Stock prediction using LSTM and Hidden Markov Models

Johan Karlsson[*] Johan Lind[*] Zacharias Nordström[*] Rickard Torén[*]

André Willquist[*]

Abstract

The area of stock prediction is a potentially extremely lucrative one to master. In this report two different approaches to stock prediction are explored, LSTMs and HMMs. Prediction models are created with each of the techniques and their performance are evaluated. The evaluations are done by measuring accuracy as well as by simulation. In the simulation the models are compared to three benchmarking methods, buy and hold, buy after trend and a simple ANN. Neither of the constructed models performed consistently better than buy and hold, and none had accuracy much larger than 50%. This indicates that either the models were not complicated enough, the training data was too limited or that the problem is too hard to solve well.

Keywords stock prediction, LSTM, HMM, simulation metric

1. Introduction

The world of stock market trading is a hard challenge to beat, it is also a market that is attractive to master because the large chance of earning money. If the stock market could be easily predicted, it would be trivial to earn a lot of money. There is some evidence according to Chen, T. Leung and Daouk that "stock returns are to some extent predictable" [5]. Historically, stock trading was performed by human investors but as of late, upwards of 70 % of trades in the United States was automated [2].

The purpose of this article is to evaluate the performance of two different models, Long-Short Term Memory (LSTM) models and Hidden Markov Models (HMM). Recurrent Neural Networks and LSTMs in particular have previously been used to handle time series and stock data well [10]. HMMs is a statistical model that also have been applied to stock prediction [11]. The models will be evaluated with two metrics, first the standard classification accuracy and secondly the yearly return when the model can buy and sell the stock in a simple simulated environment. These two metrics have both been used frequently in related work by other authors.

A final problem with accurately evaluating stock market prediction models is that it can be difficult to model the complex stock market. Brokerage fees can play a big part of expected return on investment and this could mean that it is not enough to only predict upwards or downwards shifts.

2. Background

This section presents the dataset used and how the models were evaluated. The baseline models used for comparison are also presented together with a more in depth description of the LSTM and HMM models.

2.1. Data

Yahoo financial data was used through the Python3 module `yahoo-finance`[1]. The six swedish shares ALIV-SDB.ST, AZN.ST, ERIC-B.ST, SAAB-B.ST, SWED-A.ST and VOLV-B.ST was selected, together with ATVI which is an american stock. Two index "stocks" were also selected. ^DJI and ^GSPC are not stocks but comparison indexes that are comprised by different stocks. Index stocks cannot be bought, but they usually represent the stock market well. The data that is exposed through `yahoo-finance` is historical data ranging 20 years back. A datapoint consists of the following fields:

- date - date of the datapoints
- close - last trade price during the day
- open - first trade price of the day
- high - the highest trade price of the day

[*]The authors contributed equally

[1]https://github.com/lukaszbanasiak/yahoo-finance

Figure 1. HMM structure

$$\begin{tikzcd}
Z_1 \arrow[r] \arrow[d] & Z_2 \arrow[r] \arrow[d] & Z_3 \arrow[r] \arrow[d] & Z_4 \arrow[d] \\
X_1 & X_2 & X_3 & X_4
\end{tikzcd}$$

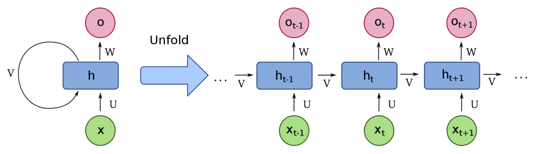

Figure 2. RNN structure[2]

- low - the lowest trade price of the day
- volume - the amount of traded stocks during the day

All fields except for the date was used for prediction.

2.2. Baseline models

The simulation metric is also computed for the following baseline models:

- Buy and Hold (BaH)
- Trend
- Random
- ANN

The BaH model initiates a single buy action at the start and keeps all stocks for the whole simulation period. Trend model only compares todays price to yesterdays price. If the trend is pointing downwards the model initiates a sell action and if the trend is pointing upwards it initiates a buy action. The random baseline model randomly tries to buy and sell at each time step. A simple artificial neural network (ANN) is also used as a baseline model. The ANN has 25 input features, two hidden layers with 15 respectively 7 nodes and outputs 2 values. The network is sequential and fully connected. Tanh is used as activation function for the hidden layers and a softmax is applied to the two output nodes. The two output values if the predicted probabilities for the stock increasing respectively decreasing in value for the net day. Input consists of features such as todays closing price in relation to low, high, open and moving average over different amounts of days.

2.3. Hidden Markov Models

The Hidden Markov Model was introduced in 1966 by Baum and Petrie [4]. When a HMM is used to model a system, the observed data X_t is said to be emitted from some unobservable state Z_t which the system is in, according to a probability distribution $p(X_t|Z_t t)$ called the emission model. The observed data makes it possible to infer which states are more likely at the moment, which in turn gives information about which data points are likely to be emitted next. Time is discretized in equal width intervals $t = 0, 1, ..n$. The states are modeled as a Markov chain, and at each time step the current state is updated according to a transition matrix which is assumed to be constant. Being Markovian, these transitions depend only on the current state, that is $Z^{t+1} \perp Z^{0:t-1} | Z^t$. Similarly, the observation X^t depends only on Z^t. Generally the structure of the model is fixed to what is shown in figure 1

In order to fit this model to data and make predictions, there are a number of problems that must be solved.

1. Given the model parameters, determine the likelihood of the observations.
2. Given the model parameters and the observations, determine the most likely state sequence.
3. Given the observations, determine the model parameters.

The problems are solved by the forward-backward [3], the Viterbi [12] and the Baum-Welch [4] algorithms respectively.

2.4. LSTM

In certain applications it is necessary to remember what you have already seen. Natural language processing (NLP) is such a domain where the next predicted word heavily depends on the previous words. Standard NNs forward the whole input vector at the start which makes it impossible for the network to account for temporal data. To solve this, recurrent neural networks (RNN) were introduced which are networks where nodes are connected temporally.

As can be seen in figure 2, the network has an intrinsic state that is updated based on all previous iterations. The current prediction of what word type is cur-

[2]`https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`, CC BY 4.0

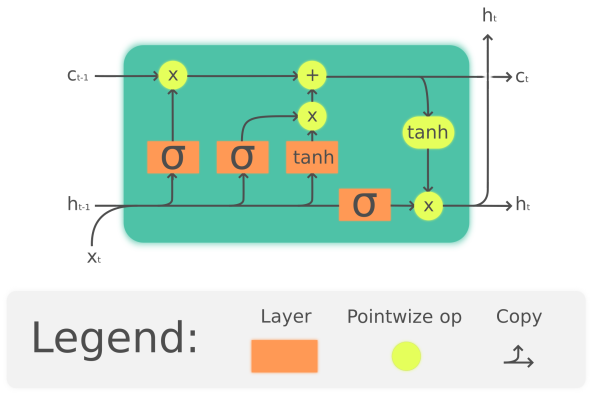

Figure 3. LSTM structure[3]

rently processed, in the context of NLP, is transferred to when we process the following word.

Long short-term memory networks (LSTMs) are an extension of RNNs due to the vanishing gradient problem [8]. By introducing memory gates the network can learn when to forget and remember certain information. In figure 3 there are a few components that needs to be described. Inputs and outputs are illustrated as directed lines while gates are illustrated as boxes. The orange circles are point wise operations. Input, forget and output gates together with the cell and hidden state are the main parts of an LSTM. The top left input is the last cell state c_{t-1}, the bottom left input is the last hidden state h_{t-1}. X_t is the current input to be processed. The top right output is the next cell state c_t and the bottom right is the next hidden state h_t. The equations below describe an LSTM in detail [1].

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hi})$$

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$

$$g_t = tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg})$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$

$$c_t = f_t * c_{t-1} + i_t * g_t$$

$$h_t = o_t * tanh(c_t)$$

The left most gate computes f_t and is the forget-gate which decide what parts of the last cell state to keep for this input. The next gate is the input gate and computes i_t that learns what parts of the input to store for this cell state. Together with the tanh gate, the input gate updates the cell state from c_{t-1} to c_t. The last gate is the output gate that computes h_t and what to predict based on the current cell state.

The benefit of using memory cells is that information can be remembered for multiple iterations, instead of only a few as in RNNs. LSTMs have been used previously in the context of stock prediction with varying results [6].

3. Method

This chapter describes what has been done during the project in such a way as to be replicable. Included in this is how the data is preprocessed, how the ANN, LSTM and HMM is set up as well as how the final tests are set up.

3.1. Data processing

Data that is used in a financial context is often normalized by applying the log-return transform

$$\log p_t - \log p_{t-1}$$

to stabilize the variance [9].

Machine learning tries to predict values or classes based on features. In end-to-end machine learning these features are learned by the system during training. There is however also the possibility of using handcrafted features in a machine learning system. Using hand crafted features means that features are specifically designed outside the system and fed into the system. These are designed in such a way that they are likely to contain useful information. Within stock prediction handcrafted features are often used, in the stock prediction context they are however referred to as technical indicators. In this project technical indicators extracted by TA-Lib[4] were used. The technical indicators which were used are "Simple moving average" and "Relative Strength Index"[5]. When used these were calculated and then added to the rest of the data for the stock.

The models tried to predict whether or not the stock would rise or decrease in value. Each of the data points gathered are then mapped to a label value that reflects this, 1 if the stock would increase in value and -1 if the stock would decrease in value.

The data given to the Hidden Markov Model was preprocessed by for each point in time calculating the 3-D vector $[\frac{Close-Open}{Open}, \frac{High-Open}{Open}, \frac{Open-Low}{Open}]$, as recommended by Nguyen [11]. A difference for the HMM is that the open, close, high, low values can represent values for a longer time period than a day. HMM can take values for a day, week or monthly,

[3]`https://commons.wikimedia.org/wiki/File:The_LSTM_cell.png`, CC BY 4.0

[4]`http://ta-lib.org`

[5]`https://www.investopedia.com/terms/r/rsi.asp`

3.2. LSTM

The LSTM model was implemented using the Pytorch library. To account for the temporal data in the stock market, a moving window approach was selected [6]. A moving window is simply a range where the current day and a predefined length of previous days are selected and predicted on. Below is a list of hyper-parameters that were examined:

- hidden layers
- number of consecutive LSTMs
- window size
- some technical indicators

The number of hidden layers correspond to the size of the internal memory of the LSTM. In Pytorch, one parameter for the LSTM model is the number of consecutive LSTMs where the result of the preceeding LSTMs is fed into the coming LSTM. A large window size means that many previous days are considered for the next day prediction. The technical indicators tested were a subset of the available indicators in TA-Lib. Either this subset was included as features for the network or they were not. Different combinations of these hyper-parameters were tested to find the best performing LSTM model.

3.3. Use of Hidden Markov Models

The emission model $p(X_t|Z_t t)$ is set to be a 3-D multivariate Gaussian distribution, emitting the vector $[\frac{Close-Open}{Open}, \frac{High-Open}{Open}, \frac{Open-Low}{Open}]$ at each point in time. First the number of states is given to the model as a hyperparameter, as this cannot be learned by the Baum-Welch algorithm. All model parameters except number of states are learned using the Baum-Welch algorithm on the previous 100 observed data points. This is repeated, for the number of states $2,3,4,5$, resulting in 4 models with differing complexity. Out of these models, the model with the highest Akaike Information Criterion (AIC) is selected as the final model. Use of the AIC takes into account both the quality of the fit and the number of model parameters, which combats overfitting.

The Forward-Backward algorithm is applied to the 100 data points used to train the model, giving a probability distribution of likely states for the final point in time. We wish to make a prediction for the next point in time, so this probability distribution is multiplied by the transition matrix, giving a probability distribution of likely states for "tomorrow" (or next week / next month). Now the expected value of the next emission is calculated by using the means of the Gaussian distributions of each possible state, weighted by how likely each state is. When predicting for a sequence of time steps, this process is repeated that many times. Once the prediction is made, only the first dimension of the prediction is looked at, as $\frac{Close-Open}{Open}$ represents how much the stock will change in value percentage-wise during the next time step, which is the value of interest.

3.4. Test setup

The data that was gathered was split into three different sets, training, validation and test. To better compare the performance of the different models, the test set data was selected in the range 2014-01-01 to 2018-31-31. For the renaming data, all data points up to 2013-12-31 was split with 80/20% into training and validation The models were trained on the training set, and based on the validation data, the best hyper-parameters were selected. The metrics that were presented in section 3.5 were used to select the best performing model.

Only one models was selected for testing, this models was the one with overall best performance on all trained stocks.

The stock was retrained for each training on the training data for that stock. When the stock was run on the test data it was still only trained on the training data and not the validation data.

3.5. Performance evaluation

To compare and evaluate the performance of the models in this article the following metrics were used:

- Accuracy
- Annual return from simulations

The accuracy is computed as $\dfrac{TP+TN}{TP+FP+TN+FN}$ and is a good measurement for classification datasets that are balanced. A negative aspect of accuracy is that a high accuracy might not indicate how good the model would perform on the stock market. A simulation metric was then introduced to better reflect the portfolio value the bot would end up with after the stock prediction period was over.

The simulation starts with a set amount of money, in this case 100000 SEK for the swedish stocks and 100000 USD for the other. At each time step T, the models predicts whether or not the stock will go up or go down. An up prediction while not owning any stocks results in an instantly buying as many stocks as possible for the price of the stock at time step T. Similarly, a

Stock \ Model	ANN	LSTM	HMM	BaH	Trend	Random
ALIV-SDB.ST	-10.8	1.3	-4.9	5.5	2.4	0.9
ATVI	10.0	-0.4	7.0	19.1	4.7	-5.6
AZN.ST	6.3	-3.5	3.3	11.5	7.1	5.0
ERIC-B.ST	-10.7	5.4	5.9	-0.2	1.3	-6.0
SAAB-B.ST	4.9	12.9	3.9	16.0	10.5	7.7
SWED-A.ST	-2.4	-2.8	3.5	2.5	3.0	-0.3
VOLV-B.ST	1.6	8.0	11.6	4.0	-7.9	0.4
Average	-0.16	2.99	4.33	8.34	3.01	0.3
ˆDJI	-0.3	3.3	-0.3	8.6	3.7	2.0
ˆGSPC	6.0	6.9	-2.6	7.5	2.0	6.6
Average	2.85	5.1	-1.45	8.05	2.85	4.3
Average over all	0.51	3.46	3.04	8.28	2.98	1.19

Table 1. Annual returns (in percentage) for the different stocks

down prediction wile owning stocks results in instantly selling all stocks own. All simulations use the closing price except for HMM that use the open price for trades. The decision at time step T is decided based on the previously seen data. Even though the simulator was developed to be able to handle transaction fees, these were excluded. This was done since the goal of the report is to investigate the possibility to predict the direction of a stock. Whether these predictions would make money or not in a real trading environment was regarded as secondary and thus not as explored. The simulation does not take into account spread, since we do not have data to model it.

The final portfolio value can then be used to, for example, calculate an annual return rate to represent the performance of the model. The annual return of a stock is calculated as:

$$\left(\left(\frac{portfolio\_value_{end}}{portfolio\_value_{start}}\right)^{\frac{\#trading\_days\_a\_year}{\#trading\_days}} - 1\right) * 100$$

It represents the trend of a stock over a longer period with a single value.

4. Result

The general performance of the implemented models was rather poor. In this section, first the overall results will be presented together with a description of the performance on the ERIC-B.ST.

4.1. Overall returns

In table 1 the predicted annual returns for the combination of stocks and models are presented. Recall that the BaH model mirrors the stock value. From the table we can see that the trend model generally performs worse than the BaH model with only beating BaH on the SWED-A.ST stock. Randomly predicting seems to be bad as the random model also has a worse result than BaH. The implemented models had a hard time with results that are very varied. For example, the LSTM model predicts a rather large negative annual return for AZN.ST while the BaH annual was 11.5%. The same applies for the HMM model on the ALIV-SDB.ST stock. Good predictions can be seen on the VOLV-B.ST stock for both the LSTM and HMM model. In general, the results are rather poor compared to BaH. The HMM model has a higher average performance on the normal stocks over both the LSTM and the ANN model.

The index stocks are apparently hard to predict with the HMM model while the LSTM does a reasonably good job. The average annual return for the BaH model over the normal stocks is very close to the annual return other over index stocks. The index stocks are a collection of stocks that usually capture the general market trend. Given that the average of the examined stocks is close to the annual return of the index stocks, the examined stocks could be representative of the stocks market as a whole.

The accuracy for all stocks and for the implemented models were around 50%. The highest recorded accuracy was only 53% for one stock and model but if compared to the positive rate of that stock, it was only ~ 3 percentage points larger. Something to note is that if the accuracy would be very large, the annual returns would be extremely large. The gains from the stock market is exponential as previous gains would be invested again.

4.2. ERIC-B.ST results

In figure 4 the value over time of the ERIC-B.ST stock is presented given the predictions of four different models, LSTM, Trend, BaH and Random. The positive rate (percentage of ups in the stock), accuracy of implemented model and confusion matrix is presented. The LSTM model performs poorly on the first half of the period, after the end of 2016, the blue line is consistently above the green line and performs better than BaH. It is difficult to analyse the performance in greater detail, we can for example see that the accuracy and confusion matrix indicate that the predictions seem randomly distributed. Something that was apparent in almost all stocks was that the accuracy was higher than the positive rate. This is something that is expected at least to some degree, the trend of a stock does not shift up and down every other day, there is usually some period where the trend stays the same. The difficulties then become when the stock shifts and that is apparently difficult.

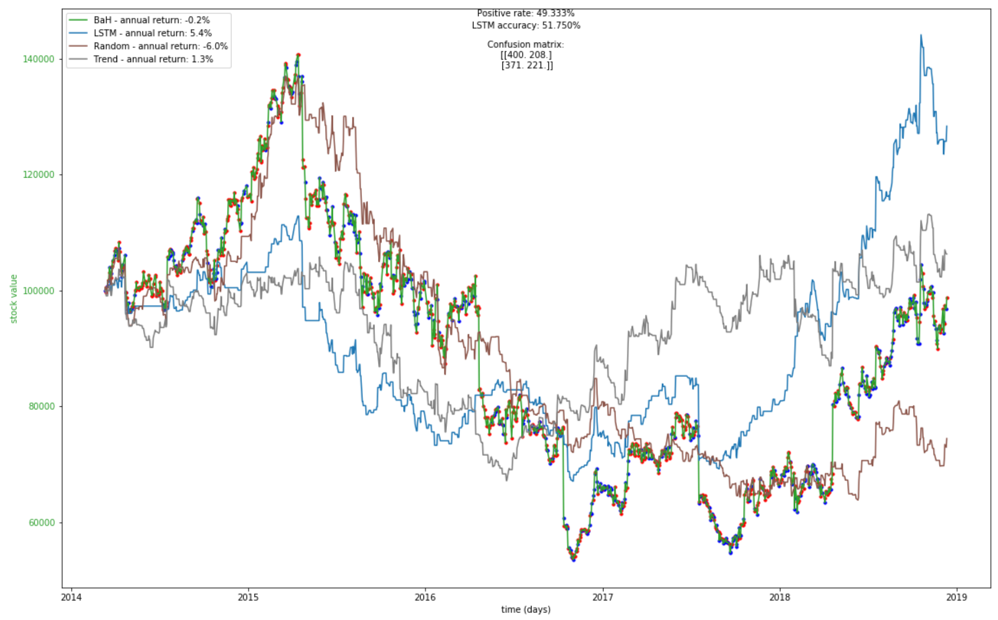

Figure 4. LSTM predictions on ERIC-B.ST

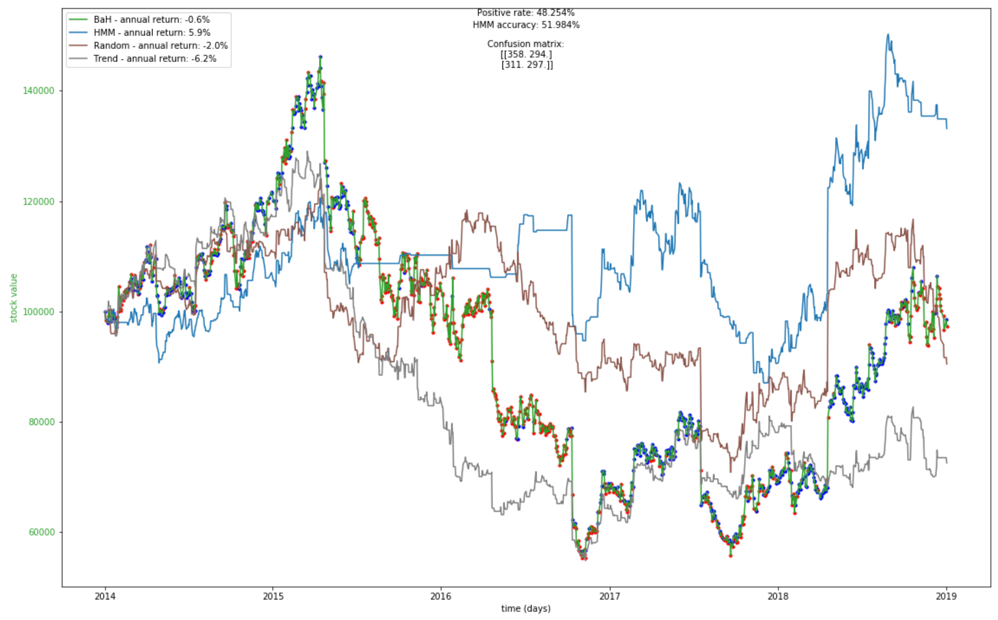

Figure 5. HMM predictions on ERIC-B.ST

Figure 5 is the HMM performance on the ERIC-B.ST stock. Please note the similarity between the BaH models of the two plots, there is although a small difference between the x-axises where the LSTM model has an earlier stop date. It is not a big difference but must be noted. The HMM model also performs well on the ERIC-B.ST stock. Both of the models seem to perform exceptionally well during the last month but the HMM also seem to perform a bit better in the middle of the test.

4.3. Framework

To be able to predict the stock market a framework for doing so was created. This framework reads data from yahoo finance and pre-preocess the data into a format that can be predicted on with LSTM:s and HMM:s. The framework can split the data into training, validation and test data; and the model can be trained then validated and lastly tested on the data sets. Easy tuning of the parameters for a network exists as it is only to expand a base class. The framework utilizes docker to handle installation of packages needed and is then run with jupyter notebook for easy training and tuning of the network.

5. Discussion

In this section the result, the performance metrics and some potential reasons for the poor performance is discussed. The accuracy was around 50% for all models and for all stocks. A different approach for when to predict is also presented in this section.

5.1. Good performance metrics

The two metrics used in this report were not without fault. As noted in section 3.5, there are some apparent limitations of the accuracy metric. In general, with a high accuracy there will be more correct classifications. However, the stock market is made up of real values and thus could also be modeled as a regression problem. This regression aspect is not reflected in the accuracy, meaning that the accuracy does not reflect if the change was a large or a small one, only if the predicted direction was correct. As a result, a good accuracy might not reflect a good overall monetary performance. This since the correct guesses might be on small changes while the misses might be on large changes The accuracy might also be lowered due to a large amount of incorrectly classified small changes while the large changes are correctly classified. It is however the case that accuracy is a widely used as a metric for machine learning with classification problems, so it was still included in the report.

The annual return based on the simulation also has some problems. The simulation is much simpler than the real market and thus not accurately represent how our models would perform in a real world scenario. A real market is very oscillating, only open some hours a day, has fees and requires both a buyer and seller willing to trade for a price for the trade to be completed. Our simulation only allows the bot to trade using one price a day, without any of the complications mentioned. Another reason why a simulated annual return can be a misleading measurement for our models is because our models were not trained to make successful trades on the stock market but to predict the stock market. Trading on and predicting the stock market is closely related but not the same thing.

If a stock have an upwards trend it can be reasoned that it is hard to beat investing in the stock and waiting. This due to the stock not dropping in price which means that there are fewer chances to sell stocks and then buy them back cheaper. Or if it is attempted, then the stock might increase and the shares have to be bought back at a higher cost, which will mean a loss instead of a gain.

5.2. Accuracy

The accuracy was rather poor (around 50%) for all the different models and stocks. Figure 6 illustrates accuracy in relation to change in the stock. The plot shows the accuracy for the predictions only on the days when the absolute value of the daily change is larger than a fixed percent. Note also that the daily change referenced in the plot is the true change in the stock and is not a predicted value. The plot indicates that it is more difficult to correctly predict small changes. Considering a threshold $< 2\%$ the model achieves around 50% accuracy, but with a threshold of $\sim 3\%$ the model achieves a higher accuracy of around 55% to 60%. With larger thresholds the accuracy fluctuates to a much higher degree due to fewer such changes in the data. Wrong predictions where the change is small does not affect the simulation result much but accuracy is affected. The accuracy could maybe be better modeled if a neutral "hold" action was introduced where the small changes in the stock value is not considered. It would be interesting to see if such an approach could help with determining what model performs the best.

This report has previously only considered the predictions from the models as binary decision, i.e. if the stock value will increase or decrease, All the models still gives some form of measurement of how confident the prediction is. For example the LSTM model gives a

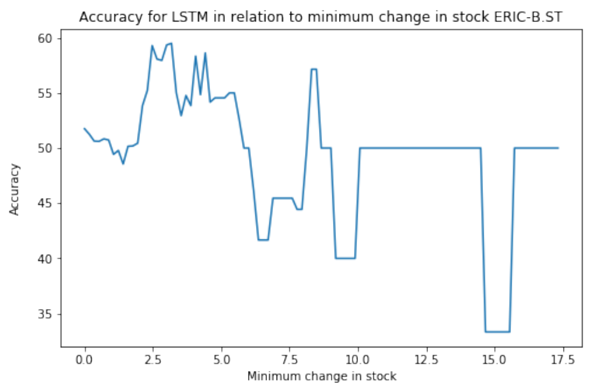

Figure 6. Accuracy for LSTM in relation to minimum change in stock ERIC-B.ST

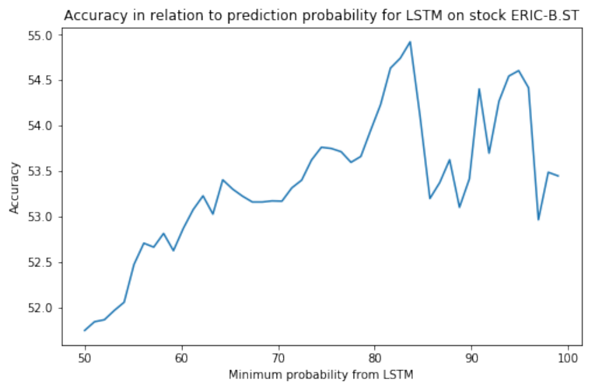

Figure 7. Accuracy in relation to prediction probability for LSTM on stock ERIC-B.ST

probability for each of the two classes. Figure 7 shows how the accuracy change when only considering predictions where the highest probability is above a fixed percentage. The plot shows that predictions with a higher probability generally has a higher accuracy.

Figure 8 shows the distribution of the prediction probabilities from figure 7. The model is often certain of its decision. In over half of the predictions, the probabilities are above 80%. A drawback of predicting when the model is more certain is that fewer trades are made, it would be interesting to see if this also affects annual returns. Based on figure 7 and figure 8 it would seem reasonable to assume that taking the probabilites into account would lead to a better performing model.

5.3. Potential reasons for poor performance

The overall performance of all of the models was in general rather poor, both in terms of accuracy and simulation results. The accuracy was mostly at or slightly

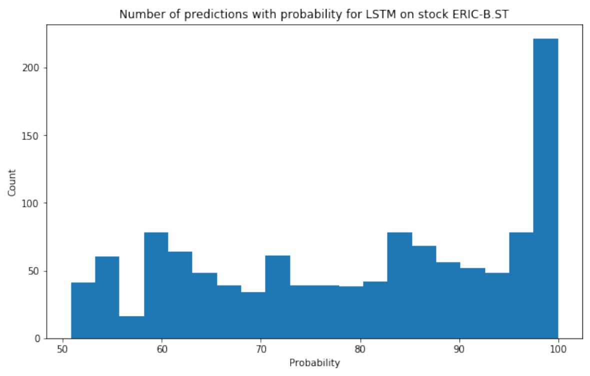

Figure 8. Histogram of probabilites for LSTM on stock ERIC-B.ST

above 50% which is just about the same as only guessing that the stock value will rise. As for the simulation results, BaH was more often than not the best model. There are several reasons as to why this might be the case.

One reason could be that the stock market fluctuations is based on random walk, as supported by some previous work [7]. This means that while the value of the new day is dependent on the previous day, whether the values increase or decreases is, essentially, random. If the stock market is indeed a random walk then this would entail that it would not be possible for the models to predict any better. The implications of the market being a random walk would be, as stated in the article, that the optimal policy would be the BaH policy. The inability to outperform BaH is then expected.

Even if the stock market is not a random walk then there are still several other potential reasons for the poor performance of the models. One of these could be due to a lack of data points used for training the models. As stated in subsection 2.1 the data used was gathered from `yahoo-finance` that records 20 years of data. It is however daily data, meaning that 20 years does not result in as many data points as one might expect. The stock markets are in general closed on weekends and other major holidays. Only assuming that it is closed for weekends this would mean that the total amount of data points for 20 years would be $52 * 5 * 20 = 5200$ data points. This then has to be split into the training, test and validation sets. The results of this is the training set being made up of about 3500 data points for most stocks. 3500 data points is a rather tiny data set compared to many other areas where machine learning is used, especially if this is considered to be a rather difficult problem. This might result in the models not having enough data to learn properly, leading to poor performance.

A third potential reason is that the final model is poorly chosen. A lot of different hyper-parameters need to be tuned in order to create a well performing final model. There were attempts to make sure that a large variety of parameters were considered. This includes the creation of several different models which were each tested on the validation data, for a final best model to then be selected. The different models tested might not have been enough however. There might have been different combinations of parameters with (much) better performance. The lack of data also affects this problem. This since larger models with more hidden nodes or LSTM layers require more training in order to learn. The more complex models might not have enough data to learn all related weights properly, leading to them underperforming. This might mean that potentially better performing models might have been discarded due to a lack of data.

6. Conclusions

As discussed in 5 predicting the stock market is a hard problem and its secrets are elusive. This project have created a framework that was used to evaluate some models if they could learn the fundamentals of the stock market and help with investments. The models are able to give advice if it is good to invest in a particular stock, but the advice is not worth listening on. The models implemented achieve around 50% accuracy and very varying performance on the simulation metric. The LSTM and HMM model achieves a positive annual return on average but it is often lower than the BaH annual return. There are some indications that a different approach on how to handle small changes in stock value could help in the final model selection. A threshold on when to predict up or down could help the accuracy. Some potential reasons for the poor performance are presented, the stock market might not be predictable, the data set uesd could be limited and the final model selection process could be improved. As a final note, if there is someone interested in investing in stocks, the advice is to buy the stock and then hold it. The BaH model performed the best on average as stocks in general increase in value.

7. Future work

In section 5.3 three potential reasons for the poor performance were discussed. This section presents possible approaches on how to handle those together with some extensions that could be made to better answer the question: “Can you predict the stock market?”.

To increase the amount of training data, one could potentially combine multiple stocks into one data set. Perhaps there are similarities between stocks that the LSTM and HMM models could make use of. The combined data could perhaps be worse for predicting a particular stock but similarities could be drawn to the many applications of image prediction where models are pre-trained using large datasets and finally trained on a particular data set.

The selection of the final model was difficult due to the random nature of the accuracy and simulation metric. If the models gave rise to more stable metrics, this selection would be easier. A more structured hyper-parameter selection process could also be useful. More hyper-parameters could also be investigated, for example, different optimizers or different loss functions could be examined.

As for the potential reason that the stock market is unpredictable, there is not much to be done. More reasearch is definitely needed to answer this. This report only focused on daily-trading, perhaps other trading methods are more (or less) predictable. Also, other models could perhaps be examined as this report only focused on LSTM and HMM.

Other areas that could be looked at in future work is to extend the framework created for different machine learning methods. These might give better performance then the ones tested in this report.

References

[1] Pytorch nn documentation. `https://pytorch.org/docs/stable/nn.html`. Accessed: 2019-10-14.

[2] J. Arnoldi. Computer algorithms, market manipulation and the institutionalization of high frequency trading. *Theory, Culture & Society*, 33(1):29–52, 2016.

[3] L. E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3(1):1–8, 1972.

[4] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[5] A.-S. Chen, M. T. Leung, and H. Daouk. Application of neural networks to an emerging financial market: forecasting and trading the taiwan stock index. *Computers & Operations Research*, 30(6):901 – 923, 2003. Operation Research in Emerging Economics.

[6] K. Chen, Y. Zhou, and F. Dai. A lstm-based method for stock returns prediction: A case study of china stock market. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2823–2824, Oct 2015.

[7] E. F. Fama. Random walks in stock market prices. *Financial Analysts Journal*, 51(1):75–80, 1995.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[9] H. Lütkepohl and F. Xu. The role of the log transformation in forecasting economic variables. *Empirical Economics*, 42(3):619–638, Jun 2012.

[10] D. M. Q. Nelson, A. C. M. Pereira, and R. A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 1419–1426, May 2017.

[11] N. Nguyen. Hidden markov model for stock trading. *International Journal of Financial Studies*, 6(2):36, 2018.

[12] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269, 1967.