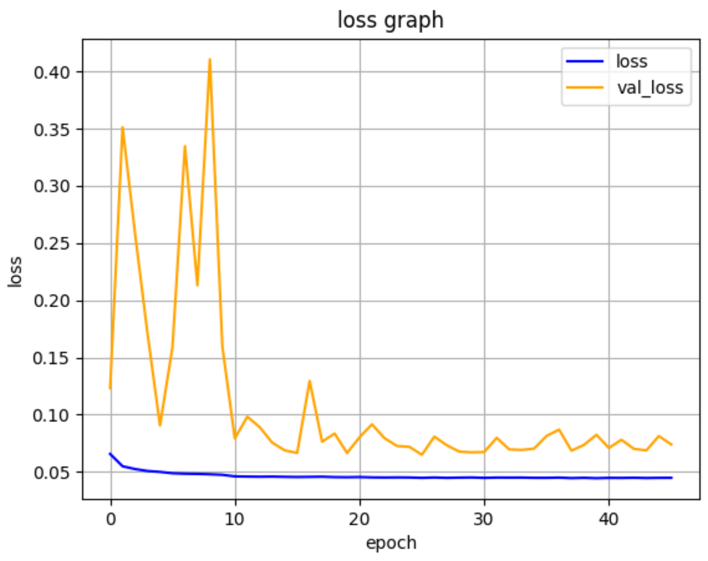


Fig. 4.1.6 Initial model's without chart patterns loss diagram. Blue – training loss, Yellow – validation loss, X axis – epoch, Y axis – RMSE loss value

The model without chart patterns used Open, High, Close Adjusted, RSI, EMA, MACD features. The loss graph is unstable which was corrected in the final test. This model also had worse RMSE value than the previous model. This could mean that chart patterns provide the model with information to make more accurate predictions. Similarly, initial model's without chart patterns metrics are shown in the figures 4.1.7, 4.1.8, 4.1.9 and 4.1.10 in the previously specified order.

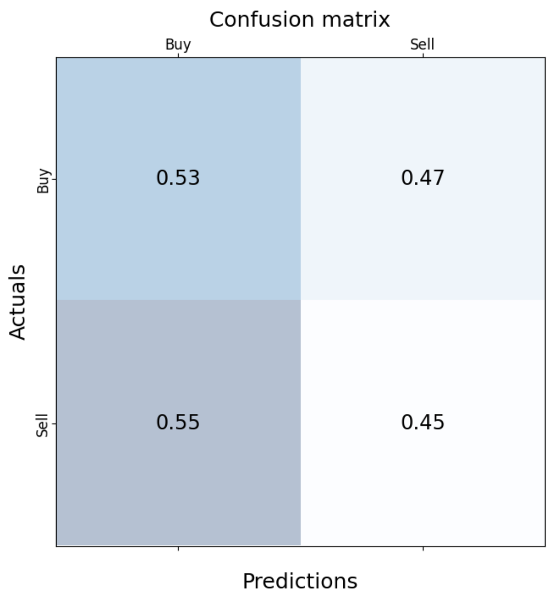


Fig. 4.1.7 Initial model's without chart patterns trading signal confusion matrix. X axis – predicted trade signal, Y axis – actual trade signal

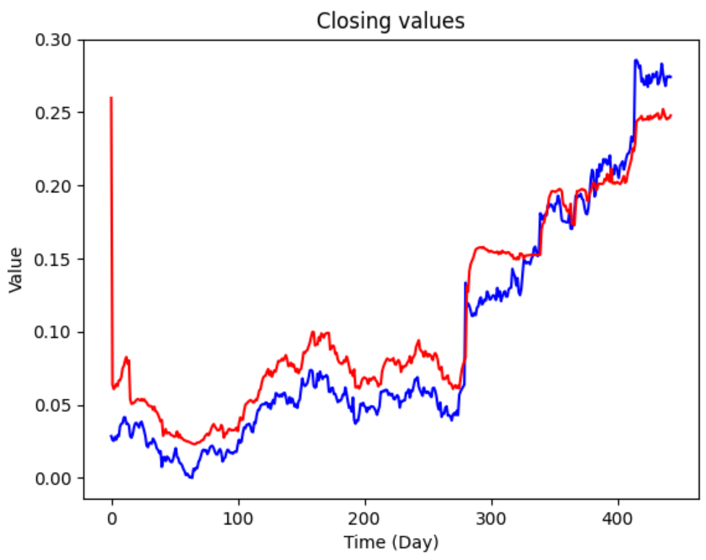


Fig. 4.1.8 Initial model's without chart patterns closing price diagram. Red – predicted price; Blue – real price, X axis – time of sample, Y axis – normalized stock daily close price.

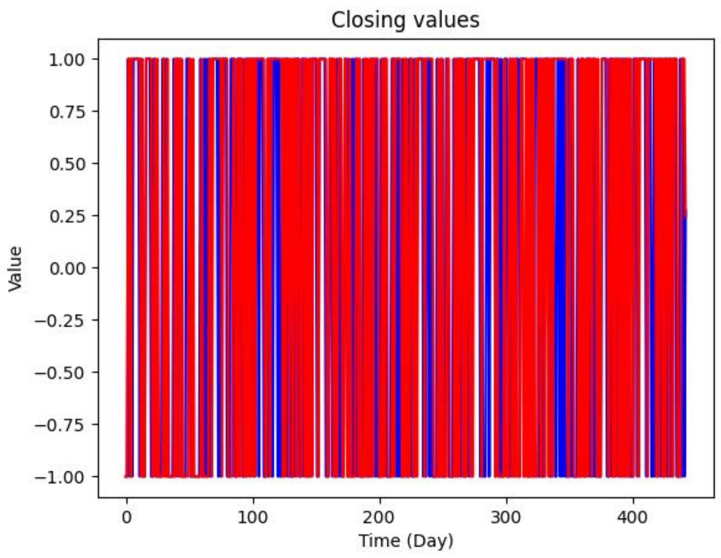


Fig. 4.1.9 Initial model's without chart patterns trading signal diagram. Red – predicted signal; Blue – real signal, X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

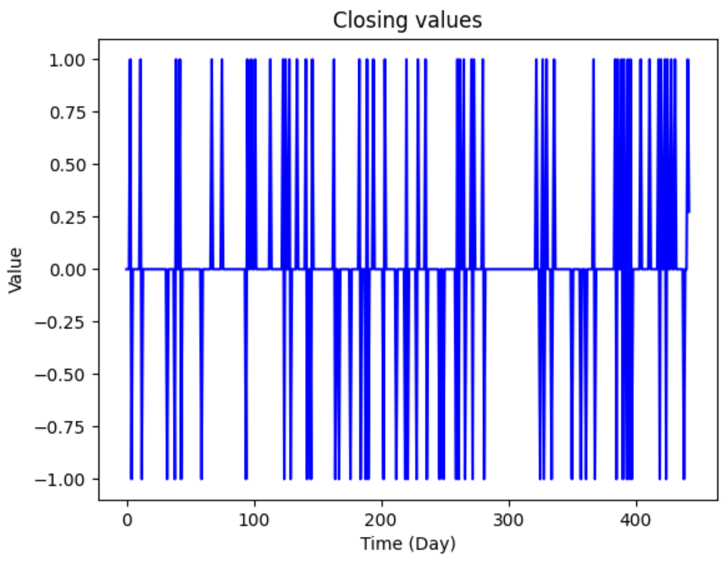


Fig. 4.1.10 Initial model's without chart patterns trade signal error. X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

From close price prediction graph and RMSE value it is apparent that the model without chart patterns had worse prediction accuracy than model that used chart patterns. However, from confusion matrix it is clear that model without chart patterns had slightly better buy signal predictions, while having worse sell signal predictions. Sell signals in general have worse prediction accuracy than buy signal accuracy and that is due to the lower number of examples of sell signals in the dataset. This is not done deliberately and is a byproduct of the economy that is mostly growing. There are still sell signal examples from daily price movements, but they cannot always be predicted from fundamental and technical analysis as they can be driven by emotional trades or individual preferences of traders.

4.2. Final model's testing data

To test preliminary conclusion, the final two models were trained. The models picked the same features as their previous iterations. Both models had a lookback period of 60 days, the model with chart patterns had penultimate dense layer size equal to 20. The model with chart patterns achieved RMSE value of 0.0169 and the loss graph for the model with chart patterns is shown in the figure 4.2.1.

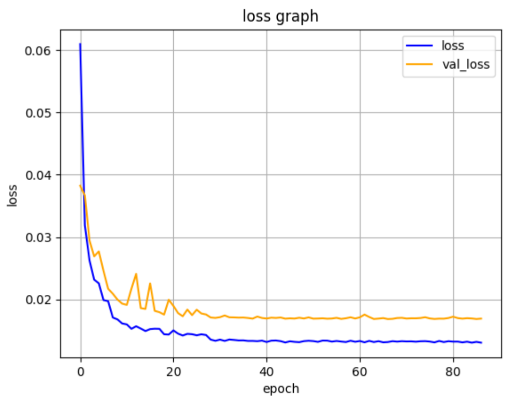


Fig. 4.2.1 Final model's with chart patterns loss diagram. Blue – training loss, Yellow – validation loss, X axis – epoch, Y axis – RMSE loss value

As can be seen from this graph, the downsizing of the model had positive effects on the overall training performance. Figures 4.2.2, 4.2.3, 4.2.4 and 4.2.5 show final model's with chart patterns metrics in the same order as previously shown results.

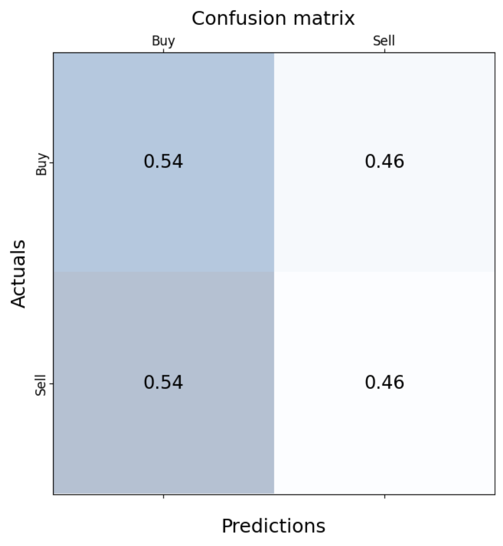


Fig. 4.2.2 Final model's with chart patterns trading signal confusion matrix. X axis – predicted trade signal, Y axis – actual trade signal

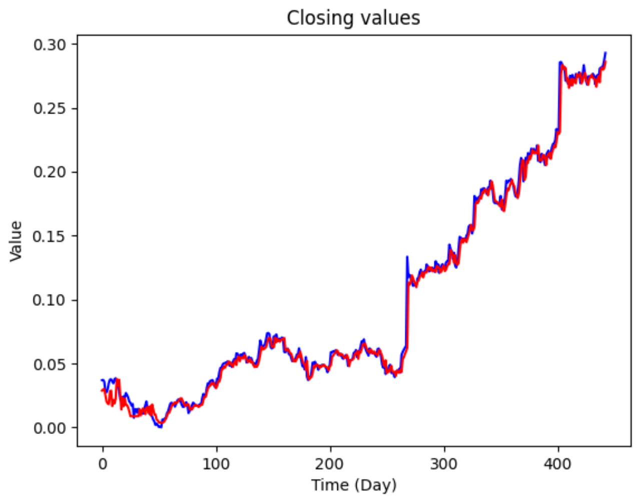


Fig. 4.2.3 Final model's with chart patterns closing price diagram. Red – predicted price; Blue – real price, X axis – time of sample, Y axis – normalized stock daily close price.

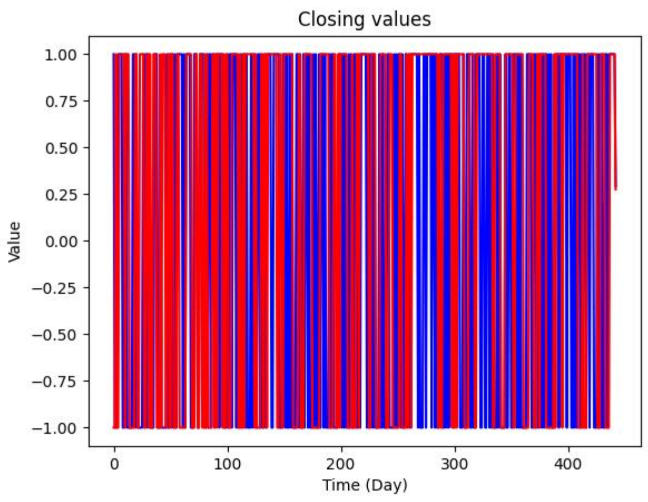


Fig. 4.2.4 Final model's with chart patterns trading signal diagram. Red – predicted signal; Blue – real signal, X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

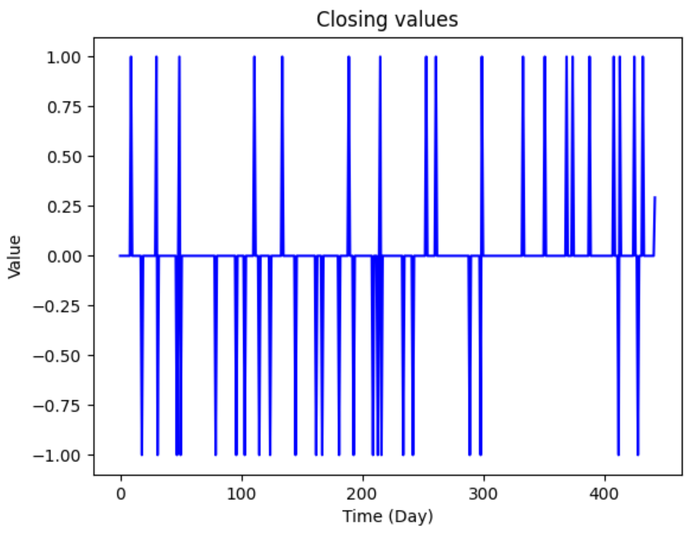


Fig. 4.2.5 Final model's with chart patterns trade signal error. X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

The confusion matrix for the final model with chart patterns shows improvement on the buy signal prediction accuracy, but it detects sell signals worse than its previous iteration. To measure this impact, accuracy metric was employed. The accuracy of this model was 49.9%. The model could not be used for trading as it could not accurately differentiate between the signals.

The model without chart patterns had penultimate dense layer's size equal to 5 and achieved RMSE value of 0.0182. Loss graph for this model is shown in the figure 4.2.6.

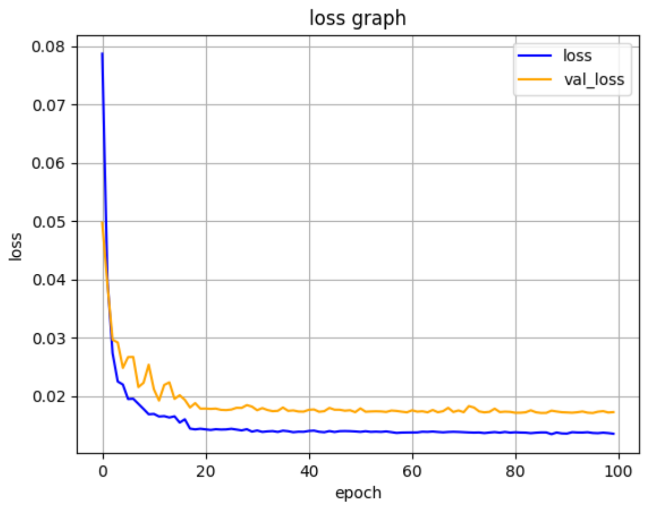


Fig. 4.2.6 Final model's without chart patterns loss diagram. Blue – training loss, Yellow – validation loss, X axis – epoch, Y axis – RMSE loss value

This graph is like the previous model in that it shows how downsizing helped the model overcome overfitting effects. The final model's without chart patterns metrics are shown in the figures 4.2.7, 4.2.8, 4.2.9, 4.2.10.

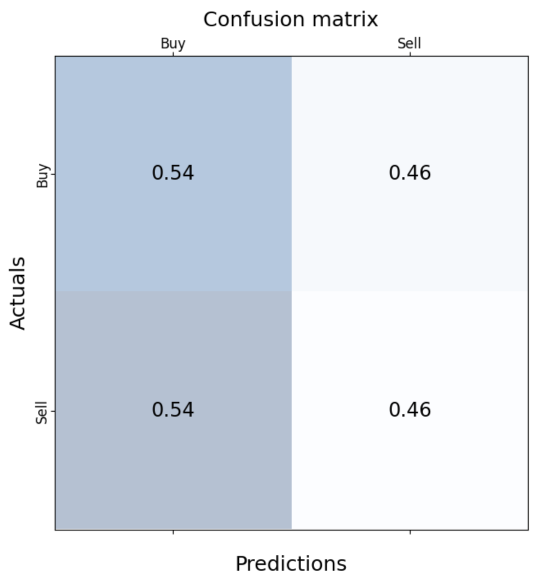


Fig. 4.2.7 Final model's without chart patterns trading signal confusion matrix. X axis – predicted trade signal, Y axis – actual trade signal

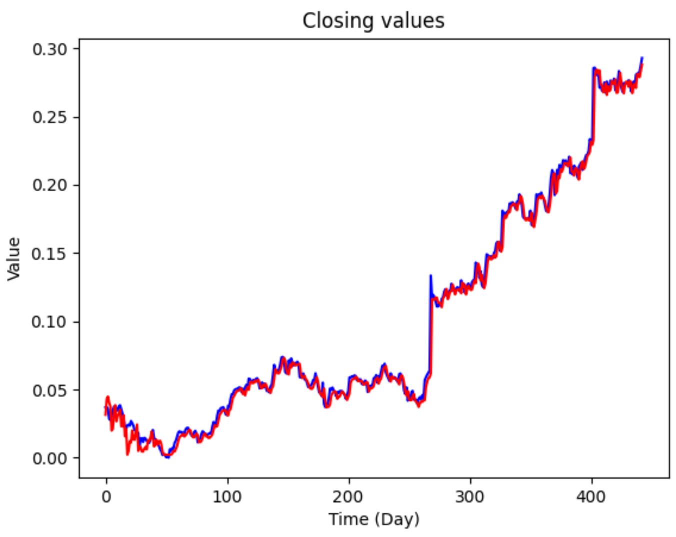


Fig. 4.2.8 Final model's without chart patterns closing price diagram. Red – predicted price; Blue – real price, X axis – time of sample, Y axis – normalized stock daily close price.

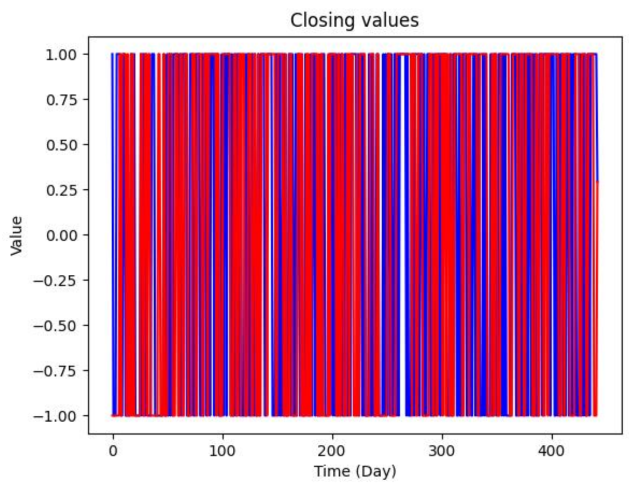


Fig. 4.2.9 Final model's without chart patterns trading signal diagram. Red – predicted signal; Blue – real signal, X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

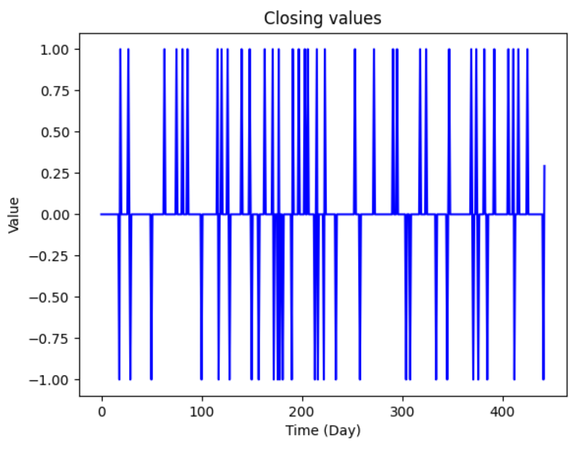


Fig. 4.2.10 Final model's without chart patterns trade signal error. X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

Final model without chart patterns achieved better sell signal recognition and price prediction accuracy compared to its previous iteration. The accuracy of the model is still a major issue as it is equal to 49.5%. The obtained results from final models further prove that there is no benefit in using chart patterns for the model training as the results are barely improved when comparing the two models. This slight difference is not significant to justify further use of chart patterns.

4.3. Ensemble testing data

The final way to improve model's trade signal classification performance was to combine the two final models into an ensemble. Confusion matrix, trade signal prediction and error metrics for this ensemble are shown in the figures 4.3.1, 4.3.2, 4.3.3, respectively. The accuracy of this ensemble was 67.5% and the trade action percentage loss due to inaction was equal to 36.5%.

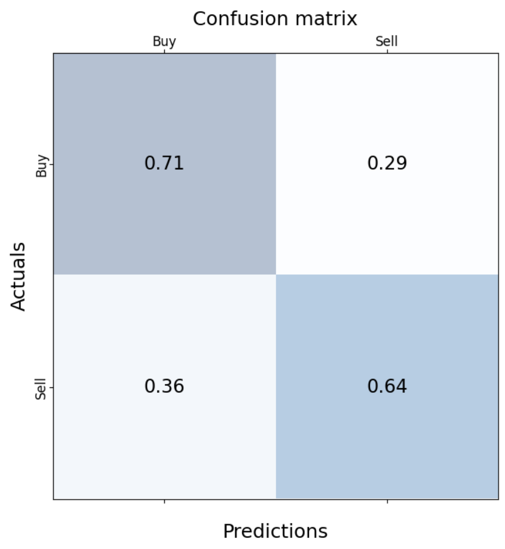


Fig. 4.3.1 Ensemble trading signal confusion matrix.

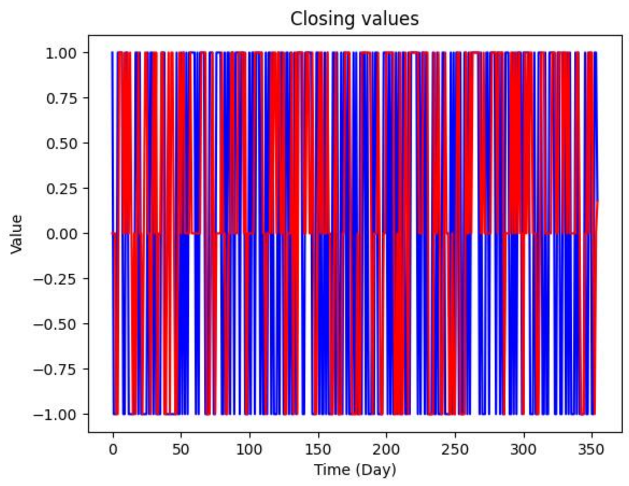


Fig. 4.3.2 Ensemble trading signal diagram. Red – predicted signal; Blue – real signal, X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

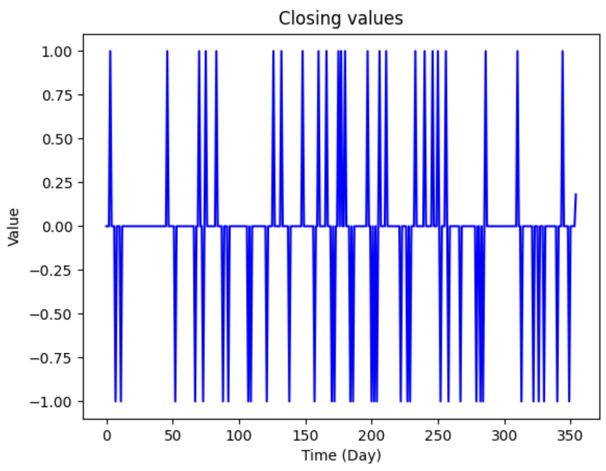


Fig. 4.3.3 Ensemble trade signal error. X axis - time of sample, Y axis – trade signal (1 – buy; -1 – sell).

Accuracy proves that ensemble provides better results than previous attempts with single model solutions. The only drawback of this implementation of ensemble was the lost trading actions which had to be considered. The inaction could be reduced utilizing predicted price change as a probability that the price will move in a given direction. This probability could also have a certain threshold for the model to take action. All this depends on the case-by-case basis and tolerance for a compromise between lost actions, accuracy and risk.

The preliminary testing showed that chart patterns did not improve the model's performance in a significant way. This was further proven by the results obtained from better iterations of the models. The method of combining the two models into an ensemble was the only significant improvement to the overall performance of models in question. All of this leads to a conclusion that chart patterns on their own do not improve model performance, unless used with other models. The underwhelming performance of chart patterns could be attributed to the fact that, as a semantic analysis tool, chart patterns rely on other people using this trading strategy. This also means that as a trading strategy it does not predict price movement accurately and consistently enough for this to work on its own. This could be either due to insufficient number of traders using this strategy or dataset containing multiple events that could not be foreseen with price movements alone.

Discussions

During the state-of-the-art research for this thesis, it was clear that there were no set trading strategies or the best practice for predicting the market. For this, the application of chart pattern feature maps was used to find chart patterns. They were later encoded into the dataset. The chosen GA-LSTM-CNN model provided automatic optimal parameter search and spatiotemporal analysis of the financial features. Finally, valuable insights into layer ordering, combinations of different models and combination impact on overall performance helped during this research to achieve better results from a baseline performance. Different performance metrics had to be evaluated according to their use cases as some metrics could have been inappropriate for this type of model and prediction type. The GA-LSTM-CNN model had LSTM layers before the CNN layers as it preserved the time-series time dimension feature patterns. The GA layer was used to control the number of LSTM unit count, lookback period and feature selection. For the experiments, multiple LSTM-CNN-GA models that could use chart patterns to predict next day's closing price were created. Combining models into an ensemble was found to be highly effective method of improving overall performance. Further work needs to be done in this area to pick the best models for the ensemble method to be optimal. In the future, difference between daily values should also be used to make model deployment easier in the real market. The main goal to test the effectiveness of chart patterns was achieved.

Conclusions and results

In this thesis, GA-LSTM-CNN models were trained with chart patterns and without them. This was done to compare the models and measure the impact that chart patterns have on the model's stock price predictions. To achieve all of this, the following tasks were carried out:

1. State of the art of relevant research was analysed. It was found that current research emphasizes model applicability in stock markets, improved prediction methods, models, and integration of novel data types or financial dataset representations.
2. The GA-LSTM-CNN model proved effective for feature selection and spatiotemporal analysis. Chart patterns were found using analytical rules. Valuable insights into model layer ordering were gained. Finally, examples of combining models into an ensemble to improve overall performance were used.
3. A dataset of 20 selected tickers (2013-07-08 to 2024-12-20) was taken from "Yahoo Finance". Dataset for the model training and testing consisted of fundamental, technical financial data and chart patterns. One dataset had chart patterns found using analytical rules and the other was without chart patterns. They were split into training and testing sets, testing was further split into testing and validation sets with a ratio of 80:20.
4. GA-LSTM-CNN model was made so that it could analyse the new dataset. The GA in this model could change model's structure like penultimate dense layer's neuron count, lookback period and features to use for model training. The model was trained and tested both with and without chart patterns to evaluate their influence on model stock market prediction performance. This evaluation was based on comparison between model performances. The results of training and testing were saved to generate metrics.
5. Model performance metrics were generated from saved training and testing results. Single model solution with chart patterns achieved RMSE score of 0.0169, without chart patterns achieved RMSE score was 0.0182. Chart patterns showed minimal impact when comparing their confusion matrices and accuracy metric which for both models was below 50%. The two model combination into an ensemble model achieved accuracy metric of 67.5%, but lost 36.5% trade actions. This is also reflected in confusion matrix where 71% of buy signals and 64% of sell signals were classified correctly. Overall, chart patterns did not have significant impact on model performance while the combination of the two models has had a great positive impact on model performance in the stock market.

List of references

[1] Y. Lin, S. Liu, H. Yang, and H. Wu, “Stock Trend Prediction Using Candlestick Charting and Ensemble Machine Learning Techniques with a Novelty Feature Engineering Scheme,” *IEEE Access*, vol. 9, 2021, doi: 10.1109/ACCESS.2021.3096825.

[2] Y. Lin, S. Liu, H. Yang, H. Wu, and B. Jiang, “Improving stock trading decisions based on pattern recognition using machine learning technology,” *PLoS One*, vol. 16, no. 8 August, 2021, doi: 10.1371/journal.pone.0255558.

[3] Y. Zheng, Y. W. Si, and R. Wong, “Feature extraction for chart pattern classification in financial time series,” *Knowl Inf Syst*, vol. 63, no. 7, 2021, doi: 10.1007/s10115-021-01569-1.

[4] N. Harish, H. Likith, G. Yashwanth, N. Krishnaswamy, Eshanya, and G. L. Sunil, “Stock Index Probability Prediction using the FB Prophet Model,” in *2022 International Conference on Futuristic Technologies, INCOFT 2022*, 2022. doi: 10.1109/INCOFT55651.2022.10094384.

[5] A. Maheshwari, A. Malhotra, S. Tuteja, M. Ranka, and M. S. A. Basha, “Prediction of Stock Prices using Prophet Model with Hyperparameters tuning,” in *2022 IEEE North Karnataka Subsection Flagship International Conference, NKCon 2022*, 2022. doi: 10.1109/NKCon56289.2022.10126796.

[6] F. V. Ferdinand, T. H. Santoso, and K. V. I. Saputra, “Performance Comparison Between Facebook Prophet and SARIMA on Indonesian Stock,” in *2023 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2023*, 2023. doi: 10.1109/IEEM58616.2023.10406940.

[7] V. Sarika, G. V. S. Kamal, S. V. Pratham, N. V. S. S. Deepak, and T. Veneela, “An LSTM-Based Model for Stock Price Prediction,” in *2023 Annual International Conference on Emerging Research Areas: International Conference on Intelligent Systems, AICERA/ICIS 2023*, 2023. doi: 10.1109/AICERA/ICIS59538.2023.10420270.

[8] S. Vohra and P. Savaridassan, “Stock Price Trend Analysis and Prediction of Closing Price Using LSTM,” in *2023 International Conference on Computer Communication and Informatics, ICCCI 2023*, 2023. doi: 10.1109/ICCCI56745.2023.10128260.

[9] O. S. Jadhav, S. P. Dhokare, S. Mote, P. Nistane, and P. G. School, “Estimation of Market Price through LSTM Model,” in *2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT)*, 2024, pp. 1148–1152. doi: 10.1109/IC2PCT60090.2024.10486384.

[10] X. Zhou, “Stock Price Prediction using Combined LSTM-CNN Model,” in *Proceedings - 2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence, MLBDBI 2021*, 2021. doi: 10.1109/MLBDBI54094.2021.00020.

[11] S. Nikhil, R. K. Sah, S. Kumar Parki, T. B. Tamang, D. Somashekhara Reddy, and T. R. Mahesh, “Stock Market Prediction Using Genetic Algorithm Assisted LSTM-CNN Hybrid Model,” in *2023 14th International Conference on Computing Communication and Networking Technologies, ICCCNT 2023*, 2023. doi: 10.1109/ICCCNT56998.2023.10306948.

[12] A. A. Varghese, J. Krishnadas, and R. S. Kumar, “Candlestick Chart Based Stock Analysis System using Ensemble Learning,” in *Proceedings of the 1st IEEE International Conference on Networking and Communications 2023, ICNWC 2023*, 2023. doi: 10.1109/ICNWC57852.2023.10127261.

[13] S. Verma, S. P. Sahu, and T. P. Sahu, “Stock Market Forecasting Using Hyperparameter-Tuned Ensemble Model,” in *Proceedings - 2023 IEEE World Conference on Applied Intelligence and Computing, AIC 2023*, 2023. doi: 10.1109/AIC57670.2023.10263934.

[14] Y. Hu, “An Ensemble Learning Model Integrating Short-term Trend and Long-term Trend Used in Stock Price Forecasting,” in *Proceedings - 2020 7th International Conference on Information Science and Control Engineering, ICISCE 2020*, 2020. doi: 10.1109/ICISCE50968.2020.00061.

[15] A. A. Semenoglou, E. Spiliotis, and V. Assimakopoulos, “Image-based time series forecasting: A deep convolutional neural network approach,” *Neural Networks*, vol. 157, 2023, doi: 10.1016/j.neunet.2022.10.006.

[16] Y. Qi, G. Guo, Y. Wang, and J. Yen, “Market Sentiment Analysis Based on Image Processing With Put-Call Volatility Gap Surface,” *IEEE Trans Comput Soc Syst*, vol. 11, no. 1, 2024, doi: 10.1109/TCSS.2022.3224054.

[17] Y. Qi, G. Guo, Y. Wang, and J. Yen, “Image Processing Based Implied Volatility Surface Analysis for Asset movement Forecasting,” in *IEEE International Conference on Industrial Informatics (INDIN)*, 2022. doi: 10.1109/INDIN51773.2022.9976175.

[18] G. Guo, Y. Qi, S. Lai, and J. Yen, “The Implied Volatility Surface Analysis Based Trading System,” in *2022 IEEE 2nd International Conference on Data Science and Computer Application, ICDSCA 2022*, 2022. doi: 10.1109/ICDSCA56264.2022.9987792.

[19] T. Sidogi, W. T. Mongwe, R. Mbuvha, and T. Marwala, “Creating Synthetic Volatility Surfaces using Generative Adversarial Networks with Static Arbitrage Loss Conditions,” in *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence, SSCI 2022*, 2022. doi: 10.1109/SSCI51031.2022.10022219.

[20] V. Suthiponpisal and D. Tancharoen, “A Comparative Evaluation of Noise Reduction Versus Data Normalization Techniques in Stock Market Prediction Using Transformer Models,” in *2024 8th International Conference on Information Technology (InCIT)*, 2024, pp. 775–780. doi: 10.1109/InCIT63192.2024.10810587.

[21] A. S. Rajpurohit, H. Mhaske, P. S. Gaikwad, S. P. Ahirrao, and N. B. Dhamale, “Data Preprocessing for Stock Price Prediction Using LSTM and Sentiment Analysis,” in *2023 6th International Conference on Information Systems and Computer Networks, ISCON 2023*, 2023. doi: 10.1109/ISCON57294.2023.10112026.

[22] Q. Chen and H. Kawashima, “Stock Price Prediction Using LLM-Based Sentiment Analysis,” in *2024 IEEE International Conference on Big Data (BigData)*, 2024, pp. 4846–4853. doi: 10.1109/BigData62323.2024.10825946.

[23] S. M. Mirjebreili, A. Solouki, H. Soltanalizadeh, and M. Sabokrou, “Multi-Task Transformer for Stock Market Trend Prediction,” in *2022 12th International Conference on Computer and Knowledge Engineering, ICCKE 2022*, 2022. doi: 10.1109/ICCKE57176.2022.9960122.

[24] J. Y. Lee and S. J. Yoo, “Stock Price Prediction Using Transformer and Time2Vec,” in *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 2025, pp. 687–689. doi: 10.1109/ICAIIC64266.2025.10920805.

[25] E. Gothai, R. Thamilselvan, P. Natesan, and K. Vignesh, “Global Stock Market Prediction Using Transformer-Based Deep Learning Techniques,” in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10724893.

[26] M. Faraz and H. Khaloozadeh, “Multi-step-ahead stock market prediction based on least squares generative adversarial network,” in *2020 28th Iranian Conference on Electrical Engineering, ICEE 2020*, 2020. doi: 10.1109/ICEE50131.2020.9260858.

[27] C. Dai, X. Yuan, Z. Tian, X. Hu, Z. Luan, and Y. Wang, “A Novel Wavelet Based Generative Model for Time Series Prediction,” in *2024 10th International Conference on Big Data and*

Information Analytics (BigDIA), 2024, pp. 178–182. doi: 10.1109/BigDIA63733.2024.10808510.

[28] L. Wang and Z. Huang, “Research on Stock Price Volatility Prediction Based on Generative Adversarial Network,” in *Proceedings - 2021 International Conference on Networking, Communications and Information Technology, NetCIT 2021*, 2021. doi: 10.1109/NetCIT54147.2021.00073.

[29] J. Gu, W. Du, and G. Wang, “RAGIC: Risk-Aware Generative Framework for Stock Interval Construction,” *IEEE Trans Knowl Data Eng*, vol. 37, no. 4, pp. 2085–2096, 2025, doi: 10.1109/TKDE.2025.3533492.