

Department of Computer Science at Wayne State University

Stock Market Analysis and Prediction

Project Report for CSC 5800

Muhammad Umar Khan
4-23-2025

Overview of the Problem

This project explores the prediction and analysis of the stock market based on historical trends and data. The stock market is a collection of markets where stocks and other forms of securities are bought and sold by investors to be invested in a foreseeable date and time. The stock market is a key component of the global economy, especially in today's modern world as it is considered the foundation of which for funding businesses growth and expansion. It is also the preliminary way individuals invest and grow wealth over a collective period of time. This project explores the numerous ways stock can be analyzed within the market for an investor or trader to evaluate and understand its performance and can help make key business decisions such as the buying and selling of the particular stock. The only problem, however, is the prediction of how the stock can affect the market in terms of pricing which makes investors and financial traders unable to understand which stocks should be prioritized to ensure the most optimized profits. Predicting prices on a long-term basis seems simple to understand, however considering how the market changes at an unprecedented rate on a daily basis makes the problem more complex and difficult to understand [2][7].

Stock market analysis and prediction refers to a machine learning practice that would help forecast future stock price movements and trends. This is an important study as the stock market is continuously evolving and shifting at an unpredictable pace and because of this, it is getting harder for investors, traders, and financial analysts to understand the typical performance of their stock and how it would affect the performance within the market. With the rise of several data mining algorithms and machine learning techniques, we can use those algorithms to help in aiding these problems of the prediction of the stock market by using the key data needed. However, predicting the prices of stocks within the stock market is not as simple as it sounds as if they are unpredictable. So, there are two general methodologies and methods used to analyze and understand stock market prices, which can then be used to create the data needed to implement various data mining algorithms and techniques; technical analysis and fundamental analysis [1].

Fundamental analysis focuses on evaluating the true, underlying value of a company—often referred to as its intrinsic value. This approach relies on a detailed examination of financial statements, economic indicators, interest rates, company leadership, and even qualitative factors like competitive advantage. The idea is that if an investor can understand the core financial health and potential of a business, they can identify stocks that are undervalued by the market. However, this method has limitations. It's more suited for long-term investing and can sometimes lead to overly optimistic assumptions. For instance, just because a company looks strong on paper doesn't mean market conditions won't shift rapidly and impact its value within hours or days. [5][6].

On the other hand, technical analysis takes a data-driven approach. It doesn't concern itself with a company's fundamentals, but rather focuses on trends and patterns in market activity—such as price changes, trading volume, and historical behavior. Tools like candlestick charts, moving averages, and momentum indicators are used to identify signals for buying or

selling. The core belief here is that historical market patterns tend to repeat themselves. While this method can be useful for short-term decision-making, it also draws criticism, especially from those who subscribe to the Efficient Market Hypothesis. This theory claims that all known information is already factored into current prices, making it hard to gain an edge through past data alone. [3][6].

While there is no concise answer for which method is best for gathering data, they provide an insight into how datasets can be differentiated and compiled depending on what factors are needed. This report will show the use of a particular dataset used in conjunction with several unique machine learning and data mining algorithms and techniques to understand and evaluate how we can use the information associated in the datasets with different parameters and determine the best method for the most elaborate and effective way to analyze and predict the stock market.

Dataset Overview Preprocessing and Visualization

Dataset Overview

The dataset chosen to analyze is the CNNPred dataset, a CNN-based stock market prediction using a diverse set of variables that can be found on the UCI Machine Learning Repository. This dataset contains 5 files and contains several daily features of S&P 500, NASDAQ Composite, Dow Jones Industrial Average, RUSSELL 2000, and NYSE composite. There are approximately 1985 instances with 84 features that is comprised from various categories of technical indicators, future contracts, price of commodities, important indices of markets around the world, price of major companies in the U.S. market and treasury bill rates. For this project, from the dataset, we will be focusing more on the file that has data on the S&P 500, which is considered one of the most important and widely followed stock market indices around the world, which is primarily used for index investing, algorithmic trading and market analysis.

Below, is a sample of some of the data that comes from this file.

Date	Close	Volume	ROC_5	EMA_10	CHF	XAG
2009-12-31	1115.10	NaN	NaN	NaN	-0.11	0.30
2010-01-04	1132.99	0.921723	NaN	NaN	-0.57	3.91
2010-01-05	1136.52	-0.3759	NaN	NaN	0.43	1.42

Dataset Preprocessing

Preprocessing the dataset to be used for our analysis wasn't complex or challenging but considering that there were an extraordinary amount of data that was either N/A or missing, it required a lot of cleaning onto the dataset for a more effective analysis and modeling as those

information would be considered irrelevant and we needed to clean and structure the dataset to make it more suitable for preprocessing, training and loading data into our respective algorithms.

Information that needs to be cleaned and structured includes reformatting of dates to proper format and sorted chronologically. Next up, any missing values that were found and identified were then removed alongside with omitting any NaN values that are presented in the dataset. This is to avoid complications during the initial analysis. For this project, both raw and feature engineered versions of this dataset was implemented and utilized. Working with both versions will allow us to see how much impact does enhancing the raw data and features impact prediction accuracy and model performance to be used later alongside our respective training models and classifiers.

Dataset Visualization

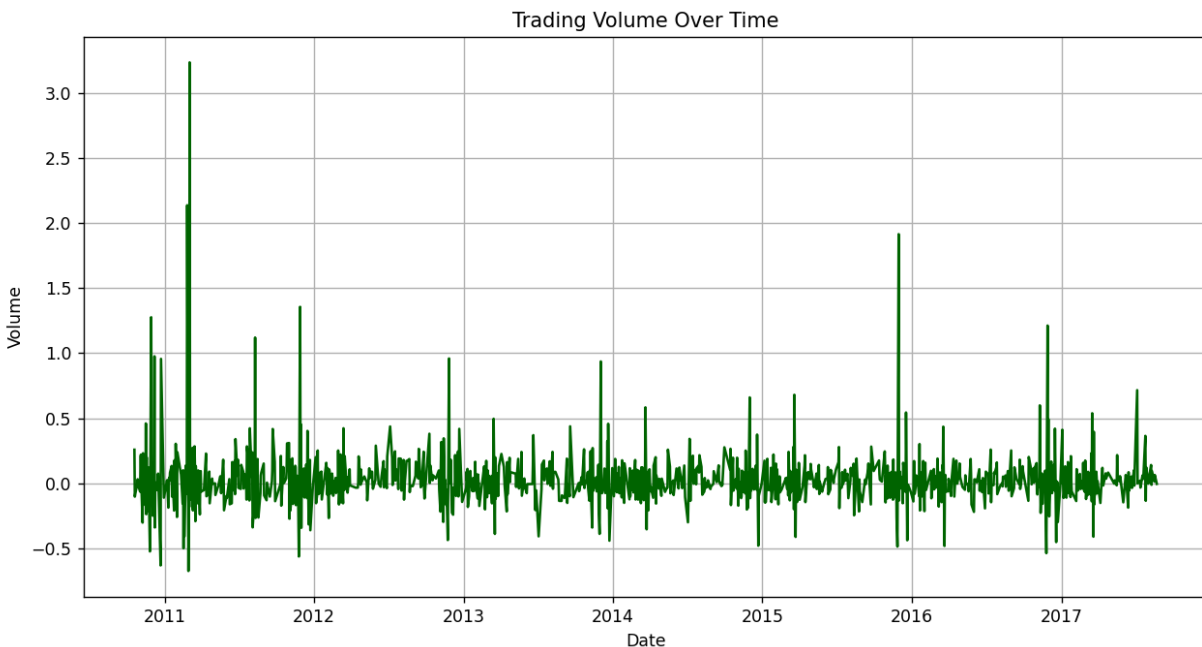
In python, we created several distinct kinds of line plots visualizations. These kinds of graphs were used to showcase what the dataset was going to bring for S&P 500, as well as understanding and gaining better insight into how the structured data was going to look like. Considering we were also implementing feature engineering to examine how well defined and structured variables would impact the data and model visualizations as opposed to regular raw defined variables, graphs and visualization techniques were also used to showcase their impact and behavior for the data. Below is a simple line plot that shows the daily closing price of the S&P 500 index.



This graph shows how the index moves on a daily basis based on the data reflected in the data set. As you can see, the close price generally increases over time, which suggests that the economy was generally strong during these times with sustainable growth. However, there are some sharp decreases every now and then which suggests these are when factors that can

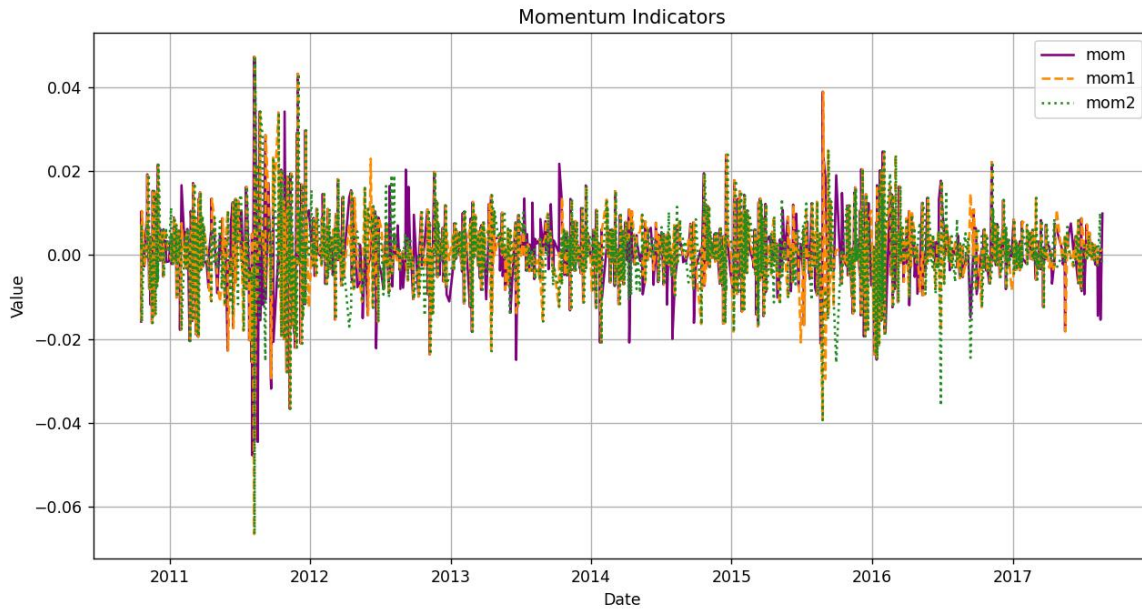
influence the market negatively have gone into effect such as interest rate hikes, financial crisis, or just in general economic events. This plot is crucial because the closing price would generally be the main target variable our algorithm would use for prediction. Observing the price over time on the market can help us understand the importance of change that happened during this time period and enable us to design and use models that captures short term and long-term trends.

Below is another line graph but it shows the trading volume over time, a principal factor when predicting and determining the stock price.



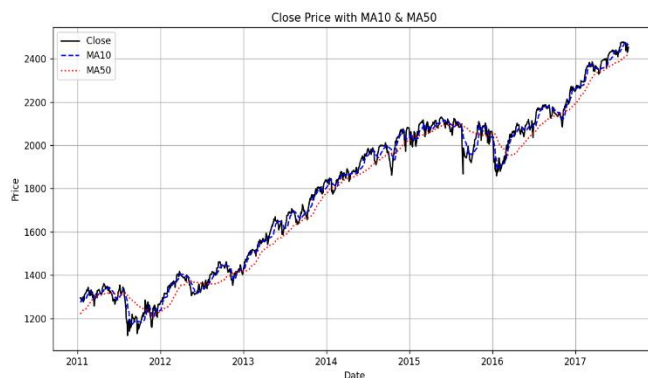
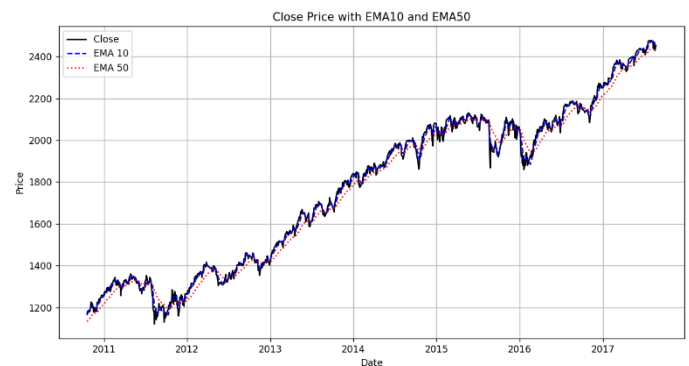
The trading volume essentially reflects the typical performance of the market and reflects traders and investors participation and activeness during these time periods. A high sharp spike as the ones observed before 2012 and just before 2016 signify that there was a substantial increase in trading activity, however it dropped just after that time period which indicates the market grown in fatigue or there was a panic in selling the stock right after traders lost interest in it. The trading volume is another feature that would be best observed and used by our models for better prediction, and to signify the importance of how a larger volume would be more meaningful and understand why there were low and high trading activity and why on certain days it increased or decreased.

Below shows the momentum of the data over different lookback periods to showcase and highlight the acceleration and the rate of change of price.



As seen from the graph, this shows how fast the value and price is accelerating over time periods. When the value goes above 0, this indicates that the price is rising which could signal the investors to start buying the stock for gaining a larger profit over time whereas when the value goes below 0, it indicates that the price is falling and signals investors to start selling their stock to minimize the loss of profits they would otherwise get if they didn't sell. With the 3 different momentum variables used in the graphs to capture different periods of momentums, we can use it to help gauge short term and long-term momentum shifts.

Below shows the close price using moving averages of 10 and 50 to showcase the trend direction of pricing using more defined and engineered data. There is also another graph to show the raw undefined variant using the standard ema 10 and 50 data.



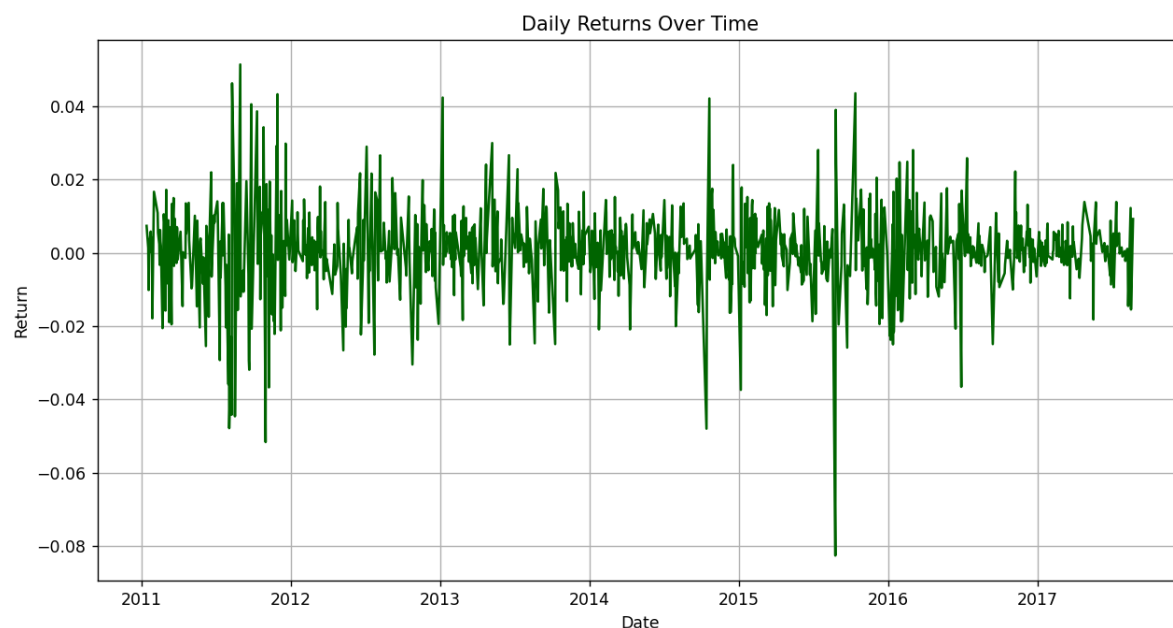
On the left:

It is similar to that of the beginning graph where it features close price of the dataset in terms of raw data. From this feature engineered data, MA10 and MA50 denoted as blue and red respectively with the original comparison data being black, we can see how the patterns closing prices fluctuate over time in short trends to mid and long trends.

On the right:

It is similar to the left side graph but using the raw defined data instead. The main comparison as seen by the graphs is that the featured engineered data showcases more long-term trend as opposed to short term trends, features a more lagging response as opposed to the raw data having higher and better response and the feature engineered data have later occurring crossovers while the other have earlier occurring crossovers. Another key difference is that the MA10 and MA50 captures trends better in general as the short-term data has been smoothed out while since EMA10 and EMA50 has both short term and long-term data intact, it gives an idea for how the momentum works from short term trends to long term trends.

Below is a graph that shows the daily returns over time. This represents the change in closing price from one day to the next. Generally, return are around the 0 mark to represent that there is a balance nature of gains and losses in the market over time so when the return is greater or lower than 0 by a significant mark, it usually would be considered an imbalance. Taking a look at the graph, the sharp spikes reflect high volatility days which signals these are times when the market could've crashed or other negatively impactful event that caused the return rate to spike versus general fluctuating zones in between consider the market calm or low volatile. This graph is useful as returns as a feature engineered defined data is important for predicting stock market prices using the data mining algorithms, but also this graph shows and helps models to detect speed and direction of price movement.



Algorithm Selection

To analyze the stock market and to help in predicting the future heading of stocks and assets in terms of trading history, pricing, and trends, it's important to select a data mining technique that would help in classifying the data with a proper and thorough analysis. That being

said, a lot of testing needed to be done with several algorithms. However, for this project, 2 algorithms in particular needed a closer look not because of their simplicity but because of their context that they were used in several other studies regarding the stock market analysis and prediction, so this would be serving as a way of building on top of that or expanding further.

Recurrent Neural Networks

A recurrent neural network is a kind of network that allows information to feed back into the system after each step rather than flowing in one direction from input to output as a standard neural network does. An output of an RNN depends on the memories or information that was used and processed from its previous steps as an RNN allows the network to remember or memorize past information by feeding the output from one step to the other instead of directly sending the information across the entire network in 1 motion. These are usually good to be used for recognizing patterns in data sequences such as time series or stock prices [9] [11][13].

RNNs process input sequences an element at a time as it uses a hidden state that captures information from previous elements. At each step, the network takes input x_t and previous hidden state a_{t-1} to compute current hidden state a_t and output y_t .

$$a_t = g_1(W_{aa}a_{t-1} + W_{ax}x_t + b_a)$$

$$y_t = g_2(W_{ya}a_t + b_y)$$

Where g_1 and g_2 are activation functions and W and b are weights and biases respectively shared across time steps [11].

However, the underlying issue with this technique is 3 very important factors crucial for this project of stock market prediction, those being short term memory problems, training complexity and the vanishing and exploding gradients problem, a rule in which when during training, gradients could become too small or large which hinders the learning capabilities. Fortunately, there is a type of RNN that solves these problems which is the kind of RNN that we will use for one of the implemented algorithms [11] [12].

Long Short-Term Memory Network (LSTM)

Long short-term memory networks are a kind of special type of recurrent neural network that uses what is called a “gate” to capture long term and short-term memory. An LSTM network is comprised of units called memory cells where each unit contains 3 gates that act as servers to regulate information flow. These 3 gates are as follows: the forget gate which determines which information from previous cell state should be discarded, the input gate which determines which new information should be added to the cell state and the output gate that controls the output based on the updated cell state. These 3 gates allow the network to maintain and update information over long sequences more effectively than a standard RNN does and also allows the same capabilities over short sequences [8][9][10].

When information passes through a cell, it first goes through the forget gate. The forget gate uses an equation to help in the decision-making process of whether we should keep information passed from previous time step or completely forget about it.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f)$$

Where h_{t-1} is the previous hidden state, x_t is the current input, W_f is the weight matrix, and b_f is the bias. The interpretation for how the gate determines whether or not the information should be kept depends if the output is 0 (that being f_t is 0), then forget that part of the cell state, otherwise if its 1, keep the information and move on to the input gate [11].

Next, right after the forget gate, we move on to the input gate that determines which new information should be added to the current cell state.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$$

$$\hat{C} = \tanh(W_C * [h_{t-1}, x_t] + b_C)$$

Where W_i , W_C , are the weight matrices, b_i , b_C , are the biases, and $\tanh$ is the hyperbolic tangent function. The synopsis of how these 2 functions work in the way in determining the new candidate values to be passed on to the output state. After filtering values to be remembered using h_{t-1} and x_t , a vector is formed using the $\tanh$ function which gives an output ranging from -1 to +1 that contains all possible values from h_{t-1} and x_t . After finding the input gate output, alongside creating the vector the values are then multiplied with the values found by the forget state function to create the updated candidate values[11].

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t$$

Where $\odot$ denotes element wise multiplication

The final gate is the output gate where we extract useful information from the current cell state to be presented to the next hidden state and soon to the rest of the network.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Where o_t is the output gate values, C_t is the current cell state which was determined after finding the updated candidate values of the input gate and h_t which is the new hidden state, as the information and output will be broadcasted to the next hidden state.

Using this kind of network, we can effectively eliminate the problems faced with standard RNN such as the vanishing and exploding gradients problem, the high computational requirements for training complexity, and the short-term memory problems as this network can handle both long term memory as well as short term memory. Due to those factors and problems being mitigated, this makes LSTM a viable contender to use as one of our algorithms for stock market prediction.

Support Vector Machines

Support Vector Machines (SVM) is a kind of classifier that has an aim goal to separate 2 classes of data and takes the form $f(x) = wx + b$, where w is the weight vector, x is the input vector and b is the bias term. This is used in finding the optimal hyperplane in an N -dimensional space to separate data points to difference classes to the maximum margin. In order to truly find the maximum margin, we use the formula $(wx_i + b)y_i \geq a$, to first get the margin and then use the formula $\max y = a/\|w\|$ where a is the margin projected onto w . However, it is easier to implement the method when the data can be linearly separable. At which point if the data is not linearly separable, then the SVM implements kernel functions to project and map the data into a higher dimensional space where it could become linearly separable. Some kernels used include polynomial functions, radial basis function (which is also considered as Gaussian or RBF) and sigmoid kernel functions [14] [15] [16].

Analysis Results and Comparison

We used the 2 algorithms of the LSTM and SVM for our testing and experiment to determine which is the best algorithm to predict the stock market price for better clarity. The reason why these 2 algorithms were chosen is because looking at from previous reports and gaining a better understanding of how several methods and algorithms work and what they look for when it comes to training and testing and implementing, it was determined that Support Vector Machines and the Long Short Term Memory model of RNN was the best at stock market analysis and prediction.

The problem with this dataset that we have chosen serves more as a dataset for regression algorithms and not classifiers. That being said, for LSTM we were able to experiment and create a regression and classifier model and use the respective metrics to determine the best hyperparameters. For the LSTM model, the main target feature was the close price feature as this is the main one, we want to take a closer look at and examine to determine the best stock market price as the close price influences the stock market.

Below is a base default model using standard parameters and comparison graph of the actual vs predicted using the model. The parameters used were:

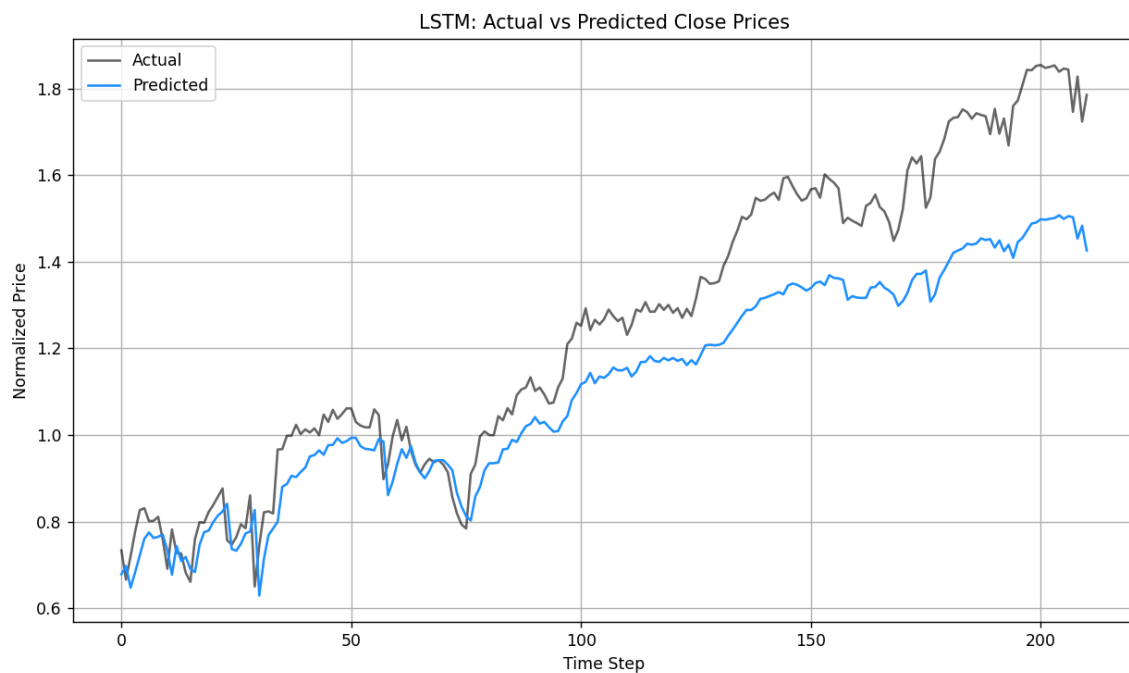
- Look back: 10.
- Units: 64
- Dropout: 0.3
- Epochs: 50 (set in stone)
- Batch size: 32.
- Learning rate: 0.001



Taking a closer look at the graph, we can see that using the prediction model, we were able to get a close match to the actual and original raw dataset. This shows getting as close as possible to the raw data, this is an excellent choice of parameters, however, note the fact that it starts strong in the beginning, and in the long term it starts to fade off.

Now, below we have according to the classifier as the best parameters for classification on LSTM model. The parameters used here were:

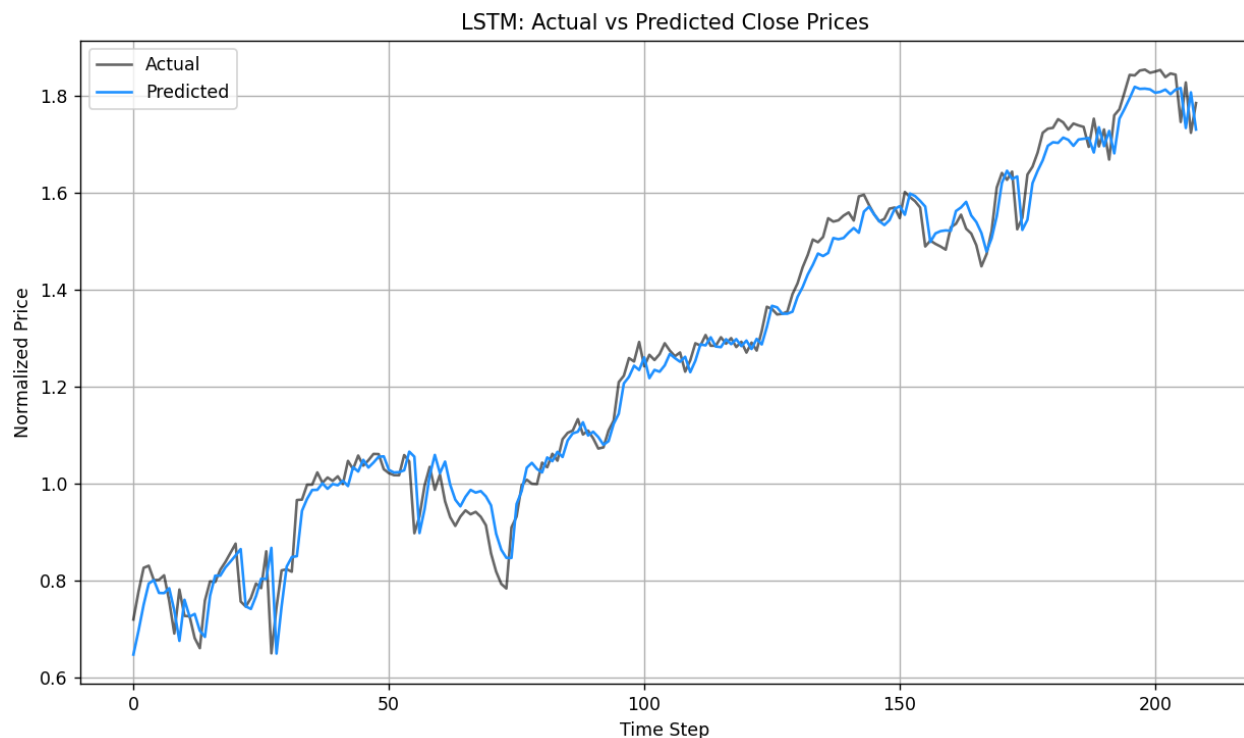
- Look back: 10.
- Units: 48
- Dropout: 0.1
- Epochs: 50
- Batch size: 32.
- Learning rate: 0.01



Taking a look at the graph, we can see just simply changing the parameters slightly even though from our classification testing where this was proven the best, it was worse than the previous model as the prediction for the long-term trend drops much greater and much sooner. This shows that while these hyperparameters may be good on a simple classification style, on a regression data style which is how this dataset is normally formatted as, its not as good. In fact, it's even worse.

Now, below we have according to the regression as the best parameters for regression on LSTM model. The parameters used here were:

- Look back: 20.
- Units: 64
- Dropout: 0.2
- Epochs: 50
- Batch size: 16.
- Learning rate: 0.001



As seen from the graph above, this is definitely the closest and far better model we have yet that has the closest prediction to the actual result, indicating that these hyperparameters are better suited for predicting the price. Especially as the problems faced in the previous test where the prediction falls off for the long-term trend is fixed with these hyperparameters.

Below is a simple comparison of all LSTM parameters with comparison by regression and classification metrics. It should be noted that these testing of regression and classification were done on separate programs with classification done on a nuanced dataset version of the original that supports classification, which is why although we have the best parameters by

classification, the model that has the best parameters by regression have better classification metrics such as higher accuracy. Also, the hyperparameters for classification and regression were chosen by implementing a grid search to look through all possible combinations of specified hyperparameters and to choose the one with the best results.

	Parameters	Metrics comparison (by regression)	Metrics comparison (by classification)				
LSTM (standard default by regression)	<ul style="list-style-type: none">- Look back: 10.- Units: 64- Dropout: 0.3- Epochs: 50- Batch size: 32.- Learning rate: 0.001	Root means squared error: 0.1258. Means average error: 0.1099. R^2 : 0.8615	Accuracy:51.13% Precision: 69.23% Recall: 21.95% F1-score:33.33% Confusion matrix: <table><tr><td>86</td><td>12</td></tr><tr><td>96</td><td>27</td></tr></table>	86	12	96	27
86	12						
96	27						
LSTM (classification best parameter)	<ul style="list-style-type: none">- Look back: 10.- Units: 48- Dropout: 0.1- Epochs: 50- Batch size: 32.- Learning rate: 0.01	Root means squared error: 0.1775. Means average error: 0.1468. R^2 : 0.7358	Accuracy:53.39% Precision: 59.09% Recall: 52.85% F1-score:55.79% Confusion matrix: <table><tr><td>53</td><td>45</td></tr><tr><td>58</td><td>65</td></tr></table>	53	45	58	65
53	45						
58	65						
LSTM (regression best parameter)	<ul style="list-style-type: none">- Look back: 20.- Units: 64- Dropout: 0.2- Epochs: 50- Batch size: 16.- Learning rate: 0.001	Root means squared error: 0.0452. Means average error: 0.0347. R^2 : 0.9826	Accuracy:56.62% Precision: 56.72% Recall: 93.44% F1-score:70.59% Confusion matrix: <table><tr><td>10</td><td>87</td></tr><tr><td>8</td><td>114</td></tr></table>	10	87	8	114
10	87						
8	114						

Support vector machine faced a similar problem when trying to load and prepare our data as it was more suitable for regression. However, as the method we used to implement the data for classification in LSTM, we implemented the same thing for SVM. As far as results go, Support vector machines was definitely the far better choice when compared to LSTM.

We first used WEKA to compare and find the best results for SVM. For the dataset, we could only use SMOReg which is the regression variant of SVM, which to use the classification variant of SVM, we created a cleaner dataset from the original by discretizing continuous output variables into categorical classes including 'up' and 'down.' The parameters we changed to find

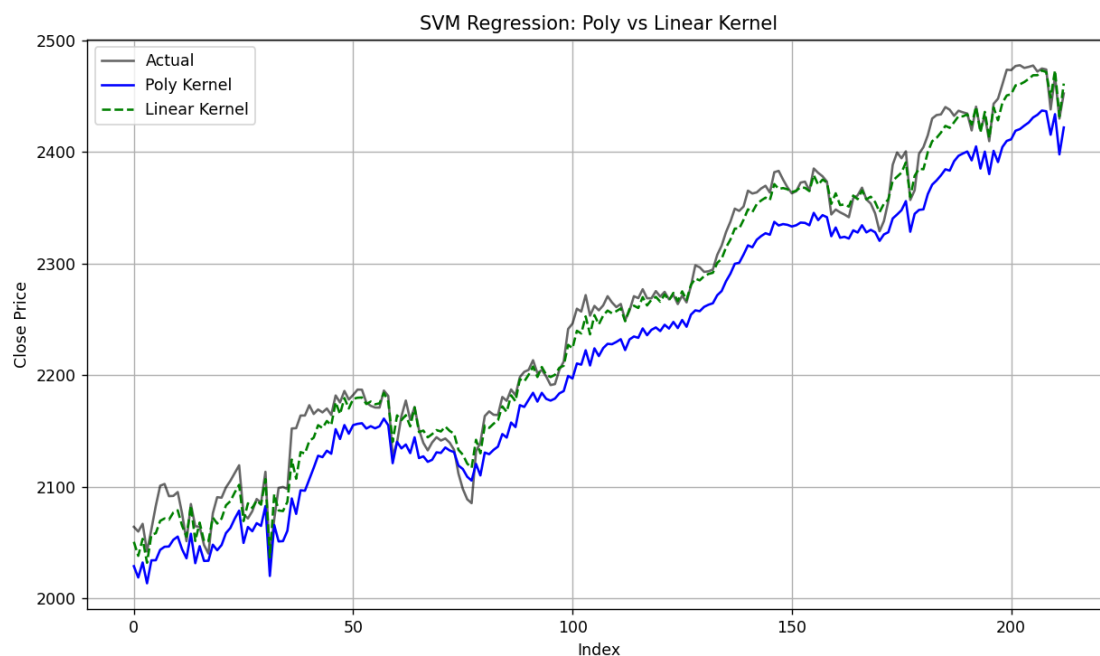
the best results for both variants of SVM were c (regularization parameter), kernel (the tool to organize information based on linearity) and gamma (controls influence of training examples on decision boundary (used only when kernel is rbf for non-linear data)). For both variants, we compared and found the best parameter for when the kernel is set to linear and nonlinear (poly and rbf respectively) and compared them to standard default parameters. It should be noted that this was done the dataset when it was raw and unfiltered just to get a base view glance as well as the dataset being trained on 10-fold cross validation.

SVM parameters and modes	Regression metrics	Classification metrics				
C: 1.0, kernel: polykernel (default and linear)	Root means squared error: 0.5378. Means average error: 0.3898. R ² : 52.9249%	Accuracy: 91.5493% Error rate: 8.4507% TP Rate: 91.5% FP Rate: 9.0% Precision: 91.6% Recall: 91.5% F-measure: 91.5% Confusion matrix: <table><tr><td>555</td><td>36</td></tr><tr><td>54</td><td>420</td></tr></table>	555	36	54	420
555	36					
54	420					
C: 100, kernel: rbfkernel, gamma: 0.01 (best for classification)	For comparison: Root means squared error: 0.5335. Means average error: 0.3863. R ² : 51.8694%	Accuracy: 90.6103% Error rate: 9.3897% TP Rate: 90.6% FP Rate: 10.3% Precision: 90.7% Recall: 90.6% F-measure: 90.6% Confusion matrix: <table><tr><td>558</td><td>33</td></tr><tr><td>67</td><td>407</td></tr></table>	558	33	67	407
558	33					
67	407					
C: 10, kernel: rbfkernel, gamma: 0.01 (best for regression)	Root means squared error: 0.5334. Means average error: 0.3862. R ² : 51.864%	For comparison: Accuracy: 87.6695% Error rate: 12.3005% TP Rate: 87.7% FP Rate: 13.5% Precision: 87.8% Recall: 87.7% F-measure: 87.6% Confusion matrix: <table><tr><td>547</td><td>44</td></tr><tr><td>83</td><td>387</td></tr></table>	547	44	83	387
547	44					
83	387					

Now we compare these parameters with the default and best parameters found when coding the SVM model in Python. It should be noted that in WEKA, a linear kernel is stated as

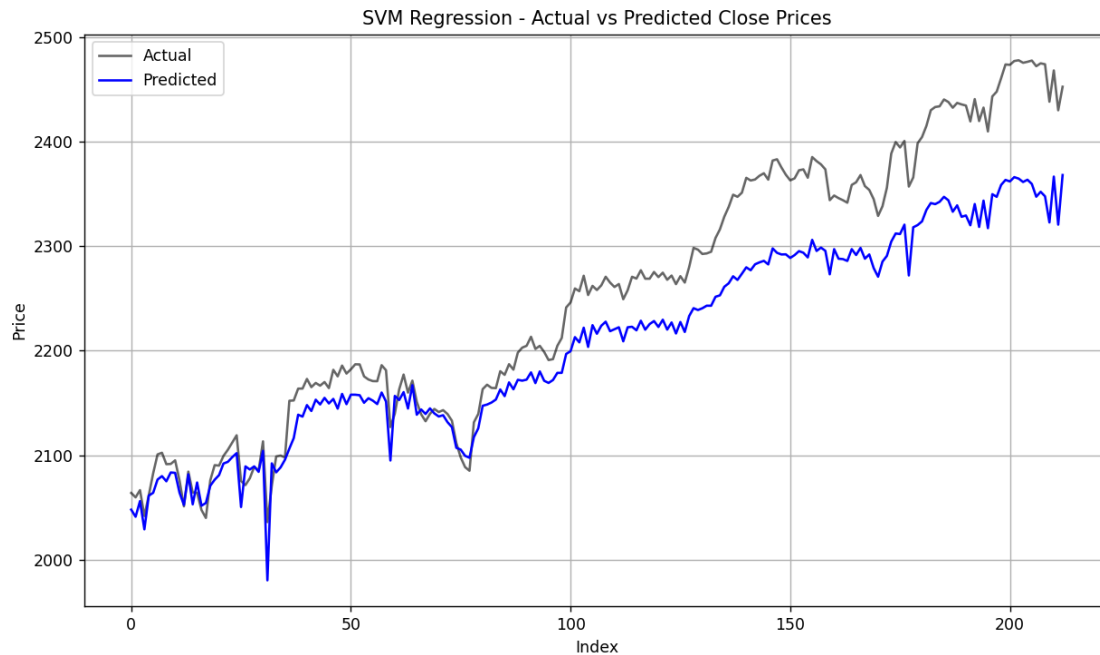
poly with degree 1 while in Python, while that can be used for linear data, it also has a true linear kernel. So, for the standard default parameters (especially as seen with the first entry in the table above), we will compare the linear and poly kernel (with degree = 1) for comparison on our dataset before we showcase the best parameters.

	Poly kernel	Linear kernel								
SVM regression	RMSE : 36.5735 MAE : 33.8123 R-squared : 0.9178	RMSE : 13.1444 MAE : 10.3704 R-squared : 0.9894								
SVM classification	Accuracy: 97.65% Precision: 99.15% Recall: 96.67% F-measure: 97.89% Confusion matrix: <table><tr><td>92</td><td>4</td></tr><tr><td>1</td><td>116</td></tr></table>	92	4	1	116	Accuracy: 96.71% Precision: 100% Recall: 94.17% F-measure: 97.0% Confusion matrix: <table><tr><td>93</td><td>0</td></tr><tr><td>7</td><td>113</td></tr></table>	93	0	7	113
92	4									
1	116									
93	0									
7	113									



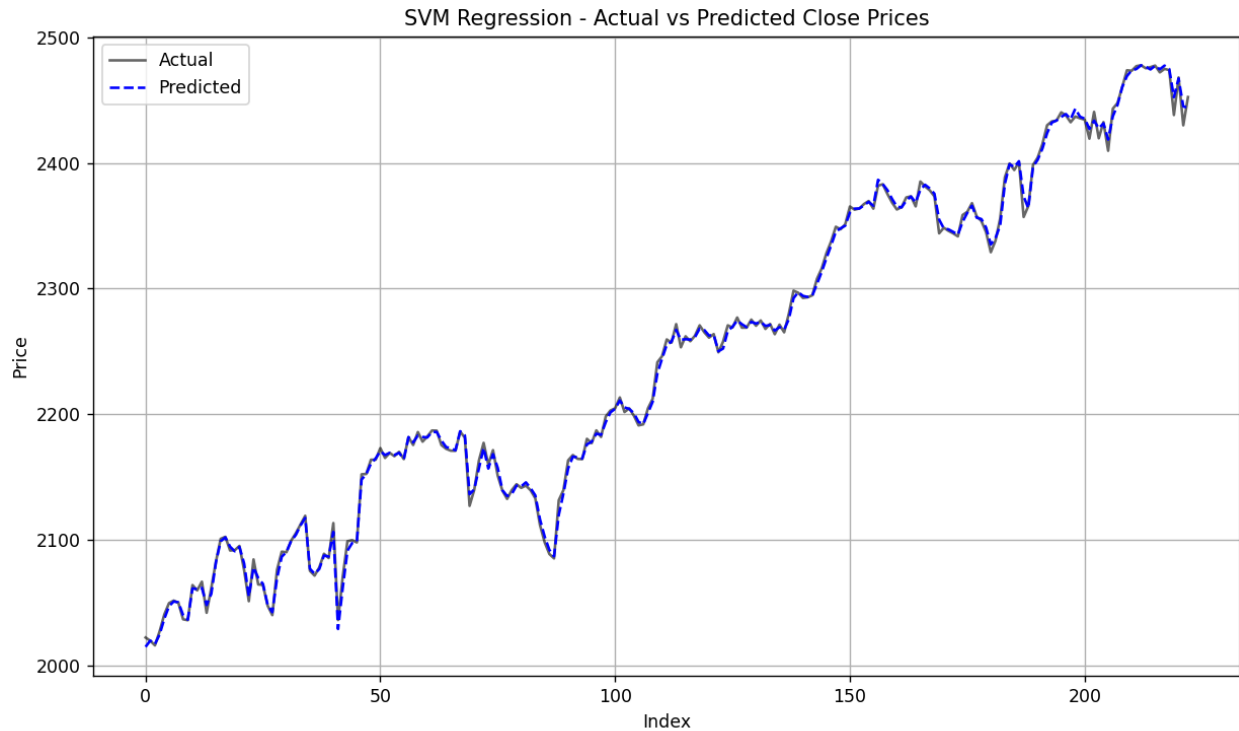
From looking at the information given in the table as well as the graph above that visualizes the comparison of the actual output alongside the models of the linear and poly kernel, we can see that the linear data is much more closer than the poly data which suggests using a linear kernel is better for our project in terms of predicting price as opposed to poly kernel as suggested with WEKA.

Implementing a grid search on our program we first look through different parameters while having the kernel set to 'rbf.' From our testing, the best parameters on SVM for both regression and classification on a rbf kernel is: C: 100, and gamma: 0.01 (and scale on the classification)



As seen from the graph above, a rbf kernel with the following parameters yielded a result similarly seen by the LSTM model. This suggests that when the kernel is set to rbf (that being non-linear), we will get the expected output of LSTM which makes either of the 2 redundant to test on. However, we also implemented a grid search to find the best parameters including poly and linear kernel.

SVM Parameters	Classification Metrics	Regression Metrics				
C: 100 Kernel: linear Gamma: scale	Accuracy: 99.1% Precision: 99.2% Recall: 99.2% F-measure: 99.2% Confusion matrix: <table><tr><td>98</td><td>1</td></tr><tr><td>1</td><td>123</td></tr></table>	98	1	1	123	Root means squared error: 3.8805. Means average error: 2.7863. R ² : 0.9991
98	1					
1	123					



The result is that we get a prediction that is almost near spot on. This data and graph suggests that from our testing and comparison of all parameters on both algorithmic models of LSTM and SVM, SVM has the closest score and prediction results, making it the far better choice for predicting the stock market price. On top of that, SVM for this graph has a linear kernel, suggesting that the data from the beginning (and with feature engineering and filter cleaning) is more likely a linear trend of data which makes SVM the better choice as it can support linear data.

Below is a proper comparison of the best LSTM model with parameters and SVM model with parameters and metric comparison.

Model	Parameters	Accuracy	Precision	F1-Score	Recall	Confusion Matrix		RMSE	MAE	R ²
Support Vector Machine	C: 100, Kernel: linear, Gamma: scale	99.1%	99.2%	99.2%	99.2%	98	1	3.8805	2.7863	0.9991
						1	123			
Long Short-Term Memory	Look back: 20, Units: 64, Dropout: 0.2 Epochs: 50 Batch size: 16. Learning rate: 0.001	56.62%	56.72%	0.7059	93.44%	10	87	0.0452	0.0347	0.9826
						8	114			

Conclusion

By leveraging the CNNPred Dataset and focusing entirely on the S&P 500 index which is a popular stock market index and useful for understanding the problem of stock market prediction and analysis as it covers the vast majority of the stock market, we were able to implement and experiment with 2 prominent models: Long Short Term Memory and Support Vector Machines (SVM), to gain insight into valuable data to understand how to predict the market price better.

The experiments performed with LSTM were well-adequate and performed as expected. The analysis of running the model on the data definitely shows how accurately it performs in the short trend but fails to comply with the longer trends as it doesn't hold up. However, with some proper tuning adjustments to the hyperparameters, the model starts to make up for the long trends and becomes better and more accurate to the original actual data, making the prediction greater. However, it isn't without limitations and when it comes to classification accuracy and training complexity, it becomes a letdown which is normally to be expected as a more regression-based algorithm. Nonetheless, it was still a solid model and algorithm that provided insight into how the data behaves.

On the other hand, the experiments done with the SVM models exceeded expectations as the hyperparameters and the model itself was more directed into learning about the dataset and the close price feature more prominently. The ability to have hyperparameters adjusted accordingly based on the linearity of the data gives it an edge over the LSTM to understand how the dataset is behaved, and how to predict prices more accurately. With the use of this model, we were able to understand that the data is sorted in a more linear matter as opposed to a non-linear way, and because we were able to tell its linearity, this makes it easier to play with the other hyperparameters to get our predictions as close as possible to the actual result. In comparison, the SVM model was definitely the far better choice and option, especially for the S&P 500 index data file we chosen.

Overall, this study helped us understand these algorithms on the dataset and helped to understand the importance of stock market analysis and prediction and which models to implement and which to choose based on certain criterion and conditions. Looking ahead, for better testing, we could have used other dataset files, not just the other data files present in the CNNPred dataset related to the S&P 500 Index, but data files that contains other kinds of classes found in other datasets that were used for this topic study. This would allow our model to work in a more dynamic situation, having to adjust accordingly to different data presented in other unique data files and understand how they perform or perhaps test data based on other features and not just the close price, which is considered the main feature to look and compare for stock market analysis and prediction.

References

Corporate Finance Institute. (n.d.). *Stock analysis*.

<https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/stock-analysis/> [1]

Nature Portfolio. (2025). *Stock market prediction using machine learning*. *Scientific Reports*.

<https://www.nature.com/articles/s41598-025-88734-6> [2]

Capital.com. (n.d.). *Technical analysis definition*. <https://capital.com/technical-analysis-definition> [3]

Capital.com. (n.d.). *Stock market prediction definition*. <https://capital.com/stock-market-prediction-definition#:~:text=A%20stock%20market%20prediction%20is,a%20combination%20of%20the%20two.> [4]

Capital.com. (n.d.). *Fundamental analysis definition*. <https://capital.com/fundamental-analysis-definition> [5]

Investopedia. (n.d.). *What is the difference between fundamental and technical analysis?* <https://www.investopedia.com/ask/answers/difference-between-fundamental-and-technical-analysis/> [6]

Analytics Vidhya. (2021, October). *Machine learning for stock market prediction with step-by-step implementation*. https://www.analyticsvidhya.com/blog/2021/10/machine-learning-for-stock-market-prediction-with-step-by-step-implementation/#Problem_Statement_for_Stock_Market_Prediction [7]

ScienceDirect. (n.d.). *Long short-term memory network*. [https://www.sciencedirect.com/topics/computer-science/long-short-term-memory-network#:~:text=Long%20Short%2DTerm%20Memory%20\(LSTM\)%20is%20a%20type%20of,used%20in%20memory%2Dbased%20problems.](https://www.sciencedirect.com/topics/computer-science/long-short-term-memory-network#:~:text=Long%20Short%2DTerm%20Memory%20(LSTM)%20is%20a%20type%20of,used%20in%20memory%2Dbased%20problems.) [8]

GeeksforGeeks. (n.d.). *Deep learning – Introduction to long short-term memory*. <https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/> [9]

Analytics Vidhya. (2021, March). *Introduction to long short-term memory (LSTM)*. <https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/> [10]

GeeksforGeeks. (n.d.). *Introduction to recurrent neural network*. <https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/> [11]

GeeksforGeeks. (n.d.). *Vanishing and exploding gradients problems in deep learning*. <https://www.geeksforgeeks.org/vanishing-and-exploding-gradients-problems-in-deep-learning/> [12]

IBM. (n.d.). *Recurrent neural networks*. <https://www.ibm.com/think/topics/recurrent-neural-networks#:~:text=A%20recurrent%20neural%20network%20or,speech%20recognition%20and%20image%20captioning>. [13]

GeeksforGeeks. (n.d.). *Support vector machine algorithm*. <https://www.geeksforgeeks.org/support-vector-machine-algorithm/> [14]

IBM. (n.d.). *Support vector machine*. <https://www.ibm.com/think/topics/support-vector-machine> [15]

Analytics Vidhya. (2021, October). *Support vector machines (SVM) – A complete guide for beginners*. [https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#:~:text=Frequently%20Asked%20Questions-,What%20is%20a%20Support%20Vector%20Machine\(SVM\)?,SVM%20helps%20classify%20data%20effectively](https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/#:~:text=Frequently%20Asked%20Questions-,What%20is%20a%20Support%20Vector%20Machine(SVM)?,SVM%20helps%20classify%20data%20effectively). [16]