COMPUTERIZED TRADING WILL BEAT YOU EVERYTIME

Youssef Iraqi
San Diego State University
5500 Campanile Dr.
San Diego, CA 92182
Email: { Youzefih@gmail.com }

Abstract

Everyone who ever traded stocks and stared at a graph changing day after day noticed some patterns continously happening, such as swings when market opens and closes, calm market during lunch time,etc. Taking a closer look, by picking one random stock and watching it day after day noticing patters, you can start to see many trends happening for that particular stock that make you believe there definitely are trends going on and it is more than just random swings. Just like psychologists study the mind and try to understand a person more, predicting behavior and even the future of someone based purely on their emotions, Machine Learning can predict a trend if there is enough data. The importance of predicting a stock value is unquestionable. In 2010, a flash crash happened on the stock market, in a period of 36 minutes, the market dropped and rebounded. After investigations, it has been determined that it was due to computerized trading systems that were triggered by a "Volume-Synchronized Probability of Informed Trading", in other words, big name traders were using computing power to outperform the market, so having such technology available to all the public is crucial for the fairness of the markets. Even a 1% higher probability of prediction would be a huge advantage considering the fact that trillions are traded daily.

1 Introduction

In this project I have used a dataset containing the historical daily prices of over 7000 stocks, with each stock containing around 12 000 daily prices, for a total of over 80 million data points. In the end, I used data from 3 stocks, and by using epoche, each stock used it's own dataset multiple times, with around 120 000 data points for each one.

At first, the goal was to predict the future stock market price based on news, in which basically a news would tell me what percent a stock would go up or down, but unfortunately there isn't enough gathered data to do this, and I would have to look up every single news and relate it to the price one by one to create my own dataset which would probably take years.

The second idea was to predict the stock value based on the daily volume, judging a stock's price change depending on volume. While the data for this was surely available, after further research I found out that there was correlation at rare occasions, in which a stock's volume is extremely lower or extremely higher than it's average volume, but for the most part there wasn't any real correlation that could be used to accurately predict a stock's future value.

The third and last idea, was to simply use daily values from the last 20-30 years, to determine exactly where a stock was headed anywhere from 1 to 30 days ahead.

The main task is predicting with the most accuracy possible if a stock is going up or down based on historical prices.

1.1 Task Description

All tasks will be completed using Jupyter Notebook, a software tool that is really useful in visualizing data. The task process and evaluation is:

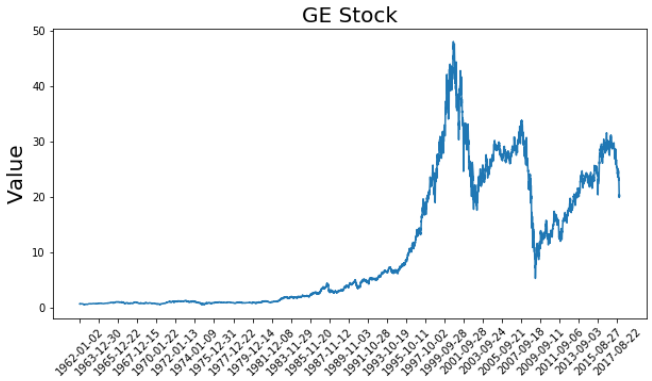
1. Gather the dataset

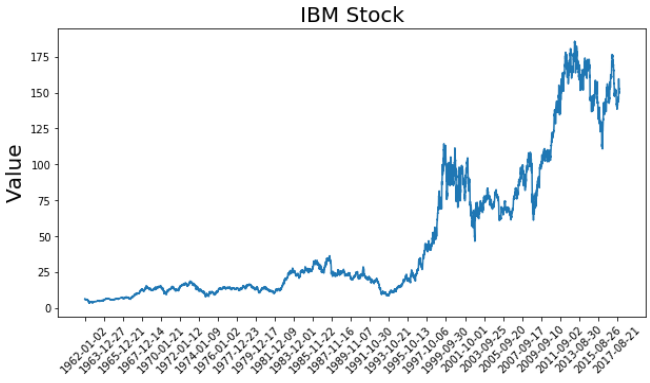
The dataset I used was readily available online. The dataset contained 7000 stock datasets, all separated in different .txt files. The dataset was downloaded from link: (https://www.kaggle.com/borismarjanovic/price-volume-data-for-all-us-stocks-etfs). Each stock originally contained:

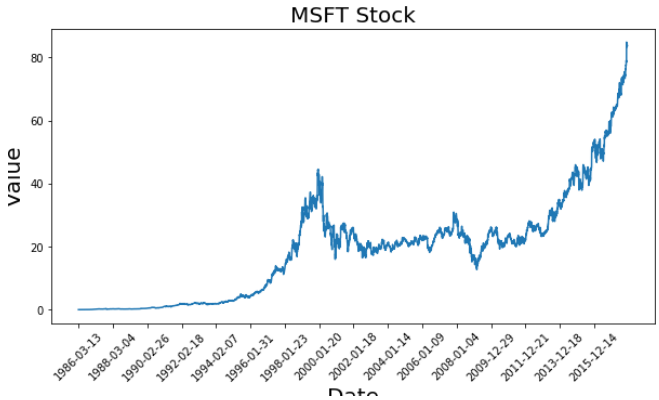
a. Date
b. Open
c. High
d. Low
e. Close

2. Import data and sort it

I choose 3 stocks, m, ibm, and Ge. I imported the stocks one at a time, testing each one separately. To do so, I had to import the dataset from a .txt file, get the dataset in a list, then, I had to choose which data point to use. Since I had the open,high,low and close, I decided to use the average of the high and low of the day as the daily price, as it seemed to be a fair estimate of the stock's price daily. Here is a visual of the datasets:







3. Separate the data

I divided the data into two matrices. Training data and testing data. I put 90% of the data as training data and 10% as testing data. This way, we can test the trained data to see the accuracy of the algorithms we use.

4. Normalize the data

Since the goal of the machine learning project is to figure out if a stock is on an uptrend or downtrend, that is, it is going to go up or down day to day, I normalized my data and adjusted it to be of values -1 to 1. 1 means going up, -1 means going down.

After that, I created a window with ¼ of the data, to perform a window-normalization to add value to the learning process.

The last part of my normalization was smoothing the data to have a smoother curve in my predictions.

5. Predict the future prices

After completing the normalization, the data was good to go for any kind of predictions, so at this point it was all about choosing which models made the most sense to use. The models I used were

a) Forecasting using ARIMA model for standard average and exponential moving average. ARIMA stands for Autoregresion integrated moving average,it is a model that uses the dependent relationship between an observation and a number of other lagged observations using the differenciation between raw observations. It also uses the moving average and residual error from a moving average model.

b) LSTMs prediction. LSTMs stand for long short term memory networks, which are a special kind of RNN. The idea behind LSTMs for this project is to predict far into

the future the prices, so, instead of one day ahead, perhaps 15 days ahead.

1.2 Major challenges and solutions

1 . Major Challenges

Some of the major challenges I faced were:

a) Big data set but small dataset for each stock (only 10000-15000 datapoints for each stock)
b) Picturing what is happening in each step(for debbuging purposes)
c) Runtime for LSTM at 1M datapoints

2. Solutions

a) I have learned that by using apache, I can multiply my dataset, using the same dataset to constantly learn, so I essentially turned my 15000 datapoints into 100 000 datapoints, making my accuracy higher.
b) I was unable to see what was occuring during each step of the process and I wanted to visualize each step to understand how it changed my data, so I included graphs and tables for most steps to have an easier time.
c) The runtime for LSTM was unberable when initially using 1m datapoints, so I proceeded to only use 100 000 datapoints instead, which made the runtime a lot more managable, with still very little mean squared error.

2 Methods and Results

Two models were used for two different cases. Case 1, was to predict the price for 1 day. In case 1 I used ARIMA model. Case 2 was to predict 30 days from today. In case 2 I used LSTMs.

2.1 Data and models *(details about params…)*

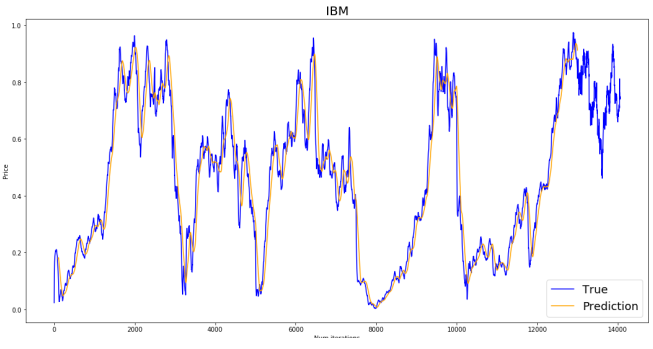
1. ARIMA

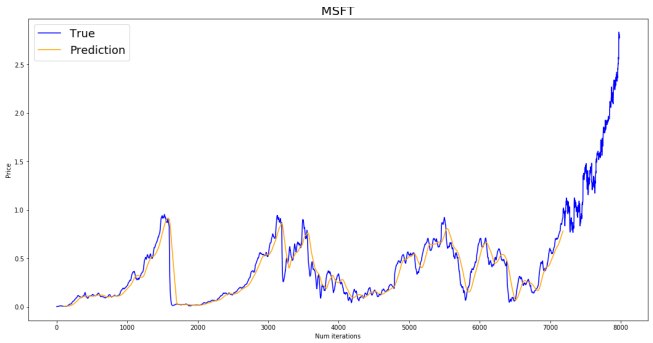
The first model I used was ARIMA, which calculated the standard average, then the exponential moving average for better accuracy. The standard average model formula is:

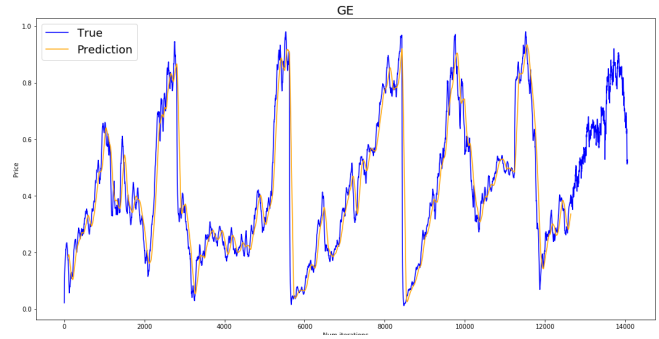
$$x_{t+1} = 1/N \sum_{i=t-N}^{t} x_i$$

The parameters I used were:

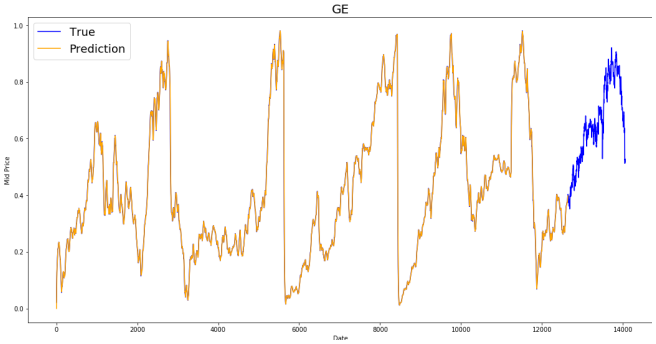
- The window size, which is how many days before you start the predictions. I choose 100, which is is enough for standard averaging since all it predicts is one day ahead.
- Mse errors, which calculated the percent error in each prediction
- Standard average x , which calculated the average price for a given number of days.

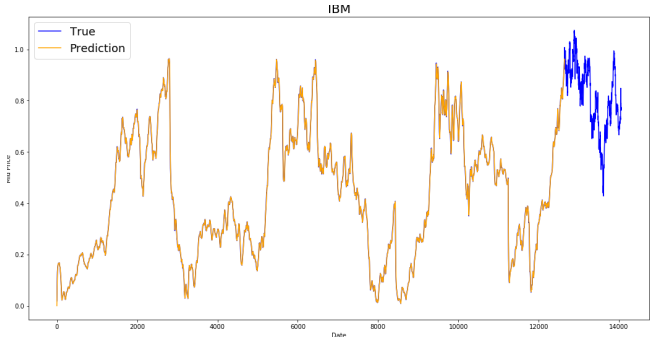


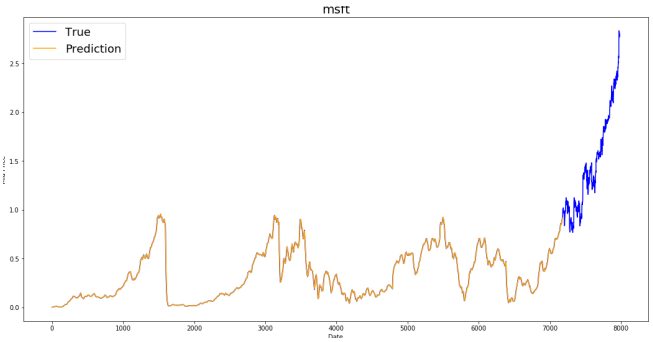




For the exponential moving average, I calculated the EMA, which is the moving average of t+1 and I used the step ahead prediction. This prediction takes a very small fraction of data from the recent data, thus preserving much older values, giving much better predictions







2. LSTMs

The Long Short-Term memory model used five parameters:

- Cell state – The internal memory of the cell that stores short and long term memory
- Hidden state – Outputs the information calculated with respect to current input, previous hidden state and current cell input.
- Input gate- Decide how much information from current input to flow on the cell state.
- Forget gate – Decides how much information from current input and previous cell state flows into the current cell state
- Output gate – Decides how much information from the current cell state flows ito the hidden state.

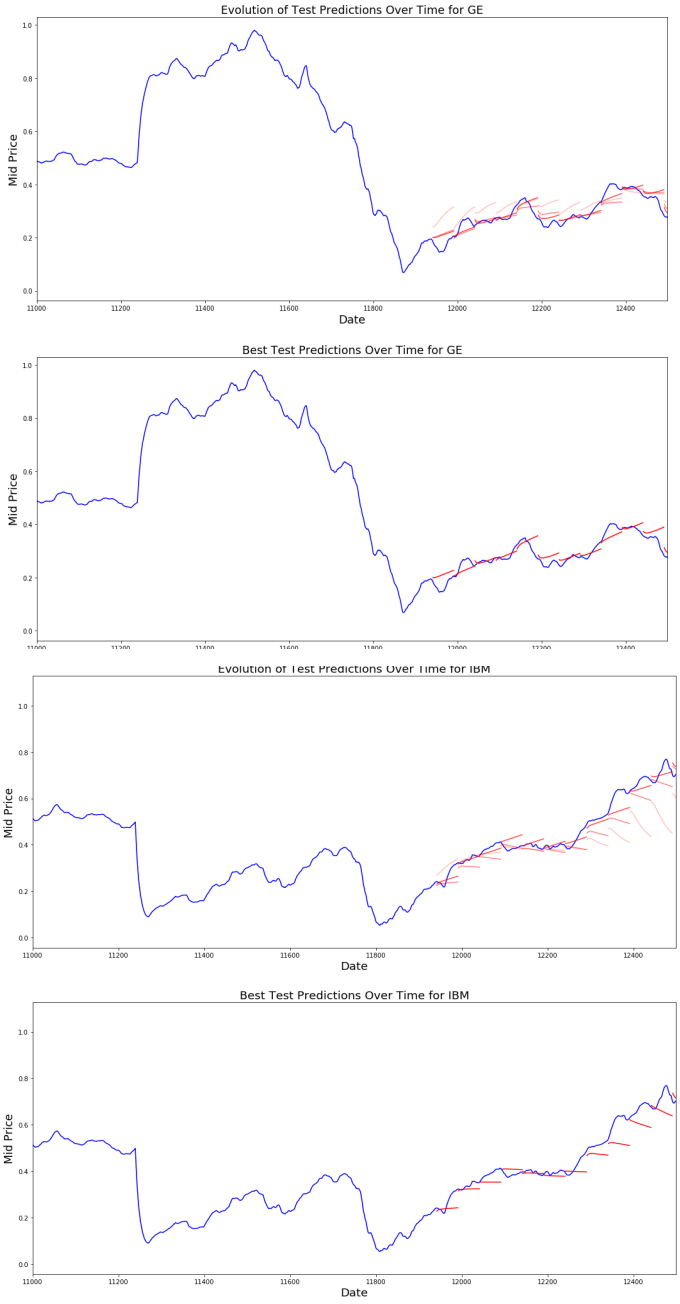
To implement the data to train the model I used TensorFlow API for RNN to implement the time series models.

HyperParameters:

- Dimensionality – 1d
- Size of batch - How many data samples per step
- Number of unrollings – Used to optimize the LSTM model, it denotes how many continous steps to consider for a single optimizatino step.
- Number of nodes – How many hidden nodes in each layer
- Apache – Number of times the LSTM uses itself for mode datapoints, that is uses the same dataset over and over again.

After definiting the parameters, I used multiRNNCell in tensorflow and dropout to improve performance and reduce overfitting.

A visual of the LSTM running on GE and IBM stocks:



These graphs above show how the LSTM learns and there is less and less MSE after each interval. The second graph of each stock shows the lowest MSE compared to the graph.

2.2 Evaluation metrics(compare models/studies)*

1. ARIMA

For the ARIMA model, comparing the standard averaging versus the exponential averaging, there is a clear difference in the results. The standard averaging is pretty good given the simplicity of it, but the exponential model is flawless when it comes to 1 day prediction. The Mean square errror for 1 day ahead predictions is:

Standard: .004 Exponential : .00002

Both the standard and exponential models were completely wrong when used to predict prices 30 days ahead, so they are uncomparable to the LSTM model.

2. LSTMs

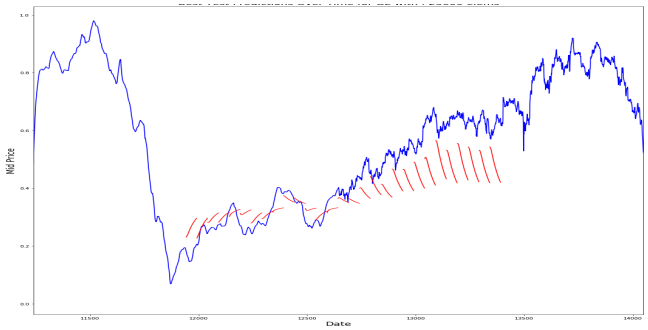
MSE for 120 000 datapoints:

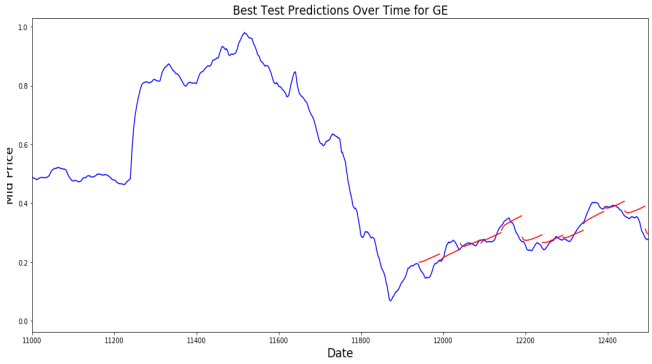
Msft: .014 IBM: .0004 GE: .0003

For 25 000 datapoints, the MSE results were :

Msft: .022 IBM:012 GE: .002

Overall, it is clear that using more datapoints helped predict the prices better. Here are two graphs of GE representing one with 25k and one with 120k:





2.3 Analysis and discussion(why ur model works)*

1. ARIMA for exponential moving average

The ARIMA model fits aperfect line that follows the distribution with very low MSE. This predicts almost with certainty the change of the market from day to day. It works, because there isn't much change from day to day. A drawback of this model, is using it to predict a price a week from today, it would have a big MSE, predicting the same thing for 7 days.

2. LSTMs

THE LSTM model works by taking small batches and saving them each time, calculating the mse and adjusting from there. Unfortunately, for miscrosoft, due to it's high volatility the model wasn't able to accurately predict it's value, giving a very poor mse, but for others, such as GE or IBM and many more, it was predicting the trend with surprisingly very good accuracy.

3 Conclusion and Future works

1. Conclusion

In conclusion the main point of this projects wasn't to crack down the stock market and predict the price exactly but rather to illustrate the fact that it is indeed very much possible, and even so, it is currently happening in hundreds of firms, with far greater computers, engineers, data, algortithms… It is completely impossible to compete with those by "trying to beat the markey" so one should rather purchase long term equity or avoid the market all together, since big firms cleary have an advantage that is growing daily, as technology is improving faster than one can even imagine. Ever since 2006, machine learning changed the way every concept in technology, including the markets, and as always, the big sharks feed themselves first before the small fish, so by the time the public receives a technology helping predict these markets, the next big thing will already be around.

2. Future Works

This project has taught me a tremendous amount of information when it comes to the algorirthms I used and RNN. Future works in this project would most likely require higher computing power, as the complexity of LSTM is fairly slow. I would like to add volumes as a feature, as it seems to affect the markets sometimes, which could increase my accuracy at times. Another thing that I really would

enjoy doing in the near future, would be minute to minute predicting a stock's exact movement. That is, predict the price one minute ahead of time for day trading. Another great concept that I would love to add to the project, given I had access to a big database would be getting the public opinion rating on the stock and adding it as a feature. I know this is very much possible, as facebook and many other companies do that all the time, for example, google knowing your liked vs disliked topics or facebook knowning exactly your political preferences. This data could be gathered from public threads such as "stocktwits" or "reddit", the only drawback that I see, is being able to use this data and add to the dataset live as they are being posted.

References

RSNA Pneumonia Detection Challenge | Kaggle, www.kaggle.com/raoulma/ny-stock-price-prediction-rnn-lstm-gru/data?scriptVersionId=2395222.

Koehrsen, William. "Stock Prediction in Python – Towards Data Science." Towards Data Science, Towards Data Science, 19 Jan. 2018, towardsdatascience.com/stock-prediction-in-python-b66555171a2.

Concoda. "Predicting the Stock Market Is Easier Than You Think." Medium.com, Medium, 8 Oct. 2018, medium.com/s/story/predicting-the-stock-market-is-easier-than-you-might-think-4f1e0bc05cfe.

Yates, Tristan. "4 Ways to Predict Market Performance." Investopedia, Investopedia, 19 Oct. 2018, www.investopedia.com/articles/07/mean_reversion_martingale.asp

"Build a Stock Prediction Algorithm in Python." Build a Stock Prediction Algorithm in Python | Enlight, enlight.nyc/projects/stock-market-prediction/.

Python Programming Tutorials, pythonprogramming.net/forecasting-predicting-machine-learning-tutorial/.

"2010 Flash Crash." Wikipedia, Wikimedia Foundation, 18 Oct. 2018, en.wikipedia.org/wiki/2010_Flash_Crash.

Mitchell, CMT Cory. "How to Use Volume to Improve Your Trading." Investopedia, Investopedia, 20 Nov. 2018, www.investopedia.com/articles/technical/02/010702.asp.

Staff, Motley Fool. "Does Trading Volume Affect Stock Price?" The Motley Fool, The Motley Fool, 20 Mar. 2016, www.fool.com/knowledge-center/does-trading-volume-affect-stock-price.aspx.

Ashutosh Kumar. "Exponentially Weighted Average for Deep Neural Networks." Ashutosh Kumar, 5 Dec. 2017, www.ashukumar27.io/exponentially-weighted-average/.

Time Series Forecasting as Supervised Learning. (2017, May 07). Retrieved from https://machinelearningmastery.com/time-series-forecasting-supervised-learning/

Intro to Machine Learning with Apache Spark and Apache Zeppelin. (n.d.). Retrieved from https://hortonworks.com/tutorial/intro-to-machine-learning-with-apache-spark-and-apache-zeppelin/#what-is-a-model

Understanding LSTM Networks. (n.d.). Retrieved from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The fall of RNN / LSTM – Towards Data Science. Retrieved from https://towardsdatascience.com/the-fall-of-rnn-lstm-2d1594c74ce0

Time series prediction using ARIMA vs LSTM. (n.d.). Retrieved from https://datascience.stackexchange.com/questions/12721/time-series-prediction-using-arima-vs-lstm

How to Create an ARIMA Model for Time Series Forecasting in Python. (2018, August 01). Retrieved from https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/

Financial Market Time Series Prediction with Recurrent Neural Networks. (n.d.). Retrieved from http://cs229.stanford.edu/proj2012/BernalFokPidaparthi-FinancialMarketTimeSeriesPredictionwithRecurrentNeural.pdf