AWS Machine Learning Engineer Nanodegree

Capstone Proposal

Philipp Richter
June 5th, 2022

Prediction of stock returns using LSTM network

Domain Background

Information science and math have long been an integral part of investment management, but recently artificial intelligence in the form of deep learning also became a mainstream part of the investment management toolkit for predicting future asset prices and/or returns. These predictions can be based on several kinds of data, like historical prices and volatility, and news articles using NLP ("Natural Language Processing").

Most neural networks work quite well for image data, but for time series data which is essential in finance, most neural network types are not very good. However, there is one type of neural network which is said to perform exceptionally well at predicting time series data and this is LSTM ("Long Short-Term Memory"), and I am courious to find out if that is true, because I have been working in the finance industry for several years and a stock return prediction tool would certainly be quite useful. The main questions is: Can AI beat the market?

Problem Statement

In order to make an investment profit, I would like to predict tomorrows returns on a selection of S&P500 stocks, based on historical returns of the same stock up until today. In order to take into account short, medium and long-term effects, the prediction will be a weighed average of the predictions based on daily, weekly and monthly returns. In order to make a profit, the predicted return and the real return need to have the same sign. However, I will have to define an "inconclusive zone" around 0 which will be treated as the investment recommendation "do not invest today", i.e. neither assume a long nor a short position in the stock. This is because very small predictions like +0.00451% or -0.00235% are very close to each other, but they would result in opposite investment positions (long vs. short). The range of this zone will probably depend on the volatility of the stock, but I have to analyse this

later, because the size of this range is not part of the problem statement. The success of the model will be measured by the percentage of correct investment recommendations (either “take a long position” or “take a short position” or “stay out of the market”). Therefore, this is a classification problem and not a regression problem, that the ML algorithm has to solve.

Datasets and Inputs

All the input data will be obtained from “Yahoo Finance” (https://finance.yahoo.com), where historical stock market quotes can be downloaded free of charge. A selection of 50 or more tickers will be downloaded as CSV file and stored locally. This is how the raw data from Yahoo Finance looks like for Intel Corp. stock (ticker “INTC”, screenshot taken from https://finance.yahoo.com/quote/INTC/history?p=INTC):

Date	Open	High	Low	Close*	Adj Close**	Volume
Jun 03, 2022	44.11	44.25	43.34	43.39	43.39	33,162,300
Jun 02, 2022	44.19	44.88	43.94	44.84	44.84	28,727,600
Jun 01, 2022	44.77	44.93	43.53	44.11	44.11	29,885,500
May 31, 2022	44.25	44.75	43.65	44.42	44.42	41,111,500
May 27, 2022	43.59	44.55	43.55	44.55	44.55	30,553,300
May 26, 2022	42.17	43.68	42.08	43.48	43.48	28,826,700
May 25, 2022	41.44	42.52	41.39	42.20	42.20	26,283,100
May 24, 2022	41.70	41.89	41.10	41.67	41.67	29,837,500
May 23, 2022	41.69	42.25	41.33	42.00	42.00	27,199,500
May 20, 2022	42.25	42.29	40.31	41.65	41.65	44,780,900

When all raw data is locally stored, it will be pre-processed. The CSV files contain the date, opening quote, highest quote, lowest quote, closing quote, trading volume and adjusted closing price. Only the dates and the adjusted closing prices will be kept for each ticker. It is important to only use the adjusted closing prices, because only those are corrected for dividend payments and any other corporate actions like share splits. Otherwise there would be unexplainable jumps in the stock prices which would confuse the neural network. Then, except for the weekends, all missing prices will be linearly interpolated, to avoid gaps. Then, the prices will be indexed to make it seem like a simple time series from index 0 until index N (which is today’s historical price), to avoid having to fill the weekends. The price at N+1 would be tomorrows price. Then, the simple daily/weekly/monthly returns will be calculated, so there will be 3 time series for each ticker. For each of these 3 time series, a rolling window of X days will be applied, so for each ticker, there will be 3 x (N-1-X) time series. These will be the basis for training the LSTM network. All data will be used for training, while the testing will be done with the actual stock return of the real market on the day after the model training.

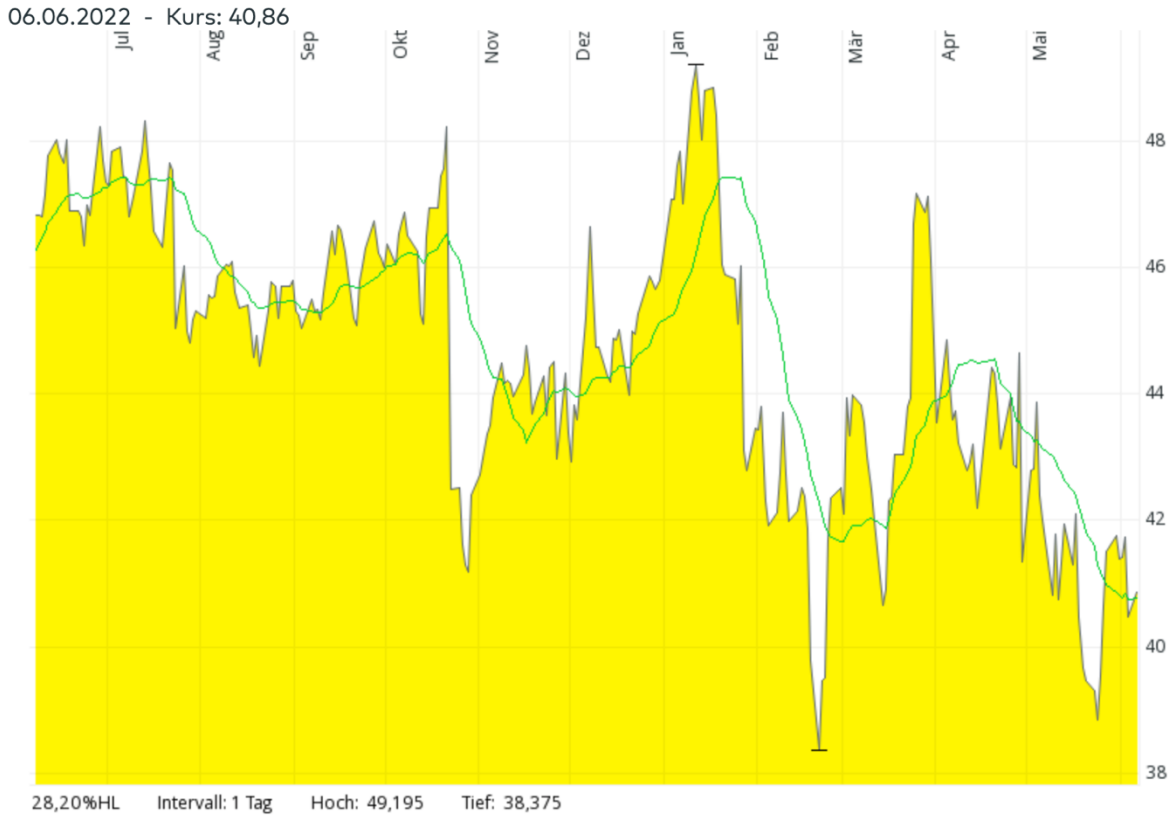
Solution Statement

The solution of the problem is to build 3 parallel LSTM neural networks and train them on the daily, weekly and monthly returns of a selection of stocks chosen by the user. Short, medium and long-term effects are quite different from each other, so I believe that having one common network for all 3 together would not work well. The network would predict a daily return for tomorrow and depending on the size and sign, the prediction would be one of the following (as mentioned above):

- 1. Buy stock (long position)
- 2. Keep cash, i.e. do not invest (neutral position)
- 3. Sell short (short position)

Benchmark Model

As a benchmark for comparing the model, an 18 days simple moving average of the stock prices will be used. The same 3 types of recommendation will be generated and compared to the LSTM model. As an example for the data, the INTC ticker with its 18 days simple moving average of the stock price are displayed in the screenshot below (taken from Comdirect Bank, https://www.comdirect.de/inf/aktien/US4581401001, the green line is the 18 days simple moving average). Note: The benchmark is based on the historical prices, while the LSTM model will be based on the historical returns.



Evaluation Metrics

There are two ways of evaluating the success of the model: 1) The percentage of correct investment recommendations, and 2) the net profit made by applying these recommendations. It might be possible, that the prediction is mostly correct for small returns and incorrect for large returns, which might still look good in terms of the percentage of correct recommendations, but would make the user loose money.

Therefore, the percentage of correct recommendations will be calculated, and then the average return over all tickers will be compared to the invested return of a user which allocates an equal weight to all tickers and follows the investment recommendation of the model. Only if the prediction beats the market, it is a good model. Otherwise it would be better to just hold equal long positions on all the tickers.

Project Design

The following steps will be taken to implement this project:

- Write a command-line tool to download historical stock prices from Yahoo Finance. This tool will take a CSV file with all relevant tickers as input and create one CSV file per ticker in an output folder. It will be executed locally.
- To validate the quality of the data, plot some charts in a Jupyter notebook (also locally).
- Write a command-line tool which pre-processes the raw data like described above in all CSV files of a given folder. All columns except Date and Adjusted Closing Price will be dropped and then all missing non-weekend prices will be filled using linear interpolation. Then the date column will be replaced by a simple integer index to simplify the data and not having to deal at all with calendar dates. At the end, the returns would be calculated on a daily, weekly and monthly basis. The result would be to have an index and a return per line, in 3 CSV files per ticker. The output files would be written in 3 separate output folders. The application of the rolling windows will be done on-the-fly later when training the model. This tool will be executed locally.
- To validate the quality of the data again, plot some charts in a Jupyter notebook (also locally).
- Now that the raw data is available locally in a cleaned and pre-processed form, it will be uploaded to an S3 bucket. Then, a tool will be build to create and train an LSTM neural network (like I have done it in the “Deep Learning Nanodegree”), which will most probably be executed using an EC2 spot instance in AWS (depends on whether or not the credits are enough). Alternatively, it will be done on a local machine.
- Finally, a predictor tool will be build with an estimator which predicts tomorrows returns using an EC2 spot instance, taking the model data from the

S3 bucket. Predictions close to zero will be interpretated as “do not invest” while positive predictions will be interpreted as “buy” and negative predictions will be interpreted as “sell short”. These predictions will be calcuated for all tickers and then the success is measured in terms of 1) Percentage of correct recommendations, and 2) Were the recommendations better than going long by an equal amount on all the tickers, i.e. following the market?

References

- Historical stock prices for Intel Corp. (ticker INTC) from Yahoo Finance: https://finance.yahoo.com/quote/INTC/history?p=INTC
- INTC ticker from Comdirect Bank: https://www.comdirect.de/inf/aktien/US4581401001
- LSTM model basics: Chapter 15.2.7.2 “Long short term memory (LSTM)” (p. 507) (Murphy, Kevin P.. Probabilistic Machine Learning (Adaptive Computation and Machine Learning series) MIT Press.)
- LSTM model details: https://colah.github.io/posts/2015-08-Understanding-LSTMs
- Recurrent layers in PyTorch: https://pytorch.org/docs/stable/nn.html#recurrent-layers
- Implementation and use of LSTM models: “Recurrent Neural Networks” from “Deep Learning Nanodegree” program, Udacity