NTNU
Kunnskap for en bedre verden

Department of Computer Science

TDT4173 - Assignment 3

Predicting short-term stock prices using LSTM neural networks

Group:
Group 9

Authors:
Espen Løvhaugen (Espenblo)
Haakon Flaarønning (Haakotf)
Kristian Bondevik Hardang (Krisbha)

September 20, 2022

Abstract

Predicting stock prices is a very important topic in the financial world. Reasonably accurate predictions have the possibility of yielding high financial benefits, and even small improvements in predictive performance can deem very profitable. The overarching goal of this paper is to develop a machine learning model which can predict short-term stock prices accurately enough to be somewhat informative for investors. The stock prices in this paper are predicted using long short-term memory (LSTM) neural networks. Through our research, we have aimed to gain knowledge and insights on how LSTM networks can be used to predict stock prices. To start off, a brief introduction to stock price prediction and related work is presented followed by a description of the data and preprocessing steps used in our experiments. Then, our models are described in detail, and applied by predicting on 3 different stocks, namely BP, Intel and Ford. Different variations of models are tested to investigate possible advantages and disadvantages of different network architectures, objective functions, hyperparameters and more. Additionally, shallow versus deep networks and training on one compared to multiple stocks are discussed. From our results we conclude that our models do not provide sufficiently accurate predictions to be highly informative for investors, however, they can be used as an indicator for short term volatility. Therefore, we have laid out some suggestions of future work which can be made in the field of stock price prediction with LSTM networks to further progress the field of research.

Our research paper has an associated website! Check it out here: https://khardang.github.io/lstm-stock-prediction

And the source code can you find in this repository: https://github.com/Espenblo/TDT4173-Project

Table of Contents

List of Figures

List of Tables

1 Introduction

The stock market is an ever growing market with millions of transactions occurring every day. The ability to predict stock prices accurately can give anyone a significant advantage in the market. Stock price prediction is therefore highly interesting for those wanting to make good money by beating the market. However, predicting the stock market has been a controversial topic for a long time. If the market is efficient, meaning that all information is reflected in the price, there will be no way to "beat" the market [Dhir, 2019]. However, the stock market is not perfectly efficient and new information can become available. [Cootner, 1964] agreed with this notion, and suggested that stock prices cannot be predicted since they are driven by new information which cannot be captured by analyzing historical data. Recent studies, on the other hand, has shown that stock prices can to some extent be predicted ([Balings, 2015], [Bollen, 2011]). In this paper we further explore that assumption by measuring the prediction accuracy of LSTM neural networks when predicting short-term stock prices. Our reason for choosing LSTM networks is due to its ability to capture long-term dependencies and perform cross-learning, both of which is crucially important when working with time series data. Our LSTM models are trained through supervised learning, and is fed with daily price and volume data from one or multiple stocks. Given data from the past 60 days, it attempts to predict the closing price for the next 1 to 5 days into the future.

2 Related Work

The stock prediction literature can be categorized into four primary metrics:

1. The type of prediction model used.
2. The prediction goal: It can either be a prediction of the stock price movement at a given time, or a continuous target like the stock price or return on investment.
3. The features included in the model (data extracted from the market, technical indicators, news sentiment analysis etc.).
4. The length of the prediction period.

In the research of [Bin et al., 2018], the researches used ensemble methods to predict stock prices of big companies 1 day into the future. During training, they fed their model with historical stock data, common technical indicators, sentiment scores for news related to the company, trends in google searches for the stock ticker and the number of unique visitors for pertinent Wikipedia pages. They created four different ensemble models, including a neural network regression ensemble, and measured the prediction accuracy for each model. On average, their models predicted the 1-day ahead stock price with a mean average percentage error (MAPE) $\leq 1.5\%$.

[Balings, 2015] did similar research when comparing the prediction accuracy of ensemble methods versus simple classifiers for long-term stock price prediction. Their models attempted to predict the stock price one year ahead by using historical data (price, volume, price to earnings) as well as several well-known technical indicators such as return on equity and return on assets. Their research found that models such as random forest, support vector machines and adaboost all performed better than neural networks.

Our research differ from these papers through our use of different variations of LSTM neural networks. However, we chose to use the same evaluation metrics as [Bin et al., 2018], to get comparable results for our short-term prediction.

3 Data

This section describes the data that has been used in the project, and how it was pre-processed before it was fed to the model.

3.1 Source

The datasets are fetched from Yahoo Finance. To decide what stock information to fetch, we simply provide the webreader with a stock ticker and the desired start and end date. The dataset contains records for all dates when the market were open, from the selected start date to the selected end date. Table 1 shows an example of the data from the dataset.

Date	High	Low	Open	Close	Volume
2000-12-08	4.479167	4.395833	4.437500	4.479167	234900.0

Table 1: Shows how the data are structured, values in USD($) except volume which are number of shares traded

The following list is a description of the features we chose to include from the dataset:

- **High** - The highest price of the day
- **Low** - The lowest price of the day
- **Open** - The price when the stock market opened
- **Close** - The price when the stock market closed
- **Volume** - The amount of shares traded that given day

3.2 Data characteristic

In our experiments, we chose to predict on three stocks from three different sectors, namely the automobile, petroleum and technology sectors. The datasets for the different sectors has different start dates, hence the difference in number of records in the training and test sets. Table 2 displays some characteristics of the data for different stocks, including the number of training and test records, and the mean and standard deviation of the closing price. The standard deviation is a number explaining how much the closing price varies while the mean closing price is the average closing price over the full period. The correlation value describes how much the 3 stocks within the same sector correlates to one another.

Automobile					
Company (ticker)	Training set	Test set	Mean closing price	Std closing price	Correlation
Ford (F)	3022	756	11.016805	3.529940	
Toyota (TM)	3022	756	103.342922	22.271931	0.725
Tata motors (TTM)	3022	756	22.342873	10.459595	

(a) Descriptive statistics for the Automobile stock group.

Petroleum					
Company (ticker)	Training set	Test set	Mean closing price	Std closing price	Correlation
BP (BP)	4578	1145	47.834454	10.687714	
ExxonMobil (XOM)	4578	1145	64.702839	21.681537	0.534
Lukoil (LUKOY)	4578	1145	44.846576	25.595065	

(b) Descriptive statistics for the Petroleum stock group.

Technology					
Company (ticker)	Training set	Test set	Mean closing price	Std closing price	Correlation
IBM (IBM)	5995	1499	98.559237	56.165728	
Intel (INTC)	5995	1499	23.044384	14.420652	0.662
Microsoft (MSFT)	5995	1499	31.705283	28.355172	

(c) Descriptive statistics for the Technology stock group.

Table 2: Descriptive statistics for the datasets used.

3.3 Split

The datasets are split into a training set containing 80% of the first records, and a test set containing the remaining 20%. Both the training and test sets have an input and an expected output component, which are structured as follows:

$$training\_x = \begin{bmatrix} [1, ..., 60] \\ [2, ..., 61] \\ . \\ . \end{bmatrix} \qquad training\_y = \begin{bmatrix} [61, ..., 65] \\ [62, ..., 66] \\ . \\ . \end{bmatrix}$$

(a) Illustration of the x- parameters (b) Illustration of the y- parameters

Figure 1: Illustration of the data: the first array in $training_y$ is the expected output of the model after being fed with the first array in $training_x$

The training and test input sets contains an array of arrays, where each inner array consists of stock data from the last 60 days. We chose a look-back window of 60 days because after analyzing the data we saw that 60 days was sufficient to capture one whole seasonal cycle. Figure 1a illustrates the structure of the input sets. The training and test output sets also contains an array of arrays, where the inner array consists of the correct closing prices for the next five days after the last record in the corresponding input set. Figure 1b illustrates the structure of the output sets.

3.4 Preprocessing

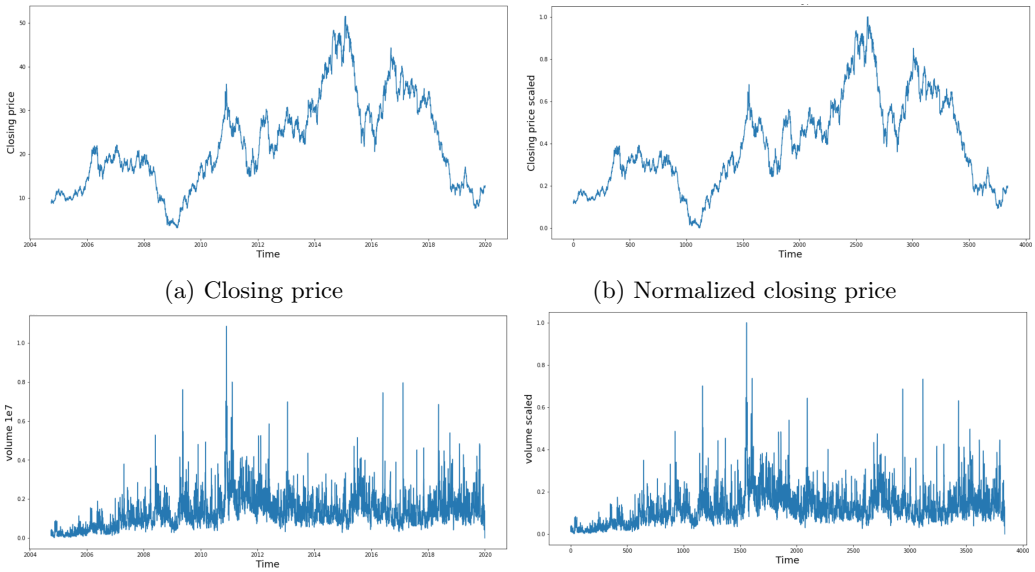
For our model to be able to train efficiently and predict accurately it has to be trained on data that is pre-processed appropriately to fit what the model should receive as input. All input data are rounded to two decimals, to remove excess noise during training. Additionally, a check is made to ensure that the dataset does not contain any missing or invalid values. The data is also reshaped to the dimensions required by the model. The dimensions are as follows:

Dataset	Dimensions	Description
Input set	(x, 60, 5)	(Records, time steps, features)
Output set	(y, 5)	(Records, number of outputs: 1-5 days)

Table 3: Dimensions of the input and output sets

3.4.1 Normalization

The data is normalized before it is used in training. This process is done individually for each feature because of the different units they have: volume is measured in number of shares traded that day which is usually a much higher number than that of the price features measured in USD ($). Individual scaling ensures that each feature can be weighted properly by the model. The normalization transforms the values to a range between 0 and 1. Figure 2 shows a plot of the data before and after normalization.



(a) Closing price

(b) Normalized closing price

(c) Volume

(d) Normalized volume

Figure 2: Difference before and after normalization

4 Methods

In this section, we go into the details of our selected machine learning method. We explain our chosen network architectures, objective function, optimizing algorithm, and hyperparameters.

4.1 LSTM neural network for supervised time series forecasting

Initially, we thought of two different machine learning methods which could work for our experiment: Either train a model through reinforcement learning and have it simulate purchasing and selling, or, train a model using supervised learning and have it predict the closing price a number of days ahead. We ended up training our models through supervised learning as we believed it would give the best results, and would make it easier to interpret and illustrate the results. After selecting the machine learning method, we had to choose what type of neural network to go for. We decided on LSTM networks as it is well researched and has proven to give accurate forecasts when predicting on time series data. However, not much research have been published on LSTM networks for stock price prediction, so we figured we could contribute to that specific field of research.

4.2 Network architectures

For our experiment, we wanted to compare the prediction accuracy of a shallow LSTM neural network against a deep LSTM neural network. Both shallow and deep neural networks are able to approximate any continuous function [Hrushikesh, 2016], so it is reasonable to assume that the models will display similar results when predicting stock prices 1 to 5 days ahead. However, deep neural networks are known for creating deep representations, where at each level, the network learns a new more abstract representation of the input which ultimately may help increase the prediction accuracy. We decided to go for a shallow neural network containing only one LSTM layer, and a deep neural network containing 3 LSTM layers. The models were built using Tenserflow's keras library.

4.2.1 Node selection

When choosing the number of nodes to include in our LSTM layers, we decided to go for some common rules of thumb as there is no definite number of nodes that give the best results. The number of nodes is determined by the following rules:

- The number of nodes should be a geometric progression of 2, e.g 4, 8, 16, etc...
- The first layer should have nodes equal to around half of the number of input data features. In our case: 60 (time steps) * 5 (features) / 2 = 150.
- The next layers should have half the nodes of the previous layer.

Both the shallow and the deep network has a dense output layer containing 5 nodes corresponding to the model's prediction of the closing price for the next 5 days.

4.2.2 Activation function

For the dense output layer we chose a linear activation function, which is a common activation function for the output layer in regression problems like ours. To choose the activation function for the LSTM layers, we did a test to see which gave the best prediction accuracy. We created a shallow neural network with a single LSTM layer containing 128 nodes, and a dense output layer with 5 nodes (Figure 3). We ran three simulations, where we tested the sigmoid, relu, and hyperbolic tangent activation functions for the single LSTM layer. All models were trained and tested on the Intel stock data, with the same hyperparameters, discussed in section 4.4. To evaluate the models we calculated the average mean absolute squared error (MAPE) of its prediction 1 to 5 days ahead. We obtained the following results:

Activation function	Average MAPE (1 to 5 days)
Hyperbolic tangent	**2.322**
Rectified linear unit	2.579
Sigmoid	2.576

Table 4: Comparison of different activation functions

The model with a hyperbolic tangent activation function had a slightly lower MAPE and it was therefore chosen as the activation function for all our LSTM layers.

4.2.3 Shallow LSTM neural network

Below is an illustration of the shallow neural network used in our experiments. It contains an input layer, a single hidden LSTM layer with 128 nodes, and a dense output layer with 5 nodes:

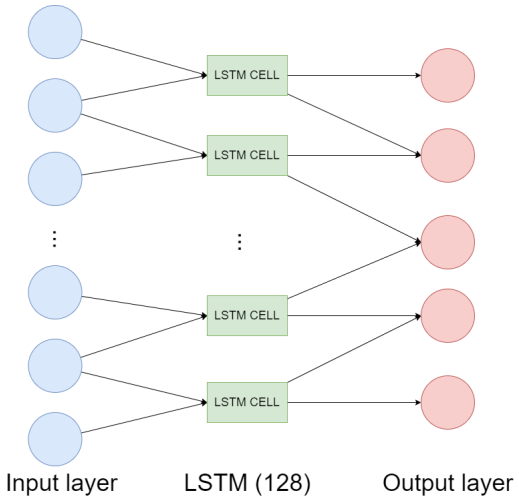


Figure 3: Shallow LSTM network

4.2.4 Deep LSTM neural network

Below is an illustration of the deep neural network used in our experiments. It contains an input layer, 3 hidden LSTM layers with a decreasing number of nodes determined by the rules described above (4.2.1), and a dense output layer containing 5 nodes:

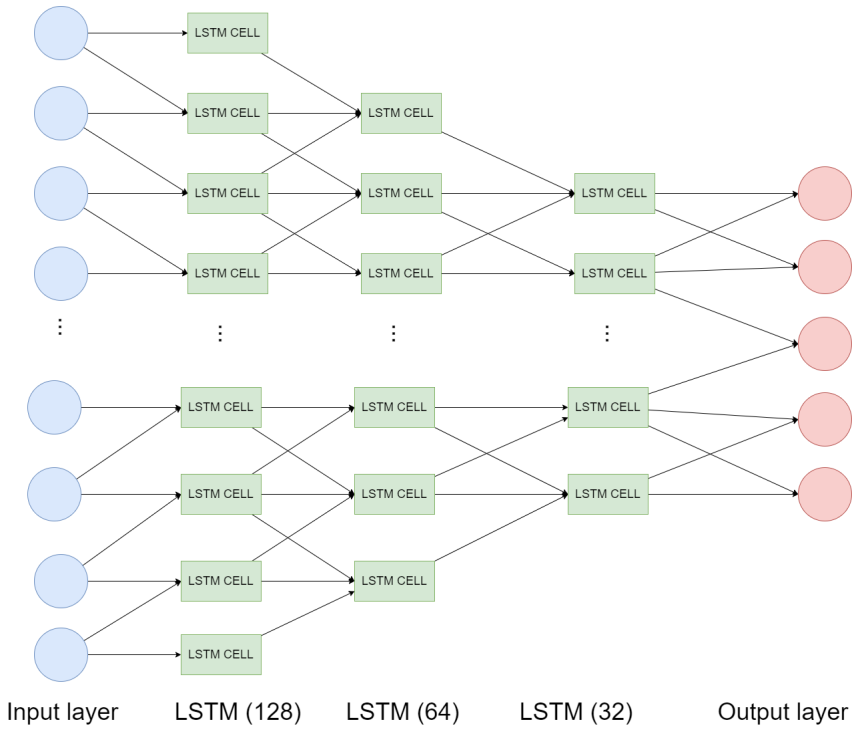


Figure 4: Deep LSTM network

4.2.5 Model output

Table 5 shows an example of the model's output. The 5 outputs corresponds to the 5 days it attempts to predict from the past 60 days of data. Day 1 corresponds to the model's 1-day

prediction, day 2 to the model's 2-day prediction, and so on.

Day 1	Day 2	Day 3	Day 4	Day 5
12.161984	12.238658	12.344881	12.458323	12.451051

Table 5: Shows an example of the model output

4.3 Objective function and optimizing algorithm

When choosing the objective function and optimizing algorithm, we performed similar experiments as for the activation function. We used the shallow network model, trained on the Intel stock data, and measured the average mean absolute percentage error of the models prediction for each combination of objective function and optimizing algorithm.

We tested the combination of two objective functions and two optimizing algorithms all commonly used for regression tasks. The results were as follows:

Objective function	Optimizing algorithm	Average MAPE (1 to 5 days)
Mean squared error (MSE)	Adam	**2.322**
Mean squared error (MSE)	Stochasic gradient descent	3.210
Mean absolute error (MAE)	Adam	2.434
Mean absolute error (MAE)	Stochasic gradient descent	2.590

Table 6: Comparison of different objective functions and optimizing algorithms

The combination of mean squared error as objective function and adam as optimizing algorithm gave the lowest MAPE and was therefore selected for our experiments.

4.4 Hyperparameters

A major reason for our selection of hyperparameters was to minimize overfitting on the training data. The hyperparameters we tested were batch size, dropout rate and early stopping patience level.

We decided to go for a batch size of 32. The reason was simply that we saw no significant improvement in prediction accuracy with a lower batch size. However, we saw a dramatically increase in training time for lower batch sizes. The research of [Breaul, 2015] and [Merity and Socher, 2017] shows that batch size has little effect on prediction accuracy for LSTM networks, so we felt comfortable leaving it at 32.

After each LSTM layer, we have a dropout layer with a rate of 0.2. The dropout layer helps with overfitting, and works by setting input units to 0 with a frequency of 0.2 at each step during the training time. That way, the network is forced not to rely on specific nodes to make predictions.

The last hyperparameter we tested was the amount of epochs to train for. We found it quite difficult to select a single number of epochs which gave the best results for every model and every stock. Therefore, we decided to implement an early stopping with a patience level of 20. For each epoch, the model calculates the prediction error on the training set, and a part of the test set (the validation set). If the error on the validation set increases from one epoch to the next, and does not hit a new low for the next 20 epochs, the training will stop. The reason for the high patience level of 20 is that we observed that the validation loss might increase for 10-15 epochs, but then hit a new low. The figure below illustrates an example of how the training and validation error changes across epochs:

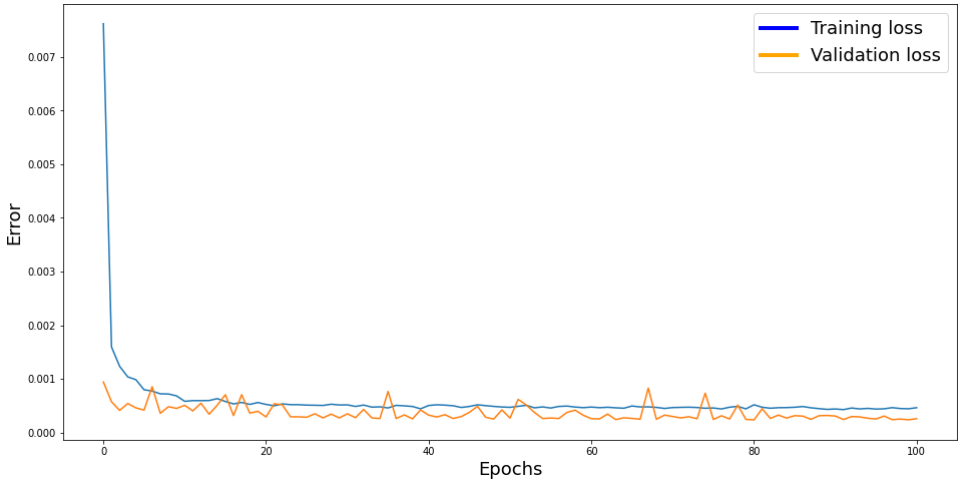


Figure 5: Loss on training set versus validation set during training

5 Results

In this section we present the results from our experiments. To start off, we describe our chosen evaluation metrics before the results are displayed and discussed. Finally, we elaborate on the effects on performance of different network architectures and the amount of data used during training.

5.1 Evaluation metrics

To evaluate the predicted stock prices both mean absolute percentage error (MAPE) and root-mean-square error (RMSE) are used. Equation 1 shows the formula for RMSE, and equation 2 the formula for MAPE:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(y_{pred,i} - y_{true,i}\right)^2} \tag{1}$$

$$MAPE = 100\% \cdot \frac{1}{n}\Sigma_{i=1}^{n}\left|\frac{y_{true,i} - y_{pred,i}}{y_{true,i}}\right| \tag{2}$$

where n is the number of observations, $y_{true,i}$ is the true stock price at time i and $y_{pred,i}$ is the predicted stock price at time i.

MAPE measures the prediction accuracy of the forecast in terms of percentages. It evaluates the absolute percentage error between the true value and the predicted value. MAPE is not suitable if the true value is close to 0, but that was not the case for our stock prices. RMSE is also used to measure the accuracy of the prediction. With RMSE, the error between the true value and the predicted value is measured in the unit of the value, in our case dollars. Our stocks varies in price range and RMSE will be a smaller number if the price average around a low value. False conclusions could therefore be drawn from just looking at RMSE when comparing different stocks. However, it still gives valuable information about the prediction. Both MAPE and RMSE are therefore used to evaluate the predicted stock prices as it gives a better understanding of the errors in terms of both percentages and dollars.

5.2 Forecasting results

Stock	Single/combined	Network	Day	MAPE	RMSE
Intel	Single model	1-layer	1 day	2.4770	1.2388
			5 day	3.3084	1.7511
		3-layer	**1 day**	**1.7353**	**0.9887**
			5 day	**3.0445**	**1.663**
	Combined model	1-layer	1 day	3.4281	1.7322
			5 day	4.6655	2.4088
		3-layer	1 day	3.3372	1.4764
			5 day	4.2775	2.0487

(a) Results predicting Intel stock. Lowest MAPE(%) and RMSE($) is highlighted for 1 and 5 day prediction.

Stock	Single/combined	Network	Day	MAPE	RMSE
BP	Single model	1-layer	**1 day**	**1.9525**	**0.9271**
			5 day	3.7053	1.6798
		3-layer	1 day	2.2935	1.0436
			5 day	3.3854	1.5336
	Combined model	1-layer	1 day	2.0512	0.9353
			5 day	**2.8581**	**1.3380**
		3-layer	1 day	2.3547	1.0728
			5 day	3.2242	1.4764

(b) Results predicting BP stock. Lowest MAPE(%) and RMSE($) is highlighted for 1 and 5 day prediction.

Stock	Single/combined	Network	Day	MAPE	RMSE
Ford	Single model	1-layer	1 day	2.0047	0.2691
			5 day	2.9923	0.3971
		3-layer	1 day	2.0335	0.2696
			5 day	3.0150	0.3974
	Combined model	1-layer	1 day	1.6789	0.2334
			5 day	2.8505	0.3772
		3-layer	**1 day**	**1.5763**	**0.2188**
			5 day	**2.8480**	**0.3701**

(c) Results predicting Ford stock. Lowest MAPE(%) and RMSE($) is highlighted for 1 and 5 day prediction.

Table 7: Forecasting results

Table 7 displays the prediction error (MAPE and RMSE) for all models when predicting the given stocks 1 and 5 days into the future. A single model is a model trained only on the stock it predicts on during evaluation, while a combined model is a model which is trained on 3 stocks within the same industry (Table 2). The highlighted values shows the best 1 and 5 day ahead prediction for a given stock.

For the Intel stock, the single model with a deep 3-layer network gave the lowest MAPE and RMSE for both the 1 day ahead and the 5 day ahead prediction. However, when predicting the stock

price of BP a different model produced the most accurate predictions. For BP it was the single model with a shallow 1-layer network which gave the best prediction 1 day into the future, while a shallow 1-layer combined model gave the most accurate prediction 5 days into the future. For the Ford stock, the model with the best prediction accuracy was the combined 3-layer model both 1 and 5 days ahead. As we can see, the most accurate model varies according to what stock is being predicted. A graphical representation of the results can be viewed in the Appendix A, where the stocks are predicted from January 2017 to January 2020.

5.3 Discussion

The MAPE of the results for 1-day predictions lies between 1.5 and 3.6. Compared to [Bin et al., 2018] (with an average MAPE ≤ 1.5) the error is relatively high, and they argued that a MAPE ≤ 0.75 is required to bring value for investors. Since 1 day ahead prediction is such a short time frame the need for high accuracy is important. Similarly, the 5-day prediction MAPE lies between 2.8 and 4.7. As we can see, the 5-day prediction is not as accurate as the 1-day prediction, which is reasonable. However, predicting 5 days into the future is still a relatively small time frame, so we were expecting a slightly lower MAPE. On the flip side, we believe it can still be a useful indicator for whether the stock price is going to increase or decrease in the next 5 days. For more precise stock price evaluation, though, it is less valuable.

Further tuning of hyperparameters like hidden layer size and number of epochs might be needed to achieve a higher prediction accuracy. Additionally, changes in the preprocessing step like for example selecting a longer or shorter look-back window (ours was 60 days) might also help reduce the MAPE and RMSE further. It is important to note, though, that the impact of these changes is likely to vary depending on what stock is being predicted. Our models are based on the general network architectures and parameters discussed in section 4, and they are meant to predict on a specific stock by training on that stock or the associated stock group. However, more specific models could be created for specific stocks, but that was not the goal here. One could also aim to create a model that can predict any stock, after being trained on a diverse number of stocks, but that was not our goal here either.

5.3.1 Single vs combined model

From the results in Table 7 it is not clear whether a single or combined model gives the most accurate forecasts. Our results depend on what stock is being predicted where a single model had the most accurate forecasts on the Intel stock. On the Ford stock it was the combined model which got the highest prediction accuracy, and on BP the results were split on the 1 day ahead and the 5 day ahead prediction. Therefore, we conclude that training a model on multiple stocks does not always increase the prediction accuracy. However, our combined models displayed a higher prediction accuracy than that of the single models when trained on stocks with a high correlation, as was the case for the automobile industry. It seems that the model is better able to understand the overall trend pattern when trained on multiple stocks with similar volatility patterns.

5.3.2 Deep vs shallow neural network

A closer look at the results for the shallow 1-layer models and the deep 3-layer models gave us no clear answer as to what network architecture results in the highest prediction accuracy. The shallow 1-layer network performed best on the BP stock while the deep 3-layer network performed best when predicting on the Intel and Ford stocks. We can therefore make no conclusion on whether a shallow LSTM network or a deep LSTM network is best for short-term stock prediction. An idea for future work would be to compare our models prediction accuracy when predicting stock prices on a longer time horizon, for example a year. In that case, the deep network might benefit with its ability of capturing deeper representation which might be necessary to capture the long-term trend.

6 Conclusion and future work

In this paper we have used LSTM neural networks for time series prediction of stock prices. Three stocks have been predicted, and we have compared the prediction accuracy of a shallow LSTM network to a deep LSTM network trained on one and multiple stocks within the same industry. From our results, we are not able to draw a definite conclusion as to which model architecture is best as both networks proved best at predicting different stocks. Additionally, models trained on a single stock had the best prediction accuracy for a certain stock, while models trained on multiple stocks gave better prediction accuracy for other stocks. Since 1-day and 5-day ahead forecasting demands high accuracy, our models do not produce desirable results. However, as mentioned in Results, we believe our model can give an indicator of the direction of the overall future trend.

To give a definite answer to what model architecture is best we suggest that more research should be done within the field of stock price prediction with LSTM networks. For future work, more stocks should be predicted and possibly over a bigger time horizon. Additionally, increasing the amount of training data might also help give a clearer view as to what model produces the best forecasts. In addition, training the combined model on more stocks can give valuable information on whether the combined model can utilize the information in other stocks. We recommend training on stocks that are somewhat well correlated. Another interesting idea for future work is to combine our LSTM method with natural language processing of news. As mentioned in the introduction, the stock market is not perfectly efficient. When news about a company goes viral, the new information will likely influence the stock price in some way. To be able to perform news sentiment analysis can possibly help improve the LSTM network's prediction.

Bibliography

M Balings. Evaluating multitple classifiers for stock price direction prediction. 2015.

Weng Bin, Xing Wang, and Lin Lu. Predicting short-term stock prices using ensemble methods and online data sources. 2018.

J Bollen. Twitter mood predicts the stock market. 2011.

Thomas Breaul. Benchmarking of lstm networks. 2015.

P Cootner. The random character of stock market prices. 1964.

Rajeev Dhir. Efficient market hypothesis: Is the stock market efficient? 2019.

M Hrushikesh. Learning functions: When is deep better than shallow. 2016.

Stephen Merity and Richard Socher. Regularizing and optimizing lstm language models. 2017.

Appendix

A Results

A.1 Intel

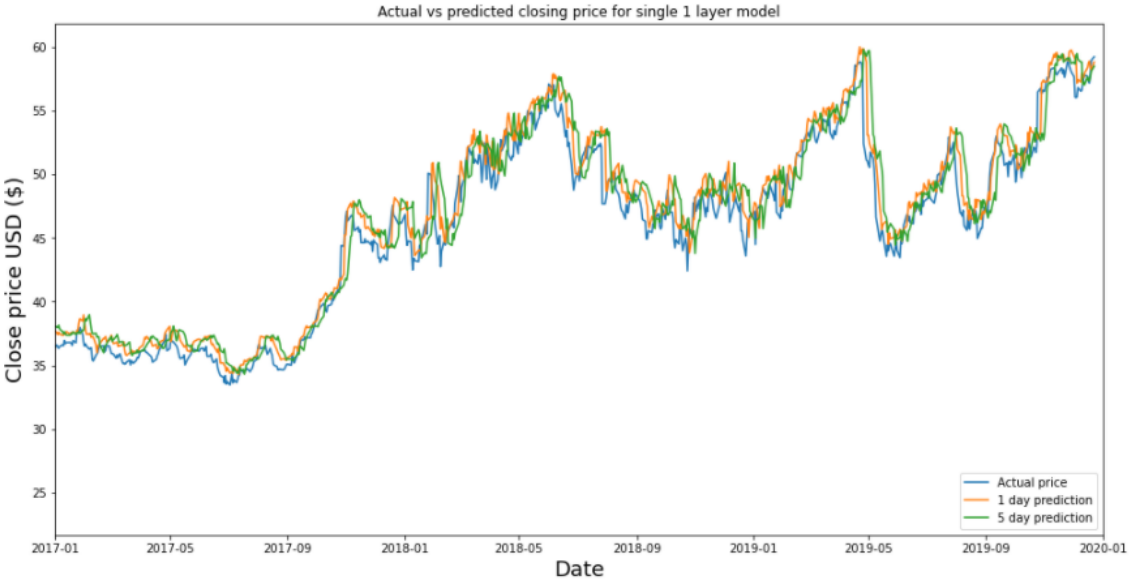


Figure 6: Results Intel single 1-layer model

Figure 7: Results Intel single 3-layer model

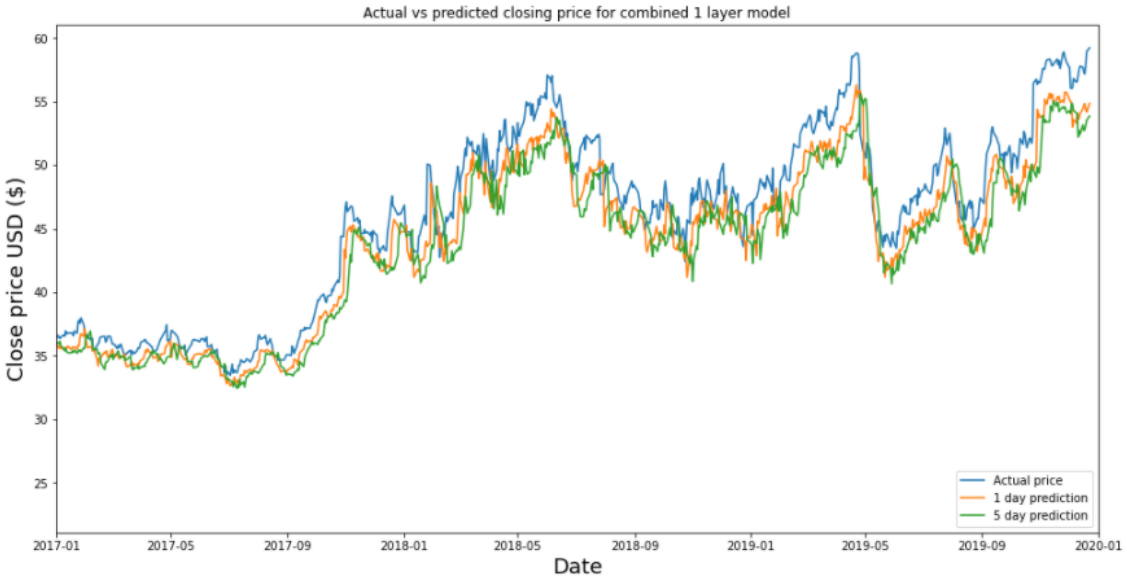


Figure 8: Results Intel combined 1-layer model

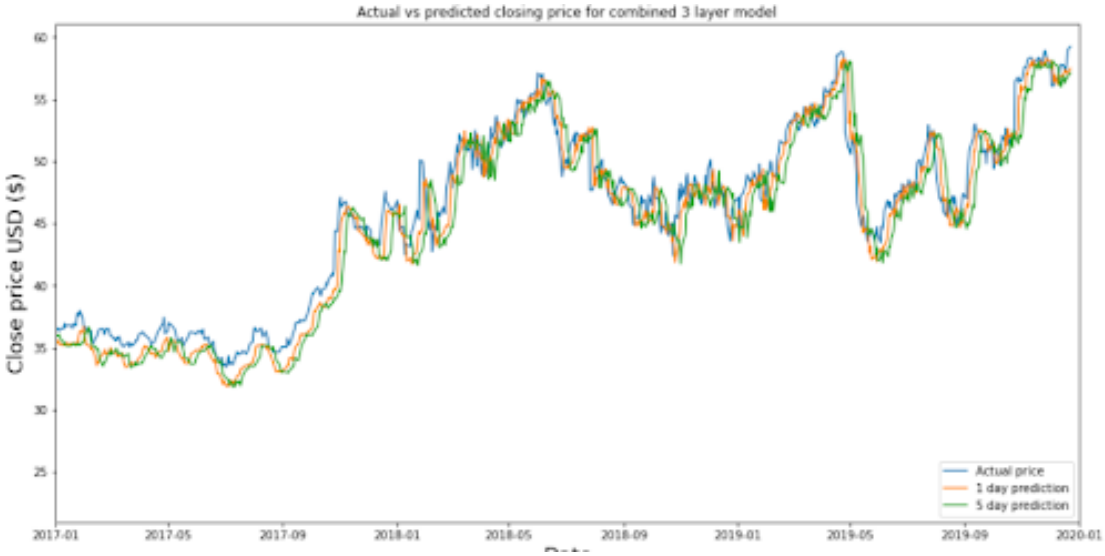


Figure 9: Results Intel combined 3-layer model

A.2 BP

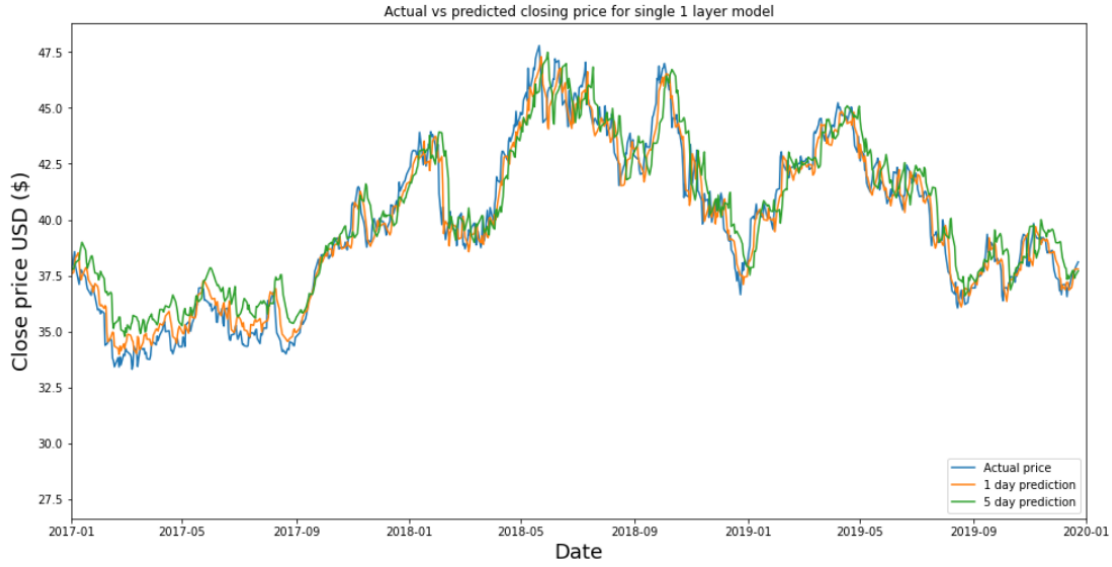


Figure 10: Results BP single 1-layer model

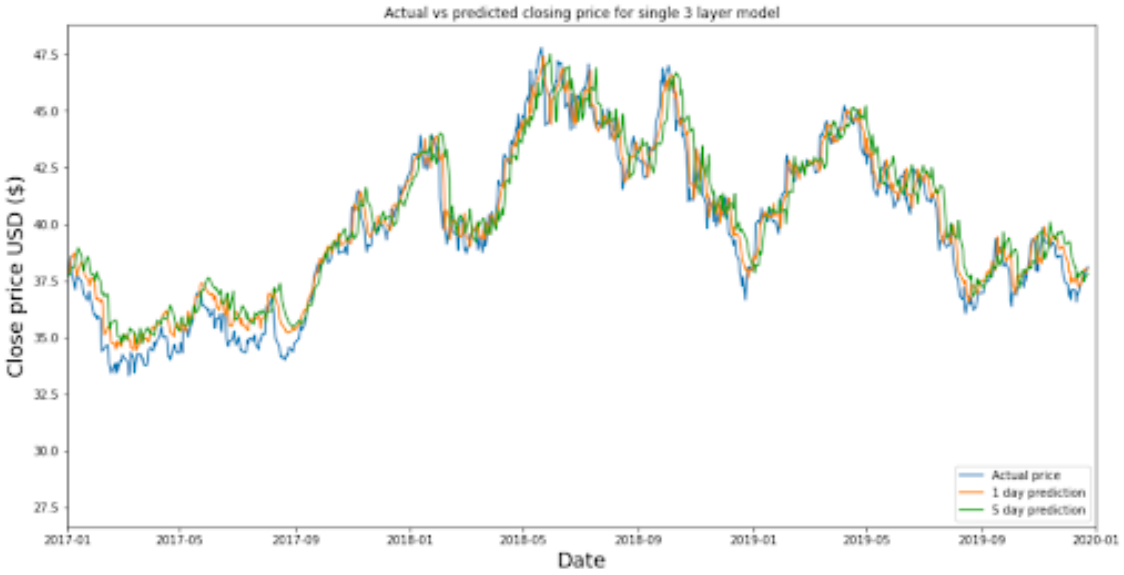


Figure 11: Results BP single 3-layer model

Figure 12: Results BP combined 1-layer model

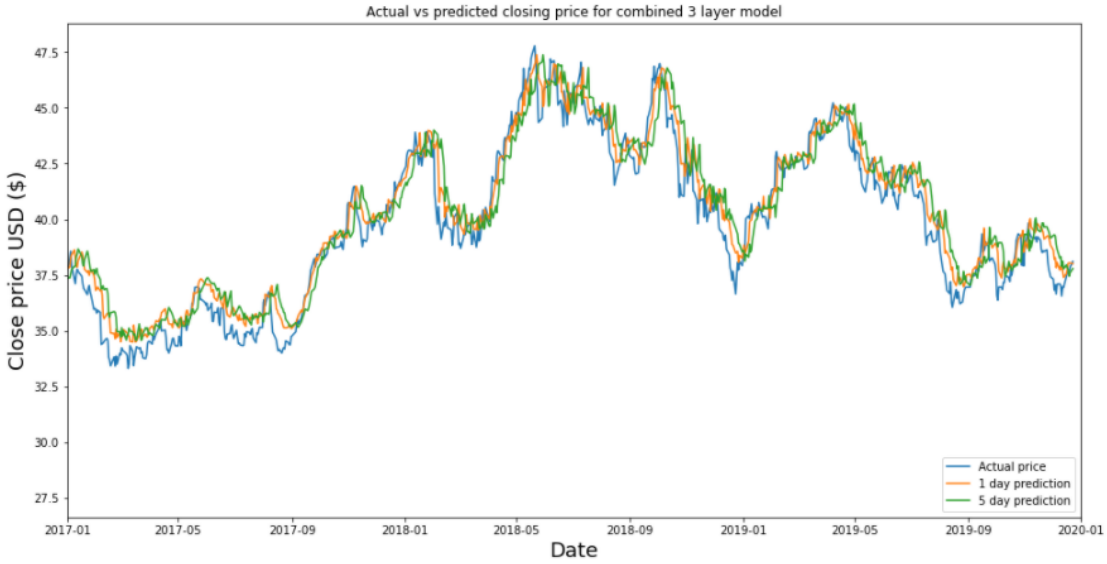


Figure 13: Results BP combined 3-layer model

A.3 Ford

Figure 14: Results Ford single 1-layer model

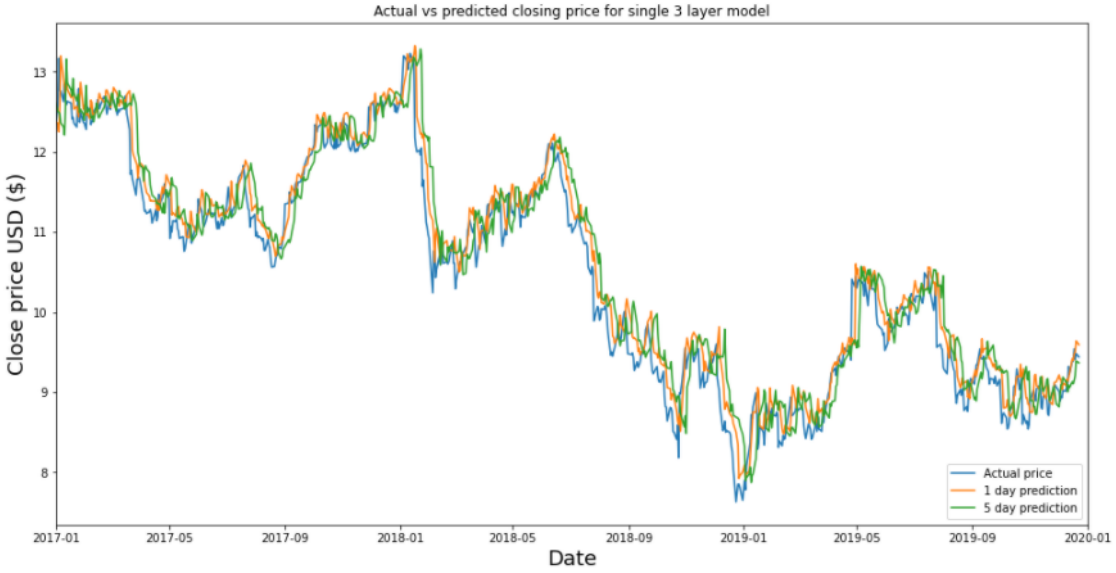


Figure 15: Results Ford single 3-layer model

Figure 16: Results Ford combined 1-layer model

Figure 17: Results Ford combined 3-layer model