Machine Learning for Predicting Price Movements of Stocks

Adityasai Koneru

Dr. Romesh Saigal

University of Michigan

College of Engineering Honors Program

Abstract

Currently, about 80% of the stock market trades are from automated systems. With the rise of systematic trading and individual investing, there is significant opportunity to apply novel machine learning techniques to this space. Since stocks are incredibly volatile, even the slightest advantage in predicting their movement is helpful for financial analysts and real-world traders.

The purpose of this project is to design and test various machine learning algorithms in predicting price movement of an asset. To simplify the problem for feasibility purposes, my goal was to accurately predict the percent change in price in the next timestep given a specific lookback period. Since stock prices can be defined as a time-series function, I mainly focused on using Recurrent Neural Networks (RNNs) given their proficiency in time-based predictions. I combined this network with technical indicators which are currently used by traders in the real world to refine this prediction.

Introduction

There are two key interesting facets in this project. The first focus of this project is exploring the potential of recurrent neural networks (RNNs). This technology is a growing subsection of neural networks. Due to the recurrent structure in which inputs are used as future outputs, RNNs are rendered as ideal options for time-series analysis. Since the stock price of an asset can be boiled down into a time-series function, this leads us to the second pillar of our project.

In addition to exploring novel neural network techniques, this project focuses on understanding the complexities and intricate factors which can influence the stock market. Over the past two years, the COVID-19 pandemic has left many aspects of our daily lives incredibly unpredictable. As a result of this pandemic, the stock market has been more volatile in recent years than ever before. Thus, it has become even more important to have a stable prediction of how the market is expected to react.

By combining these two interests, I designed my Engineering Honors Capstone project. While exploring prediction methods for the stock market, I decided to draw from my coursework and previous artificial intelligence experience. Since recurrent neural network structures were already well used for time-series analysis, I decided to explore how RNNs could be used in predicting the movement of stock prices. My goal for this project is to understand if certain stock price movements can be predicted and if recurrent neural network layers are an effective choice for a prediction algorithm.

Financial Background

As one of the pillars of this project, it was important to understand a core principles of how real-world professional traders value assets currently. Trader work for financial institutions and seek profit from buying and selling financial instruments such as stocks, derivative contracts, and bonds. Currently, there is a significant amount of technology that is used in the trading industry. As of 2017, over 80% of market trades were completed by an automated technology.

Despite this incredible technology influence, the use of machine learning in the finance industry is very limited. Often, these algorithms are too slow for short-term day trading. In order to refine my machine learning approach to this problem, I sought to incorporate current finance insights into my model.

A common tool for professional traders are financial technical indicators. Technical indicators are metrics which reduce price trends and patterns into single values. Each indicator focuses on a different aspect of price trends.

For example, the Relative Strength Index (RSI) is a momentum indicator which focuses on the overbought/ oversold condition of a stock. This metric is on a scale for 0 to 100 where under 30 represents oversold and over 70 represents overbought conditions. Here is an example graph representation of technical indicators:

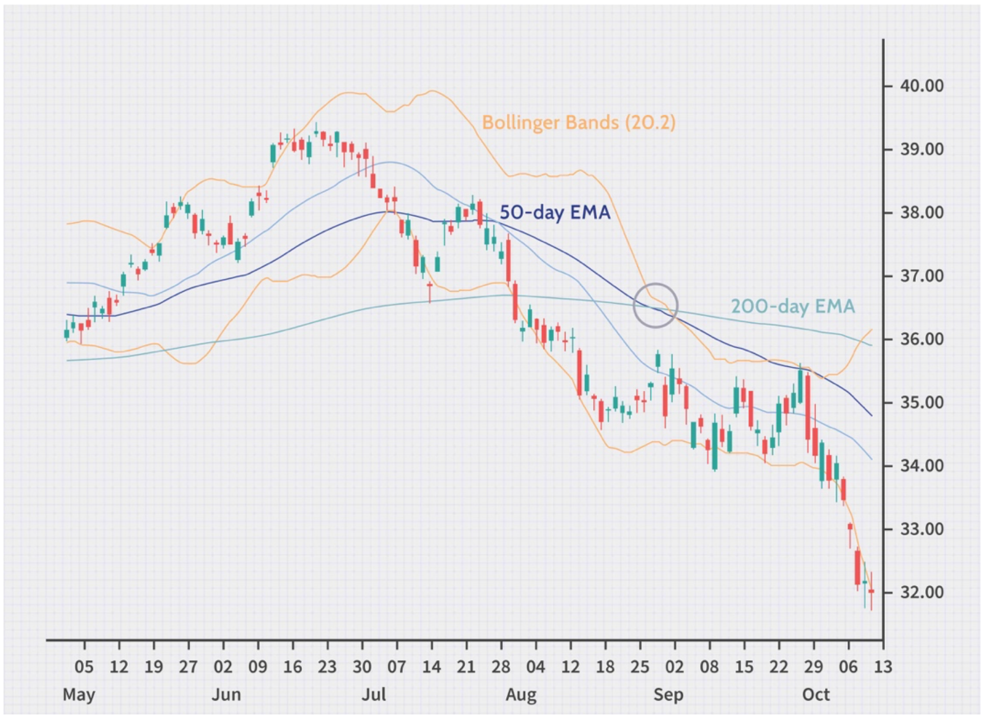

Figure 1. An example of how to use the EMA financial indicator.

During my project, I experimented which various sets of these technical indicators. They primarily helped me reduce the complexity of my neural network algorithms by providing technical indicator ratings as inputs.

Artificial Intelligence Background

As noted before, the core of this project is centered around creating an algorithm for predicting stock prices using a recurrent neural network architecture. Essentially, RNNs are a special type of neural network layer which retains a "memory" to help understand underlying patterns across time series data. This memory container can be represented in various forms depending on which exact RNN layer is chosen.

For my project, I decided to incorporate Long Short-Term Memory (LSTM) network layers. This type of recurrent neural network is more optimized towards remembering inputs over a long period of time. Each LSTM node has three gates: the input gate, forget gate, and output gate. The input gate recognizes which portions of the input should be accepted. The forget gate learns to delete irrelevant pieces of information. Finally, the output gate learns how combine the other portions of the layer affect the final output of the cell. You can visualize all three gates here:

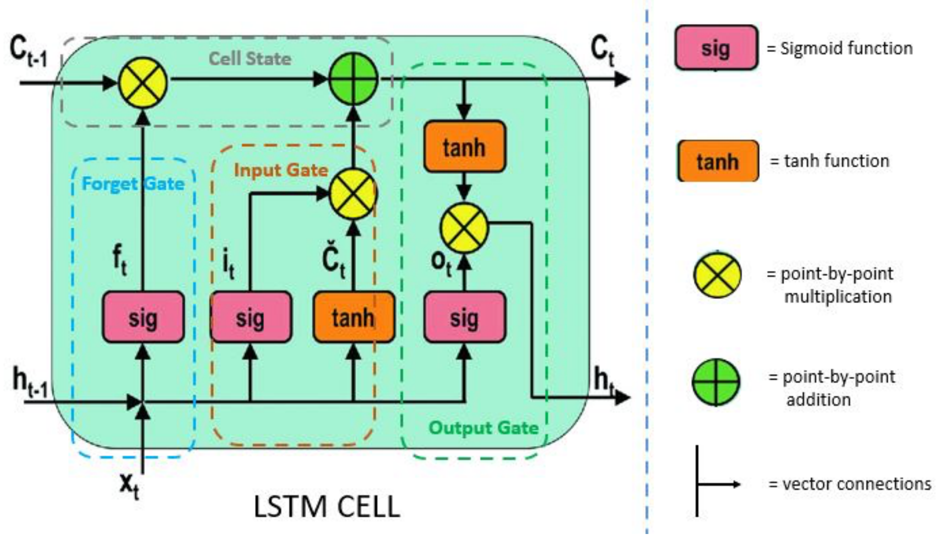

Figure 2. Expanded LSTM cell with all three gates labelled.

A series of LSTM cells are linked together to remember information from long sequences of input. This can be seen in this sample diagram:

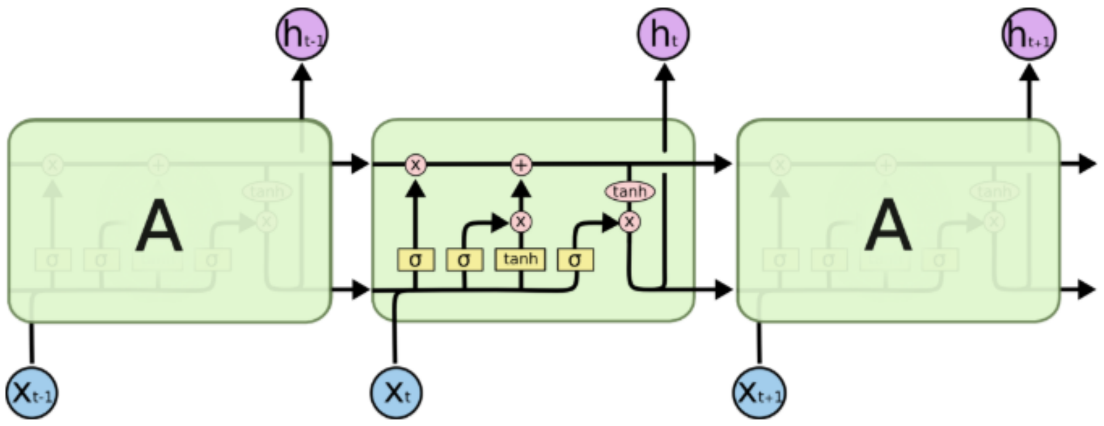

Figure 3. The LSTM layer features a repeating module system.

This makes LSTM layers much more adept at recognizing underlying patterns in the training data.

Methods and Experimental Plan

Algorithm Structure

First, we must clearly define the scope of our problem. When I mention that the algorithm will predict the movement in stock prices, the algorithm will specifically focus on predicting tomorrow's opening price of an asset given prior pricing knowledge after today's closing price is posted.

The most challenging part was designing the neural network algorithm for predicting the stock prices. Initially, I planned on simply feeding the opening, high, low, and close prices along with the volume data (OHLCV) for the past 50 days through a series of LSTM layers. These layers will return a single prediction for the opening price of the stock.

The entire model and associated code was completed in Python using its third-party APIs. For constructing neural network, I used the Keras API alongside the sklearn library for preprocessing data. To access the financial data, I used the popular Yahoo! Finance library in Python. This API provides historical information and some metrics for each stock.

Once I constructed and tested my initial approach, I realized that a major issue of this approach was that it approached the finance problem with pure computer science theory. Upon initial results, it appeared that the algorithm was simply learning to return the previous closing price with random adjustments. Since the values between the current day's closing price and tomorrow's opening price are relatively close, the quantifiable metrics such as mean-squared error still appeared to be doing well despite the ineffective prediction strategy the network used.

To incorporate financial metrics, I created a second input branch to the neural network which only accepted financial indicator values as inputs. This final neural network architecture can be seen below:

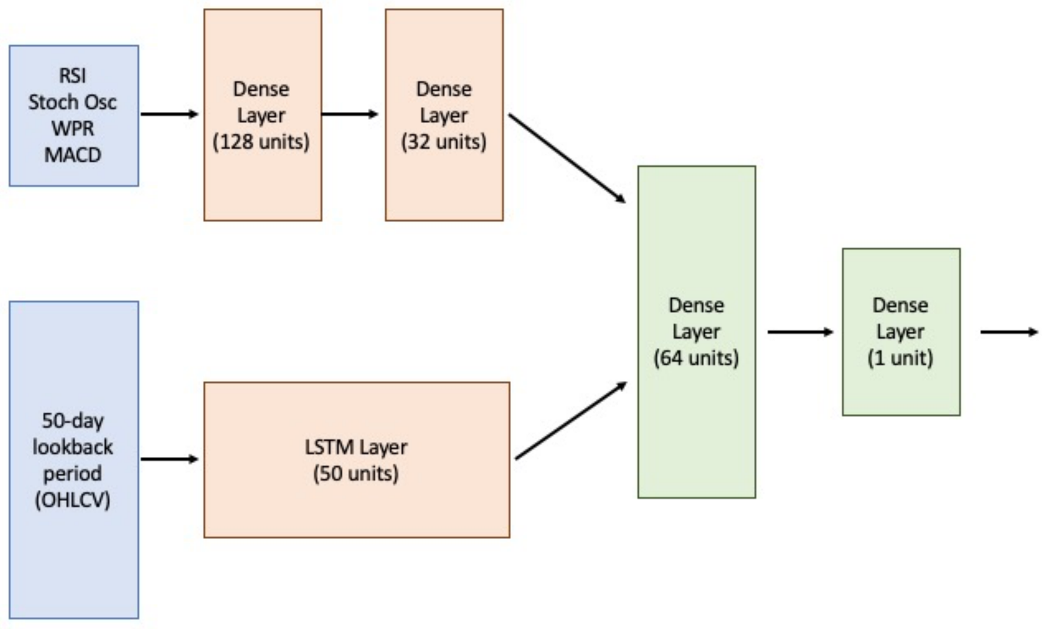

Figure 4. The Final Model Architecture

The lower branch takes the pricing data over the last 50 days as an input and propagates this data through a series of LSTM layers. This is similar to the approach mentioned above. The second branch takes in a vector of financial indicators as inputs for a series of fully connected layers. Finally, both outputs are concatenated and feed through one more layer before extracting the final prediction for the opening price tomorrow.

The theory here was that trend learned from the financial indicator branch will help adjust the prediction from the LSTM branch. This strategy effectively combines both the time-series prowess of LSTM layers and the benefits of tools used by professional traders.

Basic Trading Simulation

To test the capabilities of the predictions in a trading simulation, I employed a fundamental strategy: buy low and sell high. If the prediction is significantly higher the next day, buy 1 unit. If it was significantly lower, sell 1 unit. In addition to buying/ selling 1 unit based on the prediction, if the price increase was significant enough, the simulation would buy as many shares as possible. This aggressive position was meant to rigorously test the reliability of the predictions.

This simulation assumed unlimited bankroll and ability to short-sell.

Results

For evaluating the model, I focused on testing the algorithm on relatively stable stocks with at least 15 years of pricing history. Due to these criteria, I settled testing the model based on the Microsoft (MSFT) stock price since its conception. The model trained on pricing data from 1990 to 2018. From 2018 to 2020, the model attempted to predict the opening price of MSFT every single day and compared it to the historically accurate opening price. This comparison is evaluated via a mean-squared error calculation.

Overall, the results were much more positive than initially expected. After comparing the model prediction line and actual pricing line, the average mean-squared error is 4.327. Here is a sample test run of the algorithm:

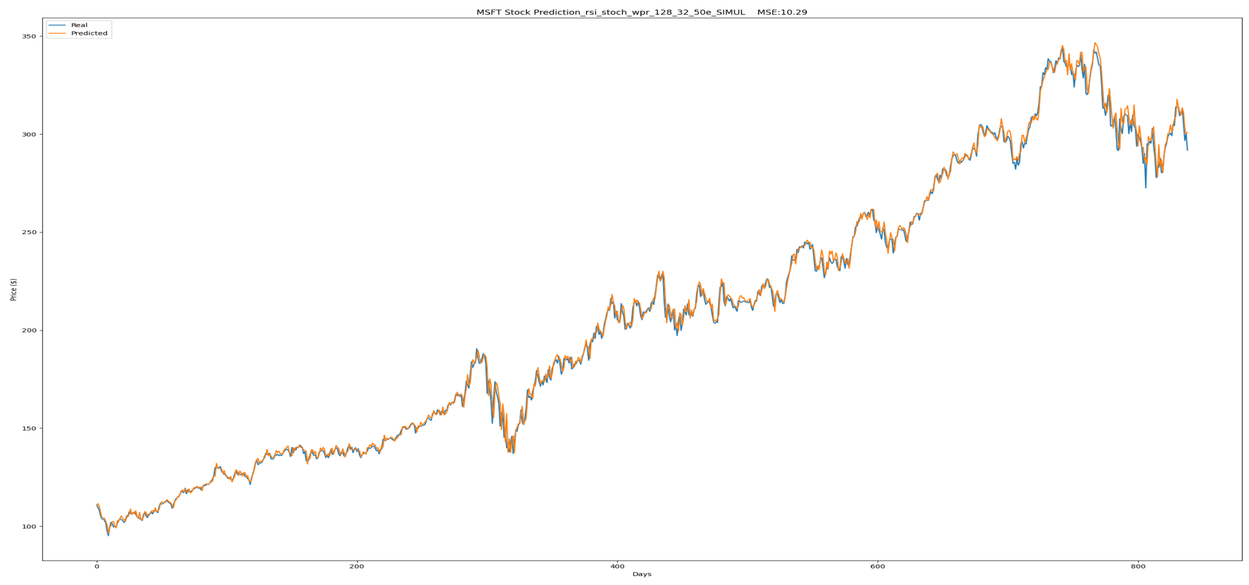

Figure 5. Sample run from final model evaluation trials.

In the figure above, the orange line tracks the price of each prediction while blue is the true value of the stock price. As seen, both lines are highly correlated.

As mentioned in the methods, the model was also evaluated based on its performance when used for a trading simulation. This basic trading simulation tried a simplistic approach and tested the reliability of each prediction. Here is a sample run of the trading simulation:

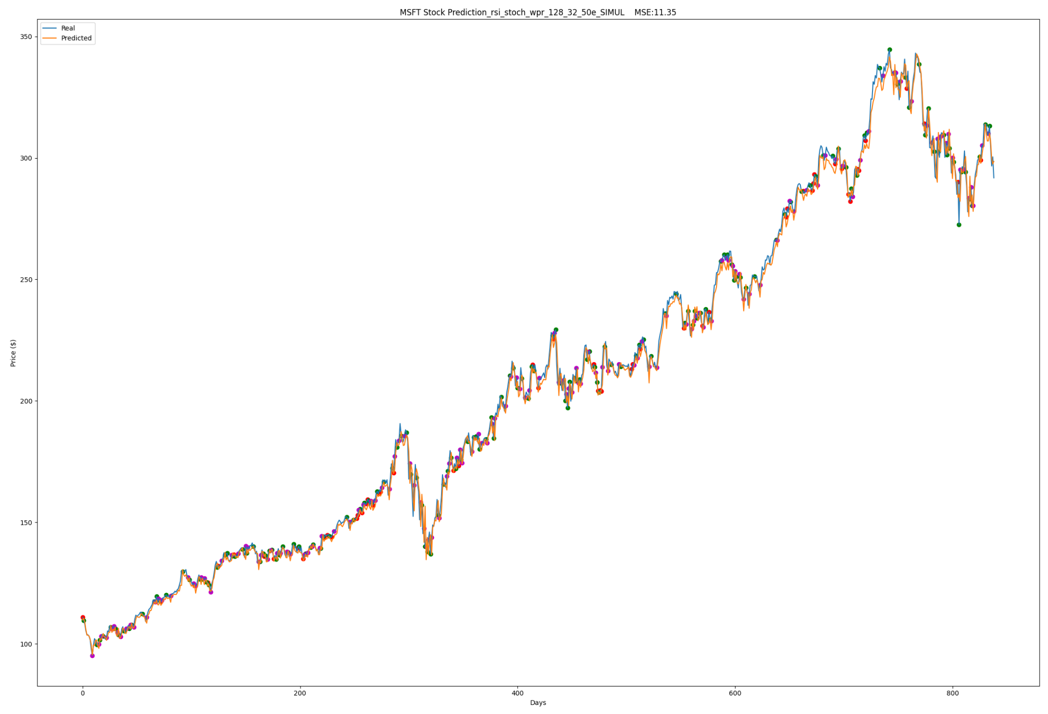

Figure 7. Sample trading simulation graph.

In the figure above, red markers indicate when the model bought while green marks when the model sold. The magenta markers indicate that the model bought assets at max capacity due to a strong prediction.

MSE	*Profit*	
5.58	$7859.6	(as 42 shares for $184.3)
6.04	$8717.10	(as 47 shares for $184.3)
9.12	$8779.4	(as 47 shares for $184.3)
5.03	$7782.82	(as 42 shares for $184.3)
5.26	$8272	(as 44 shares for $184.3)
8.62	$9021.86	(as 48 shares for $184.3)

Table 1. Results from trading simulation

Given an initial of $2500, the algorithm returned an average profit of $8901.02, or 356%, across 2.5 years.

Conclusion

The results of this study show that using an LSTM-based algorithm provides reasonably accurate predictions. However, more research needs to be conducted to expand this algorithm to a real-world trading system.

In future studies, we can explore the use of Bayesian networks to better understand the margin of error and confidence levels behind each prediction. This would help provide more depth to the current binary approach in the simulation. Another potential area of exploration is diversifying the set of indicators and potentially increasing how many are included to provide the algorithm with more information.

Acknowledgments

First and foremost, I would like to thank the Engineering Honors Program for providing me with the opportunity to pursue this research.

I would also like to thank Dr. Romesh Saigal for providing guidance and supporting me throughout this project.

ENGINEERING
HONORS PROGRAM
UNIVERSITY OF MICHIGAN