S&P-500 Index Forecast using Neural Networks

Piyush Gautam
Department of Computer Engineering
New York University
New York, USA
pg2374@nyu.edu

Prof. Chinmay Hegde
Department of Electrical and Computer Engineering
New York University
New York, USA
chinmay.h@nyu.edu

***Abstract*— Forecasting algorithms for stock market prediction are an area of a hot topic because most of them only train using data gathered on a single stock. Machine Learning for Finance provides tremendous potential to achieve better prediction by incorporating time series analysis, due to the dynamic nature of the stock market and its prices. In this study, the main focus was to improve the stock prediction models for the cumulative stock data of 500 companies on SP500 index-related stocks, similar to a professional stock exchange trader. The concepts of Neural Networks, Recurrent Neural Networks(RNN), and Convolutional Neural Networks(CNN) were applied to the SP500 dataset for best predictions of the index price. The technique of Long Short-Term Memory was implemented for improved forecasts over ten years from 2010 to 2020.**

Index Terms—Machine Learning; Neural Network; Recurrent Neural Network; Convolution Neural Networks; Long Short-Term Memory.

Github : https://github.com/pg2374/SP500

I. INTRODUCTION

With the emergence of Machine Learning, we have witnessed rapid changes in automation, and various machine learning algorithms providing automated results to enhance day to day activity of humans. The stock prediction component of an Algo-trading architecture gathers data from a variety of sources, including market trading and news. The objective of the prediction is to provide the strategy component with feed based on the subsequent pricing values. The strategy component is in charge of analyzing data on the existing trader position, risk parameters, losses, and price projection to determine whether to buy or sell specific financial equity.

The application of machine learning to finance is a new field. An advanced form of machine learning (ML) based on algorithms for artificial neural networks (NN) is known as deep learning(DL)[1]. DL has advantages over conventional machine learning methods like support vector machines (SVM) and k-nearest neighbors (kNN) due to its strong capacity for generalization, unsupervised feature learning, and robust training power for big data. DL is currently used extensively in audio-visual recognition, computer vision, image processing, and classification and prediction tasks[2]. Despite the fact that it was developed in the field of computer science, DL has found applications in a wide range of fields, including medicine, neuroscience, physics and astronomy, finance and banking (F&B), and operations management[3]. Deep Learning (DL) architectures like Recurrent Neural Networks and Convolutional Neural Networks were made accessible to common applications thanks to affordable parallel computing. The use of DL in the finance and banking (F&B) industry is poorly covered in the existing literature. A novel approach known as DeepLearning (DL) has emerged that outperforms conventional machine learning methods in terms of accuracy. Because DL can independently process raw data and learn the high-level features, it makes a strong case for its adaptability in networks with limited resources..

The majority of non-recursive neural networks, such as MLP, RBFNN, and so on, have larger architectures than recurrent neural networks (RNNs). Additionally, because of their feedback properties, they are dynamic and more effective at accurately modeling nonlinear systems—essential for nonlinear prediction and time series forecasting. RNNs have been used to accurately model many Autoregressive Moving Average (ARMA) processes for the purpose of identifying nonlinear dynamic systems.

A RNN approach typically relies on learning from sequences, where the sequence is nothing more than a list of pairs (x_t, y_t), where x_t, y_t indicates an input and the corresponding output at a given time step t. For various types of problems, we can either select a list of desired outputs for each individual x_t or have a constant output value for the entire sequence, y_t=t. We consider some hidden state at each time step to model sequence. The latter makes it possible for the RNN to comprehend the current state of a sequence, retain the context, and process it in order to produce values in the future. A new hidden state, which we'll call h_t, is added to each new input x_t in accordance with h_(t-1). At each time step, the RNN is just a feed-forward neural network with one hidden layer and an input x_t and an output y_t in the context of so-called regular fully-connected neural networks. Considering that we are presently thinking about several sources of inputs, x_t and h_(t-1), there are three weight matrices, to be specific, W_(hx) for weights from input to hidden layer, W_(hh) from hidden to hidden, and W_(yh) for the output. The back propagation through time (BPTT) algorithm typically represents the RNN training procedure. The former is determined similarly as the fundamental back propagation one. We require all of the error metric's partial derivatives with respect to the weights because the weight update procedure is typically carried out by an iterative numerical optimization algorithm that makes use of partial derivatives of nth order, such as first order in stochastic gradient descent.[4] A negative log probability can be used to represent the loss function, namely:

$$\frac{\partial \mathcal{L}^{(T)}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{(T)}}{\partial W}\Big|_{(t)}$$

The rest of the paper is organized as follows. Related works are introduced in section 2. In section 3, we give a brief introduction to our deep learning model, Long Short-Term Memory and CNN Long Short-Term Memory. The performance of the model is analyzed in section 4. Finally, we draw conclusions and propose some future work in section 5.

II. RELATED WORK

The motivation to predict market stock price using LSTM is inspired by [5]. [6] summarizes the design experience of using LSTM RNN to predict stock market. [7] provides an architecture reference to build time series forecasting model. The possibility of utilizing reinforcement learning in the equity market has been the subject of some recent research ([4-9]), but there are still a number of obstacles to overcome: 1) The data on real-world trading is limited. 2) There are numerous ways to define the reward of reinforcement learning for trading strategy. The robust learning rule, the final optimal target, and the dataset's limit must all be taken into account.

Some potential patterns may not be captured because training is only based on historical data. Due to a lack of historical data, it may then result in data sparsity issues. In case of Financial crisis, inflation, news based sentiments, the designed model or architecture can prove to be very unreliable and extreme caution should be taken on acting on the signals generated..

III. IMPLEMENTATION

A. Deep Learning Approach

In classical machine learning, important features of an input are manually designed and the system automatically learns to map the features to an output. In deep learning, there are multiple levels of features. These features are automatically discovered and they are composed together in various levels to produce outputs. Each level represents abstract features that are discovered from the features presented in the previous level.

```
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_121 (LSTM)              (None, 64)                16896

 dense_72 (Dense)             (None, 1)                 65

=================================================================
Total params: 16,961
Trainable params: 16,961
Non-trainable params: 0
_________________________________________________________________
```

Figure 1 : LSTM parameters

In our experiment, we constructed 3 different deep neural network implementations. The first one is the basic Long Short-Term Memory implementation. The second one is the Stacked layers of Long Short-Term Memory implementation and the third one is the Convolution based Long Short-Term Memory implementation. The input to these network is the index price for a period of ten years, and the output is the predicted price or the future forecast of the index price. Our model initiation parameters are setup with 20 for the batch size and 100 for the epoch. The net trainable and non trainable parameters are shown in the figure 1,2 and 3.

```
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_115 (LSTM)              (None, 60, 64)            16896

 lstm_116 (LSTM)              (None, 60, 64)            33024

 lstm_117 (LSTM)              (None, 60, 64)            33024

 lstm_118 (LSTM)              (None, 64)                33024

 dense_70 (Dense)             (None, 1)                 65

=================================================================
Total params: 116,033
Trainable params: 116,033
Non-trainable params: 0
_________________________________________________________________
```

Figure 2 : Stacked LSTM parameters

```
 Layer (type)                 Output Shape              Param #
=================================================================
 time_distributed_111 (TimeD  (None, None, 1, 64)       3904
 istributed)

 time_distributed_112 (TimeD  (None, None, 1, 64)       0
 istributed)

 time_distributed_113 (TimeD  (None, None, 64)          0
 istributed)

 lstm_119 (LSTM)              (None, None, 64)          33024

 lstm_120 (LSTM)              (None, 64)                33024

 dense_71 (Dense)             (None, 1)                 65

=================================================================
Total params: 70,017
Trainable params: 70,017
Non-trainable params: 0
_________________________________________________________________
```

Figure 3 : CNN LSTM parameters

B. Long Short Term Memory(LSTM)

When it comes to modeling short sequences, basic RNNs outperform many other architectures. Nevertheless, they demonstrate a rather extensive set of issues. This is, for instance, the situation with vanishing gradients, in which the gradient signal becomes so small that learning becomes extremely slow for data with long-term dependencies. On the other hand, if the weight matrix's values are too high, the gradient signal may become so large that the learning scheme diverges. The latter is more commonly known as exploding gradients. In order to overcome problems with long sequences an interesting approach for long-short term memory has been developed by Schmidhuber in [9].

In contrast to RNNs, the single time step cell in LSTM has a structure that is more complex than just hidden state, input, and output. Inside this structure, frequently called memory blocks, there are three adaptive and multiplicative gating units, for example the input gate, the forget gate and the output gate. With corresponding weights, the roles of the input and output gates are identical to those of the RNN's input and output cases. The new case, in particular the forget gate, plays the part of learning out how to recall or to neglect its past state. The latter feature enables the detection of more intricate temporal patterns.

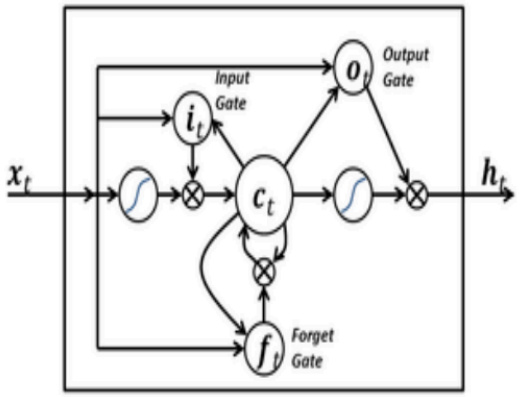


Figure 4 : LSTM Network

C. CNN - Long Short Term Memory(CNN-LSTM)

Convolutional Neural Network (CNN) layers are used to extract features from input data in the CNN LSTM architecture, and LSTMs are used to help predict sequences. CNN LSTMs were produced for visual time series expectation problems and the use of creating textual descriptions from sequence of pictures (for example video recordings).

IV. PERFORMANCE/RESULTS

In order to test the performance of different algorithms proposed in this study, namely, LSTM, Stacked LSTM, CNN LSTM and Stacked CNN-LSTM we trained the model for different parameters. Also, the price was forecasted on the basis of these models taking into account various technical signals like moving average, exponential weighted moving average, volume weighted average price(VWAP), etc, which could be useful for buy or sell signals. The rolling moving average and exponential weighted moving average were calculated for a window of 20, 50, 100, and 200 time units. The results show that the moving average follows the index price trend and depicts the price movement of the underlying index/ stock. Also, the VWAP was calculated taking into account the volume and price together and results ere calculated for 1 positive and 1 negative standard deviation. Figure shows the performance results for the different strategies tested in the experiment. The loss for the train and validation set are calculated and we can see that the loss values gradually decrease. The size of the dataset is small, hence, training for more number of epochs, or increasing the model size (which will increase the number of parameters) can lead to overfitting and we see a diminished score for accuracy as we increase the stacked hidden layers in the model. To address this issue, we kept the model size small for better results, as we can see from the Figure that best Mean Absolute Percentage Error value was achieved for the basic LSTM model.

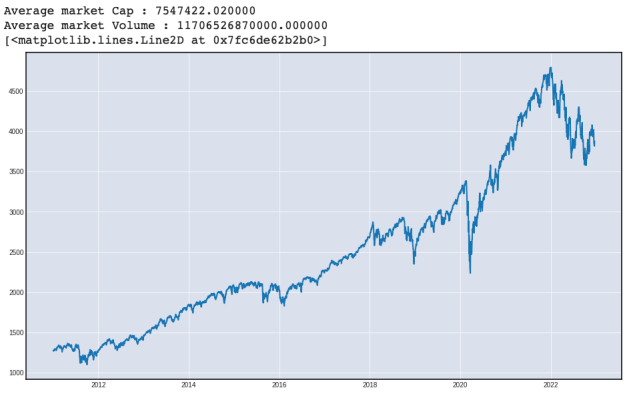


Figure 5: SP500 Index Price

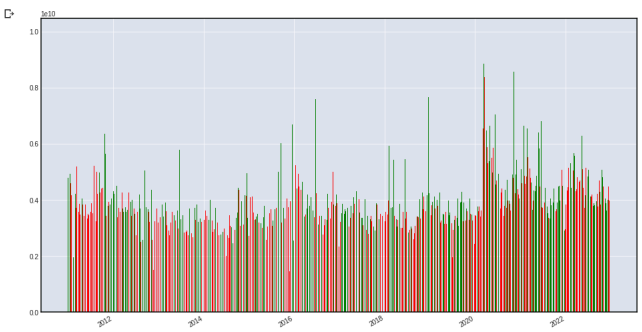


Figure 6: Volume Profile

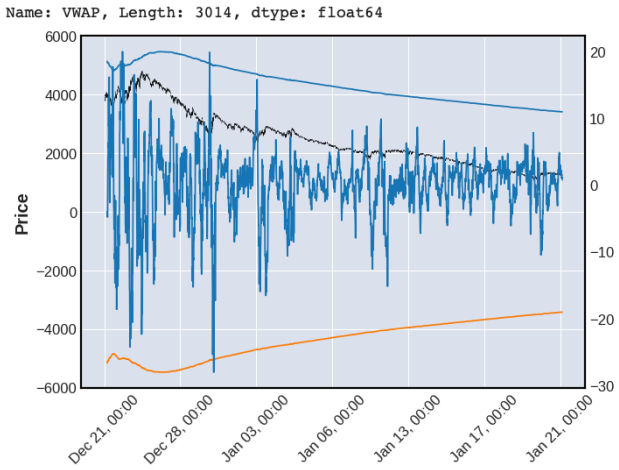
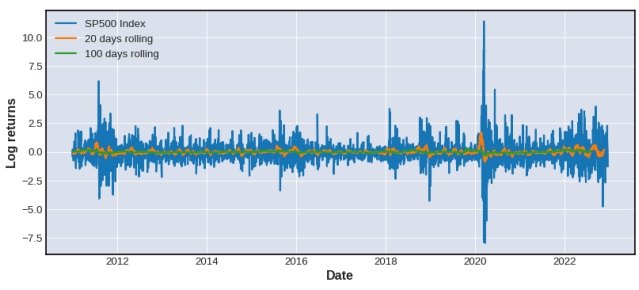


Figure 7 : VWAP (1 STDEV)



We can also see that the volume weighted average price fluctuates between one standard deviation and can be a potential signal for target price when the VWAP is at mean or a reversal trade signal when it is stretched too far from the mean.

Figure 8 : Log returns plot

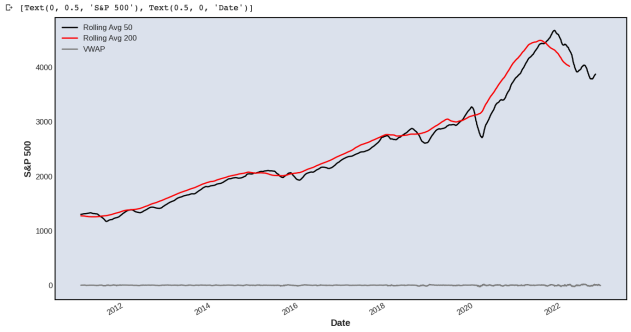


Figure 9 : 50-200 Simple Moving Average

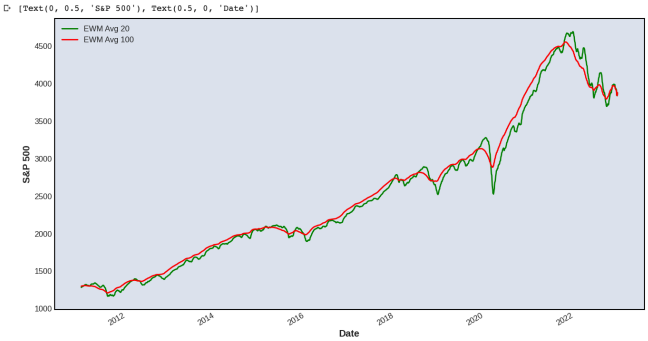


Figure 10 : 20-100 Exponential Moving Average

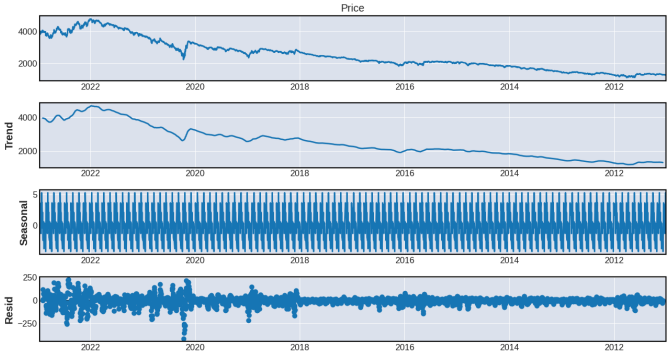


Figure 11 : Statistical values

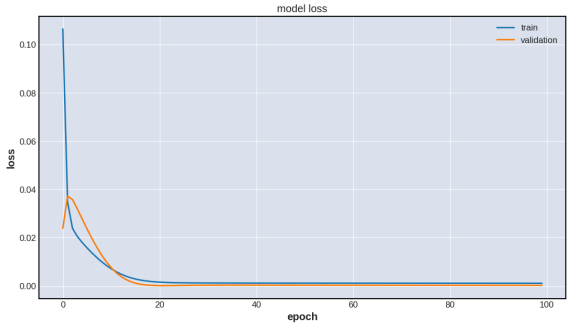


Figure 12 : LSTM Loss

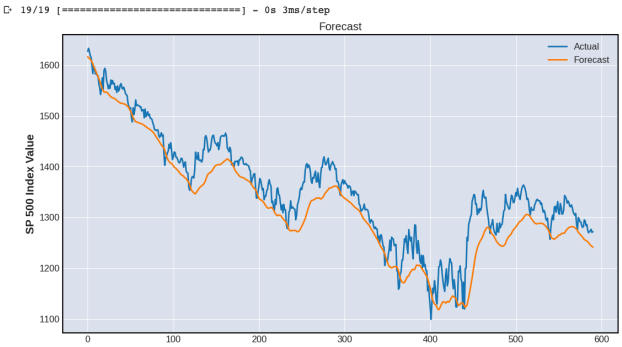


Figure 13 : LSTM Forecast

```
[133]from sklearn.metrics import mean_absolute_error, mean_squared_error
    print('Long Short-Term Memory model Mean Square Error :', mean_squared_error(test, forecast))
    print('Long Short-Term Memory model Mean Absolute Error :', mean_absolute_error(test, forecast))
    print('Long Short-Term Memory model Mean Absolute Percentage Error :', mean_absolute_percentage_error(test, forecast))

Long Short-Term Memory model Mean Square Error : 1982.2838967954685
Long Short-Term Memory model Mean Absolute Error : 36.4408887875423
Long Short-Term Memory model Mean Absolute Percentage Error : 2.7189567497765172
```

Figure 14 : LSTM Results

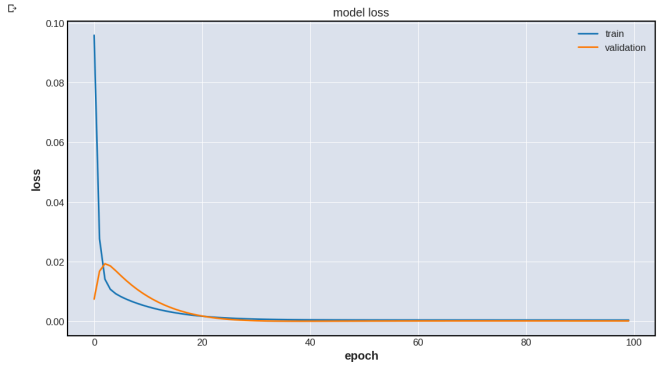


Figure 15 : Stacked LSTM Loss

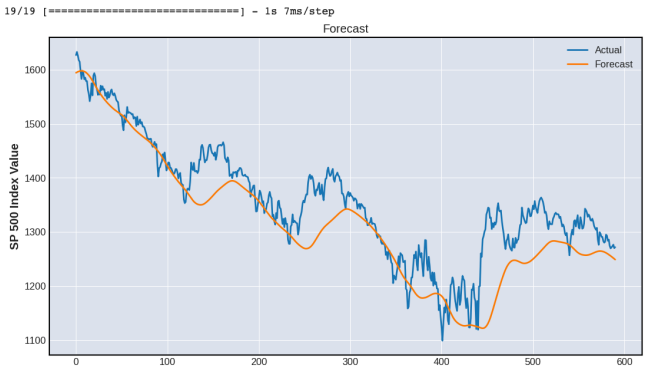


Figure 16 : Stacked LSTM Forecast

```
Long Short-Term Memory model Mean Square Error : 3391.124919071473
Long Short-Term Memory model Mean Absolute Error : 43.312314031765546
Long Short-Term Memory model Mean Absolute Percentage Error : 3.233124420228348
```

Figure 17 : Stacked LSTM Results

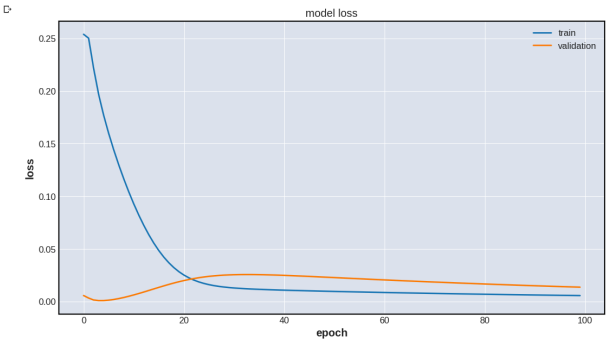


Figure 18 : CNN LSTM Loss

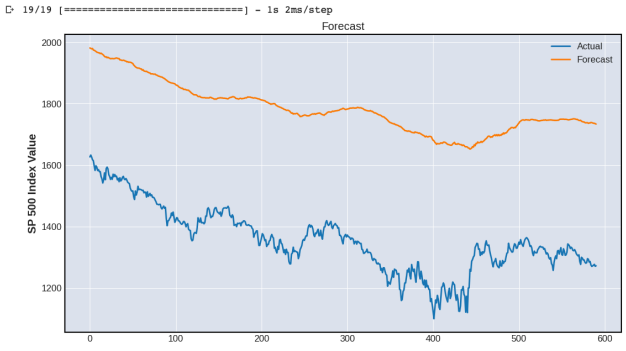


Figure 19 : CNN LSTM Forecast

```
CNN Long Short-Term Memory model Mean Square Error : 185402.47606903355
CNN Long Short-Term Memory model Mean Absolute Error : 428.14277112415397
CNN Long Short-Term Memory model Mean Absolute Percentage Error : 31.98198741918297
```

Figure 20 : CNN LSTM RESULTS

CONCLUSION/FUTURE WORK

In this paper, we used some of the most promising RRN architectures, including basic RNNs, LSTMs, and CNNs, to forecast stock market price movement. We have compared weekly-trained results for SP500 index prices over the past ten years to demonstrate that the LSTMs method is capable of providing an adequate level accuracy of about 0.48 when varying the batch size for training, analogous to 0.52 of the market standard. This indicates that it can be utilized effectively in real life, as success rate of approximately 50% in stock predictions can turn your trades into profits when minimizing the loss depending upon the signals generated. We also show that RNNs need to be trained for a large number of epochs, choosing final weights carefully, and stopping early to avoid overfitting to the dataset.

Additionally, we have examined the hidden dynamics of RNNs. The latter enables us to demonstrate that Neural Networks are not black box learning models with an inner structure that cannot be understood. As a matter of fact, Neural Networks are capable of pattern learning. In particular, they are able to spot fluctuations in time series that occur in the short term. For further time series analysis, these activations can be used as indicators.

In the future, this model could be optimized with more efficient neural classifier and use of other features to increase the validation accuracy. We will also try to implement our approach in a distributed manner to predict results depending on different strategies apart from price and volume, such as a combination of these features to predict the Price to Earnings, Dividends, Enterprise Value, or the EV multiples for improved forecasts.

REFERENCES

1. Zexin Hu, Yiqi Zhao, Matloob Khushi “A Survey of Forex and Stock Price Prediction Using Deep Learning”. arXiv:2103.09750v1
2. Ajit Kumar Rout, P. K. Dash, Rajashree Dash, Ranjeeta Bisoi” Forecasting financial time series using a low complexity recurrent neural network and evolutionary learning approach”. https://doi.org/10.1016/j.jksuci.2015.06.002.
3. Huang, J., Chai, J. & Cho, S. Deep learning in finance and banking: A literature review and classification. *Front. Bus. Res. China* **14**, 13 (2020). https://doi.org/10.1186/s11782-020-00082-6.
4. Di Persio, Luca, and Oleksandr Honchar. "Recurrent neural networks approach to the financial forecast of Google assets." *International journal of Mathematics and Computers in simulation* 11 (2017): 7-13.
5. Stock Market Analysis + Prediction using LSTM. https://www.kaggle.com/faressayah/stock-market-analysisprediction-using-lstm
6. Adil Moghar, Mhamed Hamiche. Stock Market Prediction Using LSTM Recurrent Neural Network. (2020) https://www.sciencedirect.com/science/article/pii/S1877050920304865
7. Time series forecasting. (2020) https://www.tensorflow.org/tutorials/structured_data/time_series
8. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," in Neural Computation, vol. 9, no. 8, pp. 1735-1780, 15 Nov. 1997, doi: 10.1162/neco.1997.9.8.1735.
9. Jürgen Schmidhuber, “Deep learning in neural networks: An overview, Neural Networks, Volume 61, 2015, Pages 85-117, ISSN 0893-6080”. https://doi.org/10.1016/j.neunet.2014.09.003.
10. H. Strobelt, S. Gehrmann, B. Huber et al, Visual Analysis of Hidden State Dynamics in Recurrent Neural Networks, 2016
11. X. Ding, Y. Zhang et al, Deep Learning for Event-Driven Stock Prediction, Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI), 2015