Stock Price Forecasting

Application of Deep Learning on Stock Market.

Trier University

Submitted by

Tipu Sultan

Email: s4tisult@uni-trier.de

Matriculation Number: 1599320

Department: Applied Statistics

Submitted to

DR. Habil. Jan Pablo Burgard

M.Sc. Abrar Ahmed

Course: Multivariate Statistics

Contents

1. Abstract:

Since there is no guarantee in a corporate setting, forecasting stock price is quite challenging. The stock market for a company's financial performance becomes incredibly unpredictable, dynamic, and nonlinear due to a variety of macro and micro elements including politics, world economic circumstances, and unforeseen occurrences. Deep learning, commonly referred to as deep structured learning, is one of several machine learning techniques built on artificial neural networks and representation learning. Several deep learning-related techniques may be useful to estimate stock values. Drift, Naive, Artificial Neural Networks, and Random Forests are a few examples. Despite not being frequently used for financial time series prediction, they are intrinsically appropriate to this domain [1]. Long short-term memory (LSTM) networks will be employed in this study to anticipate the stock price. The data for "Apple Inc." will be obtained from the yahoo finance website starting in January 1990 and ending in July 2022. The model will be trained using the first 80% of the data. The model will next be validated using the subsequent 10% of data. And the remaining information will be utilized to evaluate the forecasting model. The same process will then be done using data collected starting from January 2020. Finally, this paper will compare the findings to actual data to see what actually happened and how accurate the predictions are. Additionally, it will describe how the models respond when given various weights in the training set of data. Python will be utilized as the programming language for the data analysis and implementation of the results to show this article. We will see a more accurate forecast for the short-term data than the long-term data because of the theoretical mechanism of the LSTM approach, which is heavily reliant on the training data set and the test data set.

---

1. Thomas Fischer, Christopher Krauss. Deep learning with long short-term memory networks for financial market prediction. European Journal of Operation Research (2018).

2. Introduction:

Although technology is far from being able to do so, deep learning attempts to mimic the human brain. Enabling systems to cluster data and make predictions with astonishing accuracy. According to Abhaya Kumar Sahoo, and Chittaranjan Pradhan (2019), deep learning is the part of machine learning which uses different layers of neural networks that decide classification and prediction. These neural networks make an effort to mimic how the human brain functions however, they fall far short of being able to match it. Enabling it to "learn" from vast volumes of data. Additional hidden layers can assist to tune and improving accuracy even if a neural network with only one layer can still produce approximation predictions. Deep learning is at the heart of many artificial intelligence (AI) products and services that enhance automation, carrying out mental and physical activities without the need for human participation. Digital assistants, voice-activated TV remotes, and credit card fraud detection are just a few examples of common goods and services that are powered by deep learning technology. As well as cutting-edge technology like autonomous vehicles.

Forecasting stock prices is an arduous task due to the unpredictable nature of the stock market [2]. The stock price is less well-liked than the other divisions in the machine learning forecasting industry. Many variables contribute to the stock market's excessive volatility and nonlinearity. Examples include politics, geography, natural disasters, unexpected human behaviour, and so on. The future is unpredictable. We can't forecast every possibility. Individual stocks don't always follow the market as a whole. Projections can be hazy. Despite the fact that the financial market's chaotic character makes forecasting infamously difficult, preliminary research suggests that machine learning algorithms can recognise non-linear patterns

---

2. Chandola, D., Mehta, A., Singh, S. *et al.* Forecasting Directional Movement of Stock Prices using Deep Learning. *Ann. Data. Sci.* (2022).

in financial market data. More than 100 capital market anomalies that essentially rely on return prediction signals to outperform the market are surveyed by Jacobs (2015) or Green, Hand, and Zhang (2013). To build a connection between these return-predicting signals and future returns, financial models are often transparent in form and incapable of capturing intricate non-linear connections. A noteworthy example might be the work of Huck (2009), Huck (2010), Takeuchi and Lee (2013), Moritz and Zimmermann (2014), Dixon, Klabjan, and Bang (2015), as well as additional references in Atsalakis and Valavanis (2009) and Sermpinis, Theofilatos, Karathanasopoulos, Georgopoulos, and Dunis (2013). the most current research by Krauss, Do, and Huck (2017) uses the same data set for comparison. The authors projected all S&P 500 components from 1992 to 2015 using deep learning, random forests, gradient-boosted trees, and other ensembles. One significant conclusion is that deep learning algorithms perform worse than gradient-boosted trees and random forests with returns of 0.33 percent and 0.43 percent respectively each day before transaction costs [3].

The latter finding is unexpected considering that deep learning "dramatically improved the state-of-the-art in voice recognition, visual object identification, object detection, and many other categories" [4]. At first glance, one may anticipate comparable advancements in the field of time series forecasts. Krauss et al. (2017, p. 695) argue that deep learning "may be that there are well configurations in parameter space to further improve the performance," even though "neural networks are notoriously difficult to train." [5].

_____

3. Christopher Krauss, Xuan Anh Do, and Nicolas Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. European Journal of Operation Research (2017).
4. LeCun, Bengio, and Hinton. Deep Learning. Nature (2015).
5. Thomas Fischer, Christopher Krauss. Deep learning with long short-term memory networks for financial market prediction. European Journal of Operation Research (2018).

3 Methodology:

3.1 LSTM Network Model:

Every day the financial market does not begin from scratch. It does have a pattern and is influenced by its prior phases. This pattern can't be found in conventional neural networks which looks like a significant flaw. It's unclear how a conventional neural network might utilize its analysis of earlier financial market events to predict future ones. This problem is addressed by recurrent neural networks.

A recurrent neural network (RNN) is a neural network that simulates a discrete-time dynamical system that has an input x_t, an output y_t , and a hidden state h_t. In our notation the subscript t represents time. The dynamical system is defined by

$$h_t = f_h(x_t, h_{t-1}) \quad (1)$$

$$y_t = f_o(h_t), \quad (2)$$

where f_h and f_o are a state transition function and an output function, respectively. Each function is parameterized by a set of parameters; θ_h and θ_o.

Given a set of N training sequences $D = ((x_1^{(n)}, y_1^{(n)}), \ldots, (x_{T_{(n)}}^{(n)}, y_{T_{(n)}}^{(n)}))_{n=1}^{N}$, the parameters of an RNN can be estimated by minimizing the following cost function:

$$J(\theta) = \frac{1}{N} \sum_{n=1}^{N} \sum_{T=1}^{T_n} d(y_t^{(n)}, f_o(h_t^{(n)})), \quad (3)$$

where $h_t^{(n)} = f_h(x_t^{(n)}, h_{t-1}^{(n)})$ and $h_o^{(n)} = 0$. d(a, b) is a predefined divergence measure between a and b, such as Euclidean distance or cross-entropy [6]. RNNs have been used with remarkable success in recent years.

---

6. Pascanu, Razvan and Gulcehre, Caglar and Cho, Kyunghyun and Bengio, Yoshua. How to Construct Deep Recurrent Neural Networks. arXiv. (2013)

For example, language modeling (Mikolov, 2012), speech recognition (Graves et al.,2013), machine translation (Kalchbrenner & Blunsom, 2013), online handwritten recognition (Graves et al.,2009), and so on.

From equations (1) and (2) it is clear that the potentiality for RNNs to make a connection between earlier data and the current position is one of the key advantages. For example, utilizing earlier stock data may help us comprehend the current stock price. RNNs would be very helpful if they could accomplish this. Is it possible? It varies. Sometimes all that is required to complete the current work is to glance at the most recent information. Let us consider a scenario where a model is attempting to forecast a company's product sales volume. If there is a discount every Saturday, chances are that the sales will go up the next Saturday as well. RNNs can learn to utilize the prior information in such circumstances when there is close proximity between the pertinent information and the location where it is required. It could be possible through the RNN network according to the theory. However, let us consider attempting to forecast sales when discounts are offered every Saturday and this Friday is a holiday. In this case, the public holiday in addition to the discount will cause this Saturday's sales to increase. It is feasible for a very massive difference to grow between the numerous features and the point when it is required.

Unfortunately, RNNs lose their ability to learn to correlate the information as that gap widens [7]. Chang et al. (2017) with the Dilated RNNs reduced the number of parameters to enhance computation efficiency[8]. Theoretically, RNNs can handle such "long-term dependencies" with ease. To solve this kind of issue, a person might carefully choose the characteristics for them. Sadly, RNNs don't seem to be able to learn them in practice.

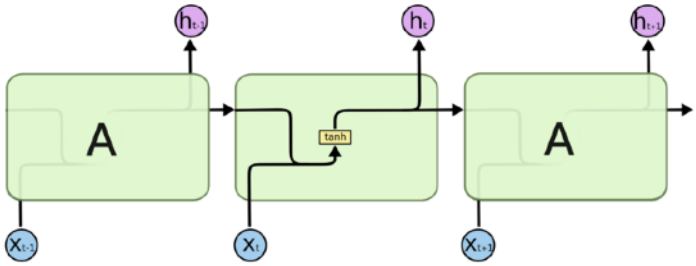
---

7. Staudemeyer, Ralf C. and Morris, Eric Rothstein. Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks (2019).
8. Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, Guangjian Tian. Do RNN and LSTM have Long Memory? PMLR (2020).

Hochreiter (1991) and Bengio, et al. (1994) investigated the issue in-depth and discovered some appropriate reasons why it would be challenging. Thankfully, this issue is not present on the LSTM Network. The usage of “LSTM Network" is a particularly specific type of recurrent neural network that performs many tasks far better than the regular version. Because Long short-term memory networks are a special kind of RNN that can learn long-term dependencies [9]. The problem of long-term dependencies is theoretically examined in further depth in [10]. The report also provides a quick overview of several solutions to this issue.

Long Short-Term Memory Networks, most often referred to as "LSTMs," are a unique class of RNNs that can recognize long-term dependencies. They were first presented by Hochreiter & Schmidhuber (1997), and several authors developed and popularized them in subsequent works. They are currently frequently utilized and perform incredibly well when applied to a wide range of issues. Purposely, LSTMs are created to prevent the long-term reliance issue. They make little effort to learn. Remembering knowledge for extended periods of time is their default habit.

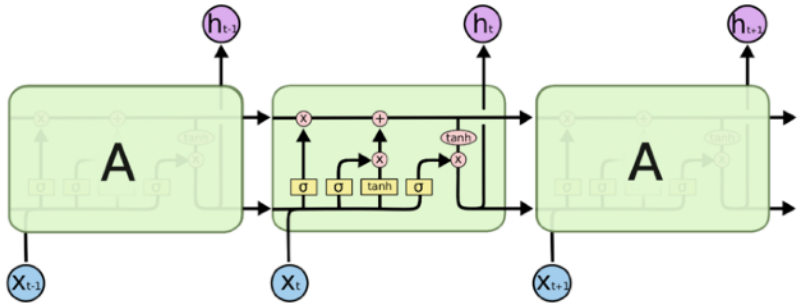
All recurrent neural networks have the structure of a set of repeating neural network modules.



The repeating module in a standard RNN contains a single layer.

9. Shahzad Muzaffar, Afshin Afshari. Short-Term Load Forecasts Using LSTM Networks. Science Direct (2019).
10. Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Ju¨rgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A Field Guide to Dynamical Recurrent Neural Networks (2001).

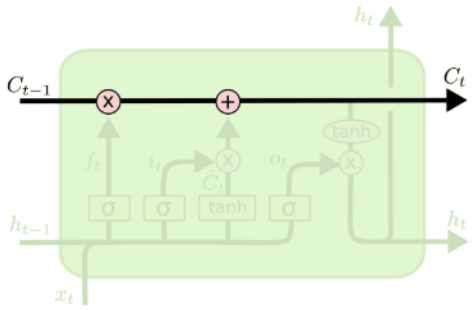
This recurring module in typical RNNs will be made up of just one tanh layer. There are four neural network layers, interacting in a highly unique way, as opposed to just one in RNN. The diagram given below is describing how the four layers are connecting one to another to capture long-term dependencies.



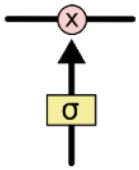
The repeating module in an LSTM contains four interacting layers.

Each line in the above figure conveys a full vector from one node's input to another's output. The yellow boxes are learned neural network layers, while the pink circles represent point-wise operations like vector addition. Concatenation is shown by lines merging, whereas lines forking indicate that their content has been replicated and is being sent to other destinations.

The horizontal line that runs across the top of the figure and represents the cell state is the secret to LSTMs. The cell state resembles a conveyor belt in certain ways. With only a few tiny linear interactions, it proceeds directly down the whole chain. Information can very easily continue to travel along it unmodified.

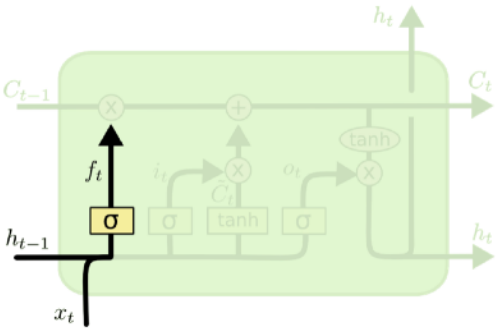


The LSTM may modify the cell state by removing or adding information, which is carefully controlled via gates. Information can pass via gates on a purely voluntary basis. They consist of a point-wise multiplication process and a layer of sigmoid neural networks.



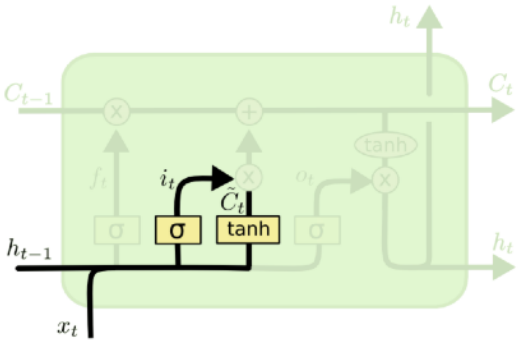
Indicating how much of each component should be allowed through, the sigmoid layer generates values between zero and one. When a value is zero, "let nothing through," and when a value is one, "let everything through," respectively. These three gates serve to safeguard and regulate the cell state in an LSTM.

The first step is to choose which information about the cell state must be ignored by the LSTM. To put this concept into practice, a sigmoid layer called the "forget layer" is responsible. It examines each number of h_{t-1} and x_t in the cell state C_{t-1}. The cell state C_{t-1} produces a number between 0 and 1. Completely keeping something is symbolized by 1 whereas completely getting rid of something is represented by 0.



$$f_t = \sigma(W_f \,.\, [h_{t-1,} x_t] + b_f) \qquad (4)$$

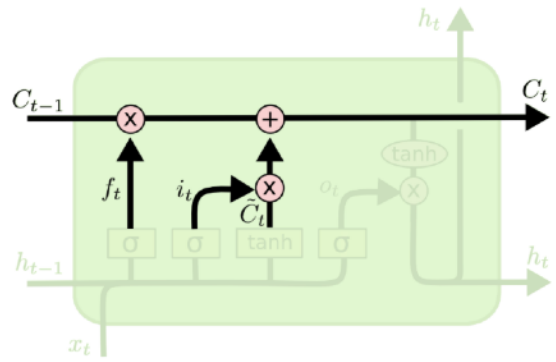
The subsequent stage will involve the information that will be kept in the cell state. There are two parts to it. The "input gate layer," a sigmoid layer, first determines which values will be updated. Then a tanh layer generates a vector of fresh potential values $\widehat{C}_t$ that might be added to the state. These two will be combined in the subsequent phase to provide an update to the state.



$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i) \qquad (5)$$

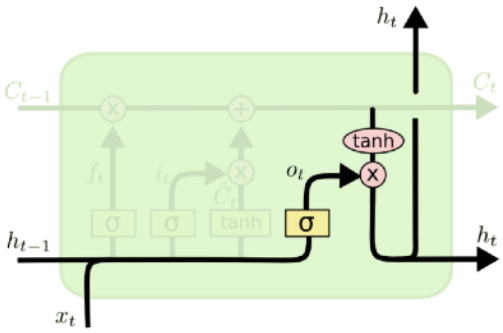
$$\widehat{C}_t = tanh(W_c . [h_{t-1}, x_t] + b_c) \qquad (6)$$

The old cell state is then meant to be updated. C_{t-1} into the newly created cell state C_t. What should be done has already been decided in the preceding states. Now It must be completed. f_t will be multiplied by the previous state to erase the data that the system previously determined to erase. After that, we append $i_t * \widehat{C}_t$. According to how much we decided to update each state value. These are the new candidate's values.



$$C_t = f_{t-1} * C_{t-1} + i_t * \widehat{C_t} \qquad (7)$$

Last but not least, we must choose what we want to produce. This output will be filtered and depends on the status of our cell. We first run a sigmoid layer to determine which portions of the cell state will be output. The cell state is passed through tanh and multiplied by the output of the sigmoid gate to only output the sections we specified after pushing the values to be between -1 and 1 [11].



$$o_t = \sigma(W_o . [h_{t-1}, x_t] + b_o) \qquad (8)$$

$$h_t = o_t * tanh . (C_t) \qquad (9)$$

3.2 Long Short-Term Memory Differentiations:

According to C. Olah (2015), The LSTM described above is not universal. It appears that a slightly different variant is used in practically every study that employs LSTMs. Gers & Schmidhuber (2000) introduce a popular LSTM version that features the addition of "peephole connectors". Surprisingly, LSTM augmented by "peephole connections" from its internal cells to its multiplicative gates can learn the fine distinction between sequences of spikes separated by either 50 or 49 discrete time steps, without the help of any short training exemplars [12].

11. Christopher Olah. Understanding LSTM Networks. Google research (2015).
12. F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count,". IEEE-INNS-ENNS International Joint Conference on Neural Networks (2000).

They allowed the gate layers to observe the cell state.

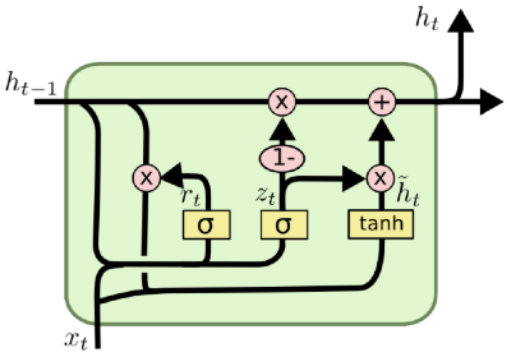
$$f_t = \sigma(W_f \,.\, [C_{t-1}, h_{t-1}, x_t] + b_f) \quad (10)$$

$$i_t = \sigma(W_i \,.\, [C_{t-1}, h_{t-1}, x_t] + b_i) \quad (11)$$

$$o_t = \sigma(W_o \,.\, [C_t, h_{t-1}, x_t] + b_o) \quad (12)$$

The design above includes peepholes for all the gates, although many documents only have partial peepholes.

The Gated Recurrent Unit, or GRU, developed by Cho et al. (2014) is a little more dramatic version of the LSTM. On the tasks of modeling speech signals and polyphonic music, they assessed these recurrent units. The study demonstrated that these more sophisticated recurrent units are superior to more conventional recurrent units like tan(h) units. They also discovered that GRU and LSTM are comparable. It creates a single "update gate" by fusing the forget and input gates. Along with making various adjustments, it integrates the hidden state and cell state. The developed model, which is more straightforward than traditional LSTM models, has been gaining popularity.



$$z_t = \sigma(W_z \,.\, [h_{t-1}, x_t]) \quad (13)$$

$$r_t = \sigma(W_r \,.\, [h_{t-1}, x_t]) \quad (14)$$

$$\widehat{h_t} = tan(h) \,.\, [W \,.\, [r_t * h_{t-1}, x_t]] \quad (15)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \widehat{h_t} \quad (16)$$

Greff, et al. (2015) compare several well-liked variations and find that they are all very similar. More than ten thousand RNN designs were put to the test by Jozefowicz et al. (2015), who discovered a few that performed some tasks better than LSTMs. Only a few of the more noteworthy LSTM variations are included here. Many others exist, such as Depth Gated RNNs by Yao et al (2015). Long-term dependencies can also be addressed using an entirely different strategy, such as Clockwork RNNs by Koutnik et al (2014).

LSTM networks can accomplish whatever RNNs might be capable of with much more precision. They are in fact superior to RNNs. Although they might be frightening, LSTMs do offer superior outcomes and represent a significant advancement in deep learning. With more of these technologies emerging, it is possible to anticipate more precise forecasts and a greater comprehension of the available options.

3.3 Experiment:

The same Apple Inc. stock price data is divided into two portions to determine the model's accuracy. Yahoo finance's official website was used to obtain Apple Inc.'s closing stock price. Data is available from 1990 through 2022. To implement the whole experiment 01 and experiment 02 Python programming language was used. 'Keras', a high-level, deep learning API (Application Programming Interface) developed by Google for implementing neural networks is used to build the models. Data from 1990 through 2022 was used to conduct experiment 01. Then, experiment 02 was conducted using the same data, starting from 2020. To execute the entire experiment, the closing stock price of the stock chart to its corresponding dates is used. In experiment 01, a python data frame is initially created with four variables according to their dates to train the model. Which are called "Target 1", "Target 2", "Target 3", and "Target". The goal is to get the model's output, while the remaining 3 variables are its inputs. The prior stock price on which the "Target" value is based is designated as "Target 1", "Target 2" and "Target

3" are the preceding values of the Target values that correspond to their dates respectively. The target price is the anticipated stock price of its prior values. The four variables all represent the same factor called the rate of price variation. The entire data frame must now be transformed into a single NumPy array and made into a single univariate variable in the following step. Then, to train the model, we divided the entire collection of four variables into three sets. The model is trained using the first 80% of the data set. The model was validated using the next 10% of the data set, and the actual events of that period were compared to the LSTM network's forecast using the last 10% of the data set. It necessarily does not have to be 80% data to train and 10% data to validate the model. One can take a different amount of data to train and validate the model. Technically it might be better to take more than 60% data to train and 10% data to validate the model. Because through more information from data, the model can learn more. In this case, the model will allocate all the information from the first 80% of data to learn and 10% of data to check the validity of the model. Not to mention that the last 10% of data is actual data. We are using this data to compare the forecast results. "TensorFlow", a library for Python from "Keras", has been used to do this. The same procedure is repeated in experiment 02. Where the stock price started from 2020. The algorithm outputs three sets of stock prices at the end. The first one learns from the stock price which is similar to the real stock price. The following two determined stock prices are shown in the "5. Results" section below. The same process is performed in experiment 02 where the stock price data started from 2020.

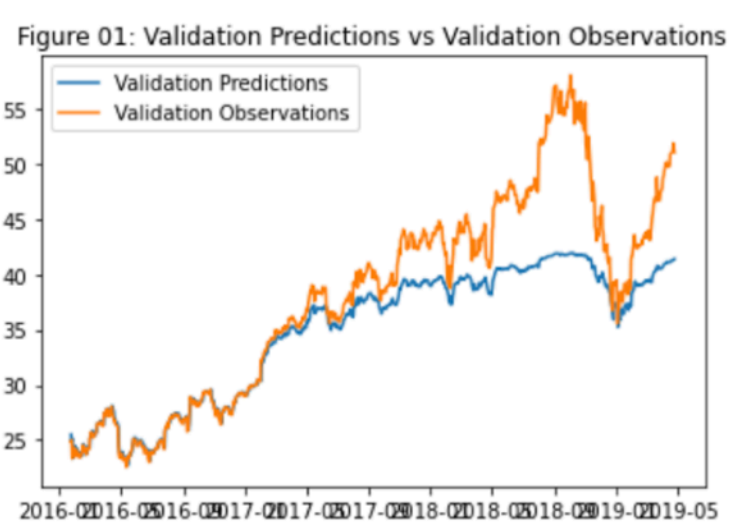
4. Data:

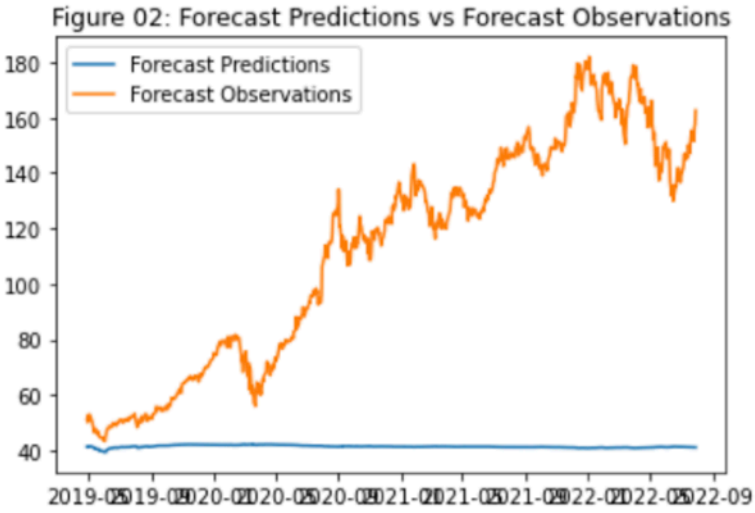
In this paper, we'll be using Apple Inc.'s daily stock price, which spans from January 1990 to July 2022. The data's source is the official Yahoo Finance page. The information represents the daily closing stock price going back to 1990.

5. Results:

5.1 Discussion:

The results of experiment 01, where the data started from 1990.





From experiment 01 it can be observed:

1. In figure 01, Validation prediction is following the validation observation pattern. But the correlation is very little.
2. In figure 02, the forecast prediction is far away from following the pattern of forecast observation. The correlation is nearly zero.

The results of experiment 02, where the data started from 2020.

Figure 03: Validation Predictions vs Validation Observations

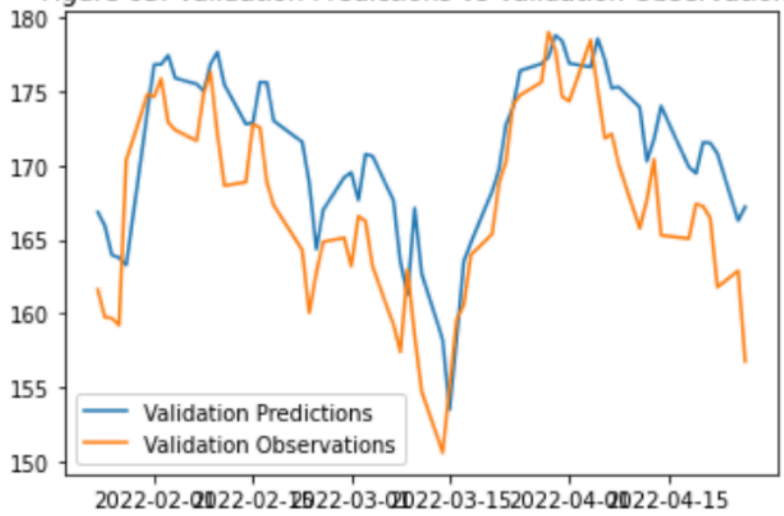
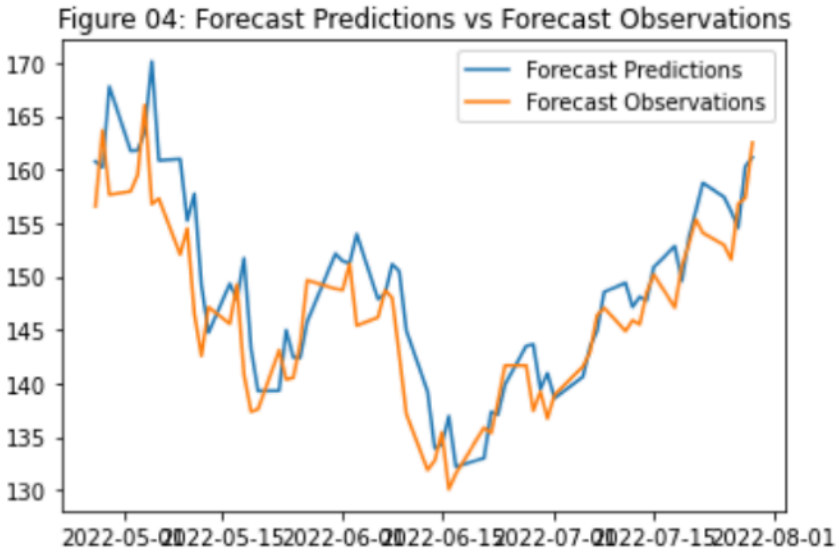


Figure 04: Forecast Predictions vs Forecast Observations



From experiment 02 it can be observed:

1. In figure 03, the pattern of the validation prediction and validation observation are almost alike.
2. In figure 04, the pattern of forecast prediction and forecast observation is almost perfect and the correlation is extremely high.

It is evident from the outcomes of experiment 01 and experiment 02 that experiment 02's model is more accurate than experiment 01's model. Why is the obvious question that follows? The data hold the answer. Because the data is extremely sensitive to the LSTM networks. We controlled the train data set model in the LSTM network. In experiment 01, the model used all the data to be supervised from 1990. It is well known that the stock market is unpredictable. Most likely, the knowledge that started from 1990 is no longer applicable in the modern world. As a result, the model of experiment 01 is not properly supervised. On the other hand, experiment 02's model is remarkably precise. Because it was supervised by using the data beginning from 2020. It could be restricted to all the information that represents the modern stock market. We all know covid-19 pandemic happened in 2109 and the stock market had taken the biggest shock during the pandemic. And still, it has the effect. It might be another possible reason that experiment 02 focused on that information and adjusted it with the model.

5.2 Some Drawbacks:

The hypothesis states that the LSTM networks perform better in sequential areas [13]. It is capable of gathering data in such a way that it is possible to assume that each pattern depends on the preceding pattern. In terms of the stock market where the sequence is very important, the LSTM network opens a lot of opportunities. Nonetheless, From the experiment and results above, the following drawbacks can be mentioned.

1. It easily overfits.
2. It is quite susceptible to various weight initializations.
3. LSTM networks require more time to train.
4. Additional machine memory is required to train LSTM networks
5. LSTM networks are extremely sensitive to the data.

13. Reimers, Nils and Gurevych, Iryna. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. arXiv (2017).

6. Conclusion:

Even though a huge sample works excellently for forecasting in general statistics, we can learn something distinct about the financial market from the LSTM networks. Sometimes a big sample size is insufficient and deceptive for model supervision. People may infer from the results that LSTM networks are limited in their ability to handle enormous volumes of financial data, but it can be the contrary. Every restriction uncovers opportunity. The LSTM networks may be used to find unwanted stock market information. Even though the data appears to be predictive, it may also be deceptive. LSTM networks have a lot of potential applications in the stock market. With the help of LSTM networks, we may clean the data set in addition to the forecast. We can select the precise data set that most accurately reflects the current stock market. And eliminate the information that was crucial in earlier years but is no longer relevant. The LSTM networks can include additional potentiality, such as data validation. We can verify the provided financial data to see whether it can be utilized to forecast or not.

7. References:

- Audeliano Wolian Li Guilherme Sousa Bastos. Stock Market Forecasting Using Deep Learning and Technical Analysis: A Systematic Review. IEEE Access (2020).
- A. Kumar. Who gambles in the stock market? The Journal of Finance (2009).
- Rafal Jozefowicz, Wojciech Zaremba, Ilya Sutskever. An Empirical Exploration of Recurrent Network Architectures. Proceedings of the 32nd International Conference on Machine Learning (2015).
- B.N. Lehmann. Fads, martingales, and market efficiency The Quarterly Journal of Economics (1990).
- Chung, Junyoung and Gulcehre, Caglar and Cho, KyungHyun and Bengio, Yoshua. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv (2014).
- Christopher Olah. Understanding LSTM Networks. Google research. (2015).
- F. A. Gers, J. Schmidhuber and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," in Neural Computation. MIT Press (2000).
- G. Sermpinis et al. Forecasting foreign exchange rates with adaptive neural networks using radial-basis functions and particle swarm optimization European Journal of Operational Research (2013).
- H. Jacobs et al. On the determinants of pairs trading profitability Journal of Financial Markets (2015).
- H. Jacobs. What explains the dynamics of 100 anomalies? Journal of Banking & Finance (2015).
- N. Huck. Pairs trading and outranking: The multi-step-ahead forecasting case European Journal of Operational Research (2010).

- N. Jegadeesh. Evidence of predictable behavior of security returns. The Journal of Finance (1990).
- Thomas Fischer, Christopher Krauss. Deep learning with long short-term memory networks for financial market prediction. European Journal of Operation Research. (2018)
- S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation. MIT Press (1997).
- Srivastava, Rupesh Kumar and Greff, Klaus and Schmidhuber, Jürgen. Highway Networks. arXiv (2015).
- Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult,". IEEE Transactions on Neural Networks. (1994)