Applications of Neural Networks in Predicting Significant Stock Price Changes

Yaolun Yin

December, 2022

Supervisor: Dr.Adam Kolkiewicz

A project

In partial fulfillment of the

requirement for

Master of Quantitative Finance

Abstract

Most of the current literature on stock prediction focuses on forecasting either the actual price or the direction of price changes. This paper considers a slightly different problem – predicting significant price changes – because the stock prices may have some small fluctuations (i.e., noise). The prediction process also differs from traditional approaches, which use various financial indicators to forecast stock prices. The proliferation of machine learning has allowed researchers to use more sophisticated models such as neural networks or random forests. This paper will present the performance of Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), Long short-term Memory (LSTM), and Random Forest (RF) in predicting significant stock price changes. In particular, it is found that LSTM accurately forecasts positive changes for Cisco Systems Co.

Acknowledgement

I would like to express my deepest gratitude to Dr.Adam Kolkiewicz and Dr.Tony Wirjanto for reviewing my master's research paper.

Contents

List of Figures

List of Tables

1 Introduction

Stock prediction is a tempting topic that attracts numerous talents and scholars. Some researchers believe that successful prediction is not feasible because stock price contains all publicly accessible information ([6]). The market is efficient enough to adjust the stock price so that no one can profit from any prediction. Despite discouraging arguments, researchers have been persistently attempting to design new prediction models. The proliferation of machine learning techniques has shed light on stock prediction in recent years. Analysts have deployed machine learning models to forecast actual stock prices or buy signals. In this article, we introduce a method of predicting *significant* changes in asset price. We use previous p days to determine if the next daily return is significant.

Economic data is enormous and complex, and the inner connections between different data may not be easily observed. Machine learning models can capture complicated nonlinear relationships within data and determine what relationships help predict significant price changes. Numerous examples have shown the success of machine learning models. One of the most convincing examples is the high returns of the Medallion Fund(Olivier, 2012 [11]). Meanwhile, we should recognize the limitation of machine learning as most models are black boxes.

There are two basic ways of forecasting stock prices: numerical prices or directions of a price change (i.e., trends). The former usually requires a regression model (i.e., regressor), whereas the latter needs a classification model (i.e., classifier). A regressor produces continuous values, while a classifier generates discrete values. Predicting numerical prices is challenging since most quantitative models cannot beat the performance of a simple random

walk. Forecasting stock trends is more viable than prices, as it requires less accuracy. In addition, we only need stock direction changes to determine the buy and sell time. However, predicting the actual price may not be necessary.

This paper extends the idea of predicting direction changes to significant changes in price([7]). The former's accuracy is low because the stock movement is highly random. Minor price changes may not truly reflect the market's future movement because they are noises in the data. We do not expect the model to capture changes driven by noise. For this reason, we decided to focus on substantial price changes that are rare and driven by different fundamentals. Although black swans are difficult to capture, there is a high probability of identifying situations where a stock is overbought or oversold and acting accordingly. Our machine learning prediction models include multilayer perceptron (MLP), convolutional neural networks (CNN), long short-term memory (LSTM) networks, and random forests (RF). The first three models belong to neutral networks. CNN and LSTM apply additional transformations to data, whereas MLP is a basic feedforward network. We compare the performance of neural network models and pick the optimal one. Also, we choose a non-neural net algorithm, RF, as a control group for our study. RF is a simple and robust model that often serves as a benchmark.

Neural networks have gained popularity through excellent performance in image or voice recognition([4][14]). They can build artificial features that allow them to improve accuracy during training. We need to choose some special neural network architectures like LSTM, which can handle sequential time series (e.g., stock prices). Traditional machine learning algorithms such as RF, SVM, or MLP, do not include the characteristics of sequential data. Hence, LSTM has inherent advantages over other methods for stock price prediction.

We select the daily stock prices of four American corporations over 11 years to build prediction models. We study the performance of models in predicting substantial positive or negative daily returns. The experiments demonstrate that machine learning models can accurately predict daily returns. To measure the accuracy of models, the Area Under the Curve (AUC) measure is used in this report. The AUC is a measure of the ability of a classifier to distinguish between classes. For example, section A shows that Area Under the Curve (AUC) achieves 0.7 for Cisco systems. An AUC of 0.7 means that if we take two data points from two different classes, there is a 70% chance that the model will correctly separate them. A model with AUC of 0.7 is considered to be relatively accurate.

In following sections, we evaluate contemporary neural network architectures by investigating the precision of MLP, CNN, and LSTM in price forecasting. We contrast the performance of neural networks with benchmark models, such as RF.

The paper is divided as follows. Section 2 describes the approaches underlying models to help readers better understand their mechanisms. Section 3 demonstrates the numerical experiments and results. The conclusion is in the Section 4.

2 Neural Networks

Neural networks consist of layers of nodes analogous to neurons in human brains. The brain relates to the best possible pattern and concludes the result.([10]) The same logic applies to neural networks. In this section, we will introduce the feedforward architecture of neural networks and discuss three members of them – Multilayer Perceptron (MLP), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) – which are widely used

in the industry.

2.1 Feedforward Architecture

Feedforward neural networks are a type of supervised machine learning model that uses layers of abstraction to produce high-dimensional non-linear predictors. The deep learning paradigm for data analysis is entirely different from the traditional modeling and testing framework. Conventional standards, such as R^2, t-values, p-values, and the concept of *statistical significance* have been substituted with out-of-sample forecasting and understanding the *bias-variance tradeoff*, the tradeoff between a more complex model versus overfitting. The primary tool for selecting variables is *dropout*, which can be used to prevent overfitting([1]).

Mathematically, the feedforward neural network takes the form of a composition of simple functions:

$$\hat{Y}(X) := F_{W,b}(X) = f^{(L)}_{W^{(L)},b^{(L)}} \circ ... \circ f^{(1)}_{W^{(1)},b^{(1)}}(X)$$

where

1. $F_{W,b}(X)$ is a deep neural network with L layers.

2. $W = (W^{(1)}, ..., W^{(L)})$ are weight matrices

3. $b = (b^{(1)}, ..., b^{(L)})$ are bias vectors

Any weight matrix $W^{(l)} \in \mathbb{R}^{m \times n}$ can be expressed as n column m-vectors $W^{(l)} = [\mathbf{w}^{(l)}_{,1}, ..., \mathbf{w}^{(l)}_{,n}]$. Each weight is denoted by $w^{(l)}_{ij} := [W^{(l)}]_{ij}$.

In practice, *semi-affine* functions are often adopted to form the above parametrized map under certain restrictions. Heaton et al.([3]) gave the following definition for semi-affine functions.

Definition 2.1 (Semi-Affine Functions)**.** *Let $\sigma : \mathbb{R} \to B \subset \mathbb{R}$ be a continuous and monotonically increasing function whose image is a bounded subset of $\mathbb{R}$. A function $f^{(l)}_{W^{(l)},b^{(l)}} : \mathbb{R}^n \to \mathbb{R}^m$ given by*

$$f(v) = W^{(l)}\sigma^{(l-1)}(v) + b^{(l)}, \text{where } W^{(l)} \in \mathbb{R}^{m\times n}, b^{(l)} \in \mathbb{R}^m$$

is a semi-affine function in v, e.g. $f(v) = wtanh(v)+b$, and $\sigma(\cdot)$ are the activation functions of the output from the previous layer.

$F_{W,b}$ is just a linear regression when all activation functions are linear, no matter how many hidden layers L exist. For a linear network, we can find an equivalent network without hidden layers, which follows from fact that the composition of many linear functions is still linear. For instance, consider the map with one hidden layer and $\sigma^{(1)}$ is an identity function:

$$\hat{Y}(X) = W^{(2)}\left(W^{(1)}X + b^{(1)}\right) + b^{(2)} = W^{(2)}W^{(1)}X + W^{(2)}b^{(1)} + b^{(2)} = \tilde{W}X + \tilde{b}$$

In simple words, the primary purpose of the activation function is to add non-linearity to the model. Concretely, the activation function should introduce interaction terms between the input. We will present a particular situation in which the activation function gives interaction terms X_iX_j in a network with one hidden layer. Consider the partial derivative,

$$\partial_{X_j}\hat{Y} = \sum_i \mathbf{w}^{(2)}_{,i}\sigma'(I^{(1)}_i)w^{(1)}_{ij}$$

where $\mathbf{w}^{(2)}_{,i}$ is the i-th column vector of $W^{(2)}$, $I^{(l)}(X) = W^{(l)}X + b^{(l)}$. The second order partial

derivative is

$$\partial_{X_j,X_k}\hat{Y} = -2\sum_i \mathbf{w}_{,i}^{(2)}\sigma(I_i^{(1)})\sigma'(I_i^{(1)})w_{ij}^{(1)}w_{ik}^{(1)}$$

Note that the second derivative is generally not equal to 0 unless $\sigma(\cdot)$ is an identity function.

2.2 Multilayer Perceptron

Our classification problem is to predict a binary class representing significant price changes in the next day given some previous prices. In the following sections, the previous price will be referred to as X matrix, and the target values will be called a y vector. To find the connection between today's and previous prices, we can adopt a Multilayer perceptron (MLP). MLP is an elementary type of feedforward neural network that contains an input layer, some hidden layers, and an output layer Figure 1. Each layer is built with several nodes interconnected by weights. In the training stage, MLP adjusts these weights to improve classification accuracy. Also, the network can either be a forward-feeding (forward) or backward-feeding (backward) pass. The former means the data flow is from the input to the output layer. The latter refers to obtaining partial derivative functions of the cost function with respect to the weights and using derivatives to gauge the weights. Despite its simple structure, MLP is still a robust and effective model for prediction ([8]). The following paragraphs detail the difference between a forward and backward pass in the training stage.

A forward pass starts with the input layer and calculates node values of adjacent layers at the training stage where the model tries to find the relationship of the X matrix and the y vector. Notice that the y vector exists in the training stage because we typically use past data to train the model. The number of nodes is equal to the number of input features.

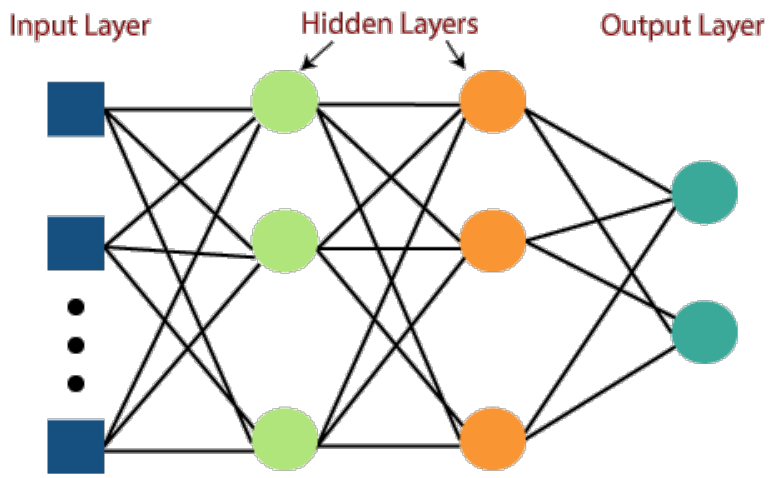


Figure 1: Multilayer Perceptron. (Source)

After the input is passed to the first hidden layer, it transforms the weighted sum of input features and a bias term using a nonlinear activation function, such as *sigmoid*, *tanh*, or *Relu*. The process stops when the output layer is filled. In a sense, MLP is equivalent to combining a series of affine transformations with several nonlinearities. The cost function, which is defined by the mutual information between predicted and actual values of the target variable, evaluates the performance of our machine learning model.

The backward pass uses a different approach in the training stage. The network weights are tuned based on the corresponding partial derivatives of the cost function, which is done by a single gradient descent step in minimizing the cost function. Partial derivatives are calculated from the output to input layers, suggesting a 'backward' process. The chain rule simplifies the derivative calculation in a great deal. For this reason, the chain rule plays a crucial part in generating partial derivatives for the weights of successive layers, making MLP an excellent candidate for stock prediction.

2.3 Convolutional Neural Network

Convolution has gained popularity in the field of mathematics and computer science. It refers to an integral that represents how much one function overlaps the other. The invention of convolutional neural networks (CNN) is inspired by the application of convolution in image processing. CNN architecture is nearly identical to MLP except for one significant difference: CNN uses convolutional layers. Convolution is a process where small matrices of numbers are transformed based on the values from filters. For example, the following formula describes how the transform happens.

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

where the input matrix is f and the convolution function is h. A convolutional layer is constructed by a sliding window over the input array and taking the dot product with the corresponding array. Hence, we can adjust the sliding window size to allow CNN to exploit any existing structure within the data.

Another special feature of CNN is pooling layers, which refines the signals fed into the network. Thus, a typical CNN has a few convolutional and pooling layers followed by regular dense layers. CNN combines the advantages of pooling layers and MLP. Popular CNN architectures include AlexNet, ResNet, and Inception, which are highly successful in image processing and making machine learning well-known([7]). CNN is a terrific candidate to explore any hidden patterns of time series.

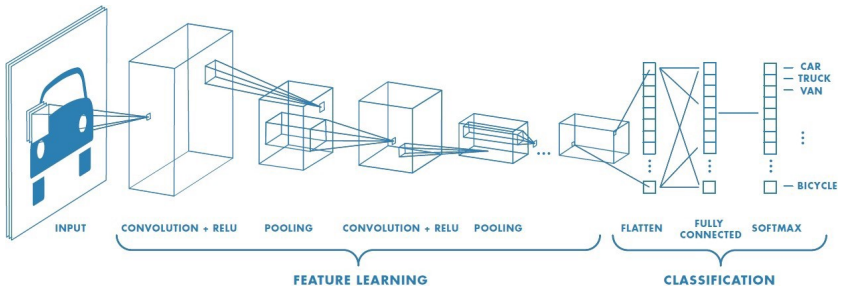


Figure 2: Convolutional Neural Network. (Source)

2.4 Long Short-Term Memory

Long short-term memory (LSTM) belongs to the recurrent neural network family. The applications of LSTM are successful in natural language processing([9]). To explore the mechanism of LSTM, we need to understand the Recurrent neural networks (RNN) because LSTM is an extension of RNN. RNN focuses on processing sequential data, which consists of multiple steps. For example, the stock price is a type of sequential data. Concretely, the output of RNN is calculated using the current input and the previously hidden state obtained at the last step. Thus, the previous input is preserved and passed to the network to calculate the current output.

A common problem of RNN is a vanishing gradient, where the updated gradient quickly approaches 0 after some iterations in the training process. A slight gradient is problematic because the weight will not be updated effectively. To solve this problem, LSTM inserts a long short-term memory unit (LSTM unit) into a regular RNN. An LSTM unit has three gates – input gate, forget gate, and output gate – that control the memorizing process of the network. Overall, LSTM is empirically suitable for processing sequential data; hence, it is a great candidate for our stock prediction problem.

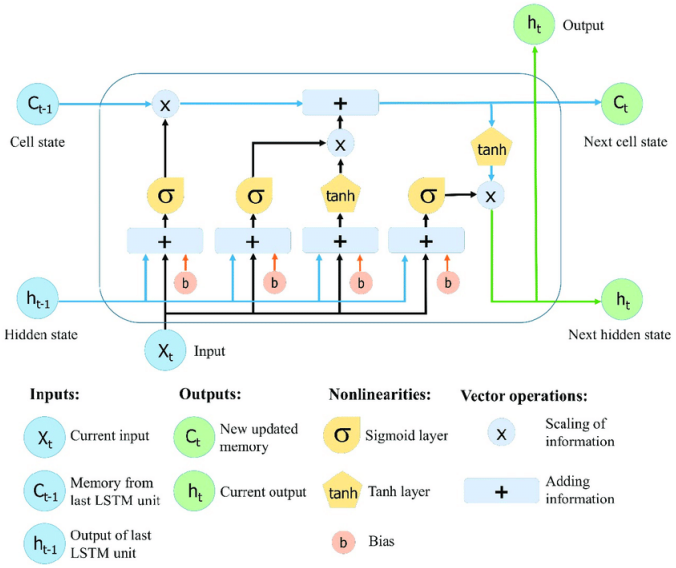


Figure 3: Long Short-Term Memory. (Source)

2.5 Random Forest

Random forest (RF) is a typical machine learning model that aggregates the output of many decision trees through majority voting. Concretely, RF consists of N decision trees, where N is an input from users. The RF algorithm recursively splits different values of input features to generate decision trees. The splitting point will be determined by the corresponding information gain. After the construction process, each tree will generate an output after the training process. Then, the majority decision will be the final output of RF. In this way, RF reduces overfitting, which is a classic problem for decision trees. Also, we can easily interpret the mechanism and the result of RF, which is an advantage compared to other machine learning algorithms that may be criticized as a "black box." Additionally, an individual decision tree only selects a subset of all features at one split. It is a strategy to reduce the correlation between trees and output variance.

3 Numerical Experiments

This section demonstrates the results of our experiments designed to test the performance of machine learning algorithms in forecasting significant changes in stock price. As we demonstrate, the machine learning algorithms, especially neural networks, have proven to be effective in predicting large changes in stock price.

3.1 Methodology

Our experiments examine three primary neural networks: MLP, CNN, and LSTM. The implementation of these models comes from the TensorFlow library developed by Google. We use the same general architecture for all networks to make them comparable to others. Table 1 presents the architecture for all models. Each model contains an input layer, two hidden layers, and an output layer. The ReLu activation function, $f(x) = max(0, x)$, is used in every layer except the output layer. One main advantage of ReLu is that it does not simultaneously activate all neurons (i.e., $f(x) = 0$), making ReLu more computationally efficient than other activation functions. Also, we adopted the dropout method, which randomly removes nodes during the training stage. The purpose of the dropout method is to simulate different architectures for a single model. A dropout rate of 0.2 is applied to certain layers in CNN and LSTM models. Aside from neural networks, the RF model represents the traditional machine learning models and the benchmark. We import the RF model with 100 decision trees from the scikit-learn library.

In our experiments, we have used data from four American publicly traded companies: Coca-cola, Cisco Systems, Nike, and Goldman Sachs. The data can be obtained from the

Network	Hidden Layer 1	Hidden Layer 2
MLP	Dense 64	Dense 32
CNN	Conv1D 64, length = lookback window, dropout rate = 0.2	Dense 32, dropout rate = 0.2
LSTM	LSTM 64 with return sequence, dropout rate = 0.2	LSTM 32, dropout rate = 0.2

Table 1: Neural network architectures

Yahoo Finance website using the 'yfinance' package in Python. We use adjusted daily stock prices from 2009 to 2019 inclusive. The following formula generates the daily return:

$$r_t = ln(\frac{p_t}{p_{t-1}})$$

where r_t and p_t represent the returns and price for day t respectively. The returns data is split to training (accounting for 75% of the original dataset) and testing datasets (accounting for 25% of the original dataset) to prevent the overfitting problem. The input feature vectors consist of prior returns over the previous p days, and the output is the current value:

$$x_k = (r_{k-p}, r_{k-p+1}, ..., r_{k-1}), y_k = r_k$$

where r_k is the current return and r_{k-i} is the past i-th day's return.

The candidate values for p are 7, 14, 30, and 60 days. We want to study the dependence between the current return and the past week, two weeks, month, or two months. A *significant* daily return is defined as a value that exceeds the predefined threshold. The threshold is a fraction of the standard deviation of daily returns over the training period. For instance, if the fraction value is 1.5, then any return greater than 1.5σ is considered a positively significant value; any return less than -1.5 σ is a negatively significant value, where σ is the standard deviation of daily returns in the training set. In addition, we adjust

the values of fractions and observe the effects of changing thresholds on the performance of models.

The significant returns only account for a small percentage of all returns. For example, the training set of Coca-cola with $p = 7$ and fraction $= 1$ has 271 significant returns out of 2070 observations. We want the model to capture the significant returns, but they only takes 13% of all observations. The significant and non-significant returns can be referred as the minor and major class respectively. If we feed a new X matrix to the model, it can achieve a high accuracy of 87% by only outputting major class regardless of the input. The accuracy cannot measure the ability of our model to capture minor classes. Thus, we need to choose a different metric that reflects the actual performance of the classifiers. Area under the ROC curve (AUC) is a metric to combat this issue. The ROC curve is derived by plotting a model's true positive (TR) rate against the false positive rate (FR) at different threshold levels. A high AUC of 0.7 or above shows that the model can accurately capture minor classes.

3.2 Overfitting Prevention

Overfitting happens when the statistical model fits precisely to a particular dataset. Thus, an overfitted model is undesirable since it produces poor predictions on another dataset. In Section 3.1, we mentioned the 'dropout' method, which is a crucial way to prevent neural networks from overfitting ([5]). Dropout is a computationally efficient variance-reduction technique that averages the prediction of many model configurations. The layer input space $Z = (Z_1, ..., Z_n)$ requires dimension reduction methods designed to avoid overfitting in the training process if n becomes large. Dropout works by arbitrarily removing input nodes with

a predetermined probability θ. For instance, if there are 1000 variables, then a model with $\theta = 0.2$ will only search for 200 variables. Mathematically, the dropout architecture with a stochastic search is presented in the following equations

$$d_i^{(l)} \sim Ber(\theta),$$

$$\tilde{Z}^{(l)} = d^{(l)} \circ Z^{(l)}, 1 \leq l < L$$

$$Z^{(l)} = \sigma^{(l)}(W^{(l)}\tilde{Z}^{(l-1)} + b^{(l)})$$

The dropout architecture replaces the layer input Z by $d \circ Z$, where $\circ$ denotes the element-wise product and d is a vector of independent Bernoulli variables. By Heaton et al., the overall objective function closely relates to ridge regression with a g-prior ([3]).

3.3 Results

Appendix A presents the performance of several machine learning models considered in this study. Each company has a table of eight graphs, where the left (right) column shows performance in forecasting significant positive (negative) changes in daily return. The result for Cisco Systems is impressive Figure 4. LSTM achieves large values of AUC on average for predicting positive changes. In particular, the highest AUC is attained by LSTM with a 7-day lookback period. Let p denote the number of days in a lookback period. MLP has a high result of around 0.8 when $p = 7$, which is significantly higher than its peers with different values of p. We also notice that $p = 7$ is the best parameter for Cisco's stock. The worst AUC happens for $p = 60$. For negative changes, a 14-day lookback window seems to be the best candidate because the AUC curve stays at the top half for LSTM, CNN, and RF. An interesting observation is that LSTM is more accurate with a larger threshold. Overall,

results for $p = 7, 14$ are excellent, while for $p = 60$ are poor. For model selection, LSTM is the best model for both positive and negative change prediction.

The experiments on the Coca-Cola Co yield mixed results shown in Figure 5. The performance of RF is the best for forecasting both positive and negative changes. For positive changes, the graph of RF shows that all lookback windows have similar AUC between 0.6 and 0.65 if the threshold is less than 1.3. AUC curves diverge at thresholds larger than 1.3. The MLP model with $p = 7$ is the best for neural nets. Although LSTM with $p = 14$ attains a higher result, it is not convincing because the LSTM lines are unstable. For negative changes, the RF model is most effective for thresholds between 1 and 1.3. MLP with $p = 7$ is still the best for neural nets. When thresholds exceed 1.3, the results for RF and CNN decline, implying that the threshold should have an upper bound. LSTM still has many oscillations which may not suggest any useful information. To sum up, the traditional algorithm, RF, is the best model for Coca-Cola Co and the MLP is better than CNN and LSTM. It requires further experiments to find the optimal lookback window size.

The result for Nike has many interesting characteristics shown in Figure 6. For positive changes, LSTM is more consistent than other models for $p = 7, 14$, and 30. The average AUC of LSTM lies in the range of 0.6 to 0.65. RF's result is around 0.6 for small thresholds, but the result of $p = 30, 60$ plunges for large thresholds. CNN and MLP are generally less accurate than others because their average results are less than 0.6. For negative changes, MLP with $p = 7, 14$ produces solid results around 0.58. In contrast, MLP with $p = 30, 60$ only achieve 0.5. LSTM also has decent performance for $p = 7, 14$. CNN and RF have fluctuated results; hence, they may not represent the true AUC level. Overall, LSTM is the best model for forecasting positive and negative changes.

The experiment on Goldman Sachs delivers mixed outcomes, as shown in Figure 7. All models have an AUC of around 0.6 on average. For positive changes, the result of RF and LSTM with $p = 7, 14$ increases as the threshold becomes large. However, RF's value peaks at threshold = 1.5, while LSTM's value drops significantly. Both RF and LSTM achieved 0.68, the highest AUC among all models. CNN and MLP underperform compared to RF and LSTM. The maximum AUC for CNN or MLP is only around 0.6. The graphs of these two models also show a high degree of variation. For negative changes, the results of all models stay around 0.6. It may suggest that machine learning models are random in predicting significant negative changes for Goldman Sachs. Overall, it requires further research to determine the optimal model.

The four experiments demonstrate that neural networks have reasonable accuracy in predicting significant changes, especially for Cisco Systems Co, where the AUC of the LSTM model goes up to 0.8. Despite the outstanding performance, we need careful consideration when deploying the neural networks into production because the AUC results may not be consistent. For example, the AUC of all models for Goldman Sachs is only around 0.6 on average, which is not far from random prediction. MLP, CNN, and LSTM can serve as terrific candidates for industrial research projects. In addition, traditional machine learning algorithm, like RF, has decent precision compared to neural networks.

3.4 Discussion and Improvements

In section 3.3, we observe that all machine learning models that we have considered in our experiments are helpful in different situations. The learning process of neural networks is not fully understood. It is still an attractive research area that contains various theories. On

the one hand, some researchers have shown that neural networks can memorize enormous amounts of data, which makes them achieve high accuracy on unstructured data ([2]). On the other hand, neural networks are proven to partially learn the underlying structure of data ([13]). In addition, some information theorists point out that the learning process of neural networks has two phases ([12]). In the first phase, the layers accumulate the information on labels and inputs while saving the data processing inequality order (i.e., lower layers have more information and vice versa). The second phase is the compression that follows any explicit regularization or a related technique. This phase is way more time-consuming compared to the last phase. In addition, the layers lose irrelevant information until convergence in this phase. It is widely recognized that the performance of neural networks relates to the depth of architecture and training data size.

Although neural networks are shown to achieve excellent AUC for Cisco Systems, the experiments can be refined in the following ways:

- **Hyperparameter tuning**: the neural networks have many parameters to change. For example, we can customize the number of layers and nodes, learning rate, regularization, activation function, training batch size, and others. These parameters can affect the performance of the classifiers. Hence, a common approach to increasing AUC is to test different parameters and select the optimal ones. However, it requires a massive amount of time and computational resources.

- **Adopting deeper neural networks**: deep neural networks may produce more accurate results. Contemporary architectures such as ResNet contain more than 1000 layers. However, deep architectures are computationally inefficient to train.

- **Additional price indicators**: there exists a wide range of indicators for stock prediction. We can insert some technical indicators as additional features in our model rather than only using the previous closing prices. Additional information may provide more insights into the model and improve the AUC.

4 Conclusion

In this paper, we study the performance of neural network models in predicting significant daily returns using prior daily returns. We select three well-known neural net architectures: MLP, CNN, and LSTM. A traditional machine learning model, RF, is used as the benchmark. The models are tested using 11-year daily price data of four US public corporations in different industries. The data is split temporarily for independent training and testing.

The results show that neural networks effectively predict significant changes in asset prices highly accurately, as in the case of the LSTM model on the Cisco Systems data. The AUC graphs of other companies also achieve 0.65, suggesting that neural networks are helpful. We conjecture that LSTM predicts more significant changes better than smaller ones.

Most literature concentrates on predicting the actual price or the direction of price changes. Our report looks at this problem differently by predicting significant price changes to filter out noise. In conclusion, our study has three achievements:

1. Explore the relatively new problem of forecasting significant changes in asset prices.

2. Compare the modern neural network models to other traditional machine learning models.

3. Attain a high prediction score of 0.8.

The results reveal that neural networks have the potential to achieve high accuracy in predicting significant price changes. The effectiveness of neural networks needs a further examination with more data.

A Appendix

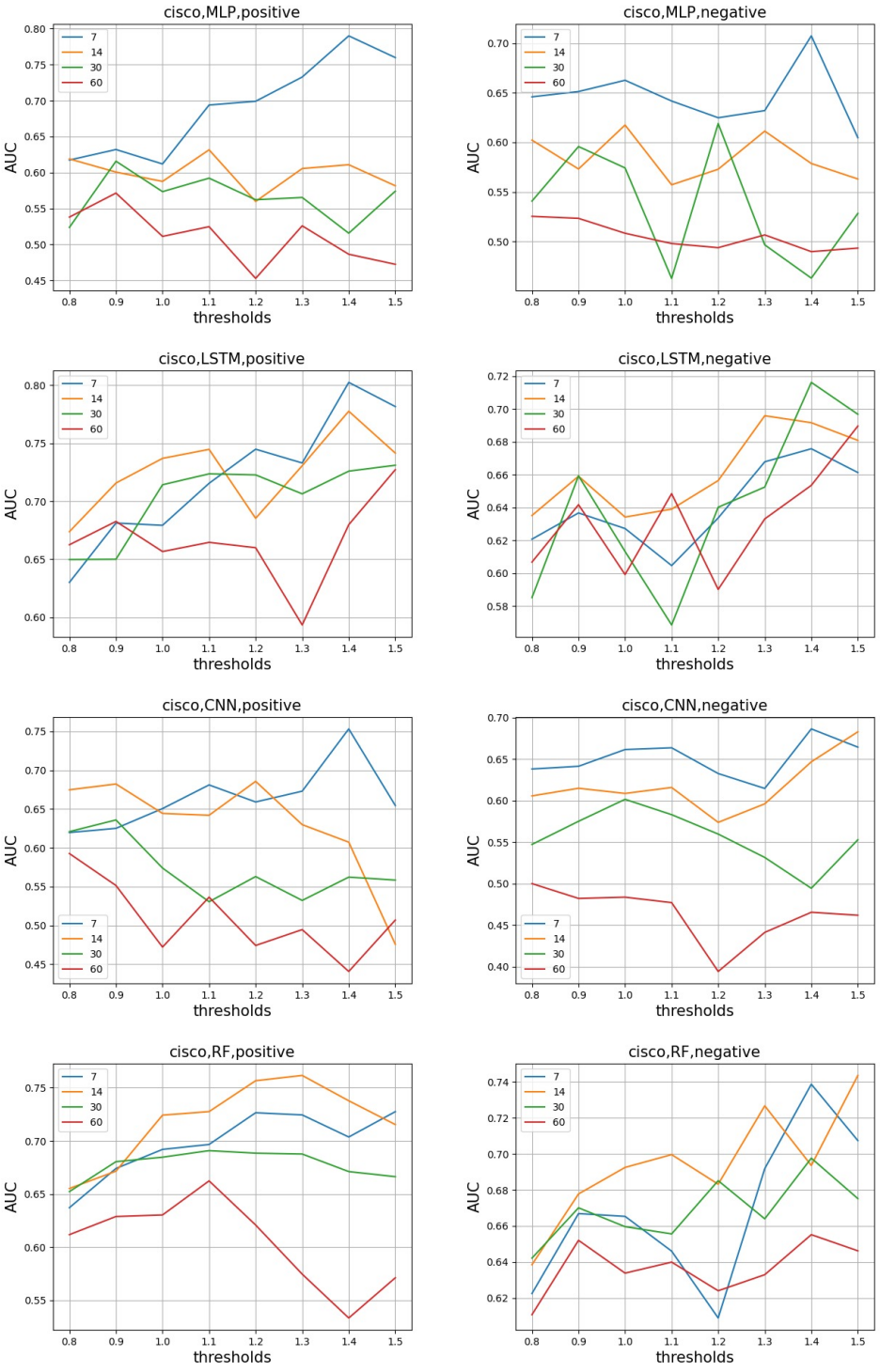


Figure 4: Cisco Systems, Inc.

Predicting significant positive and negative returns using various number of prior days: 7, 14, 30, 60. AUC measures the performance of each model. The LSTM model using a 7-day lookback period achieves the highest result.

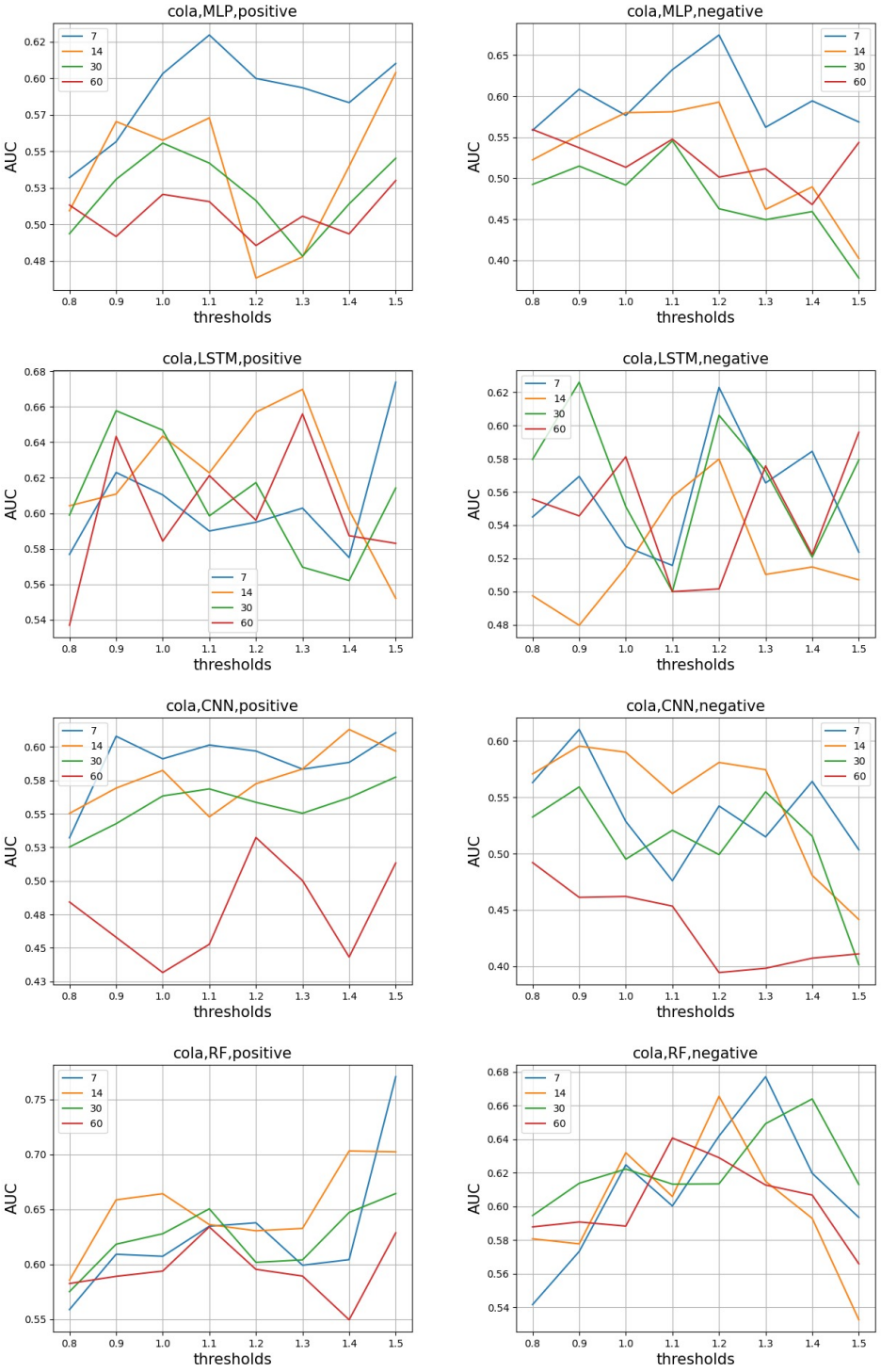
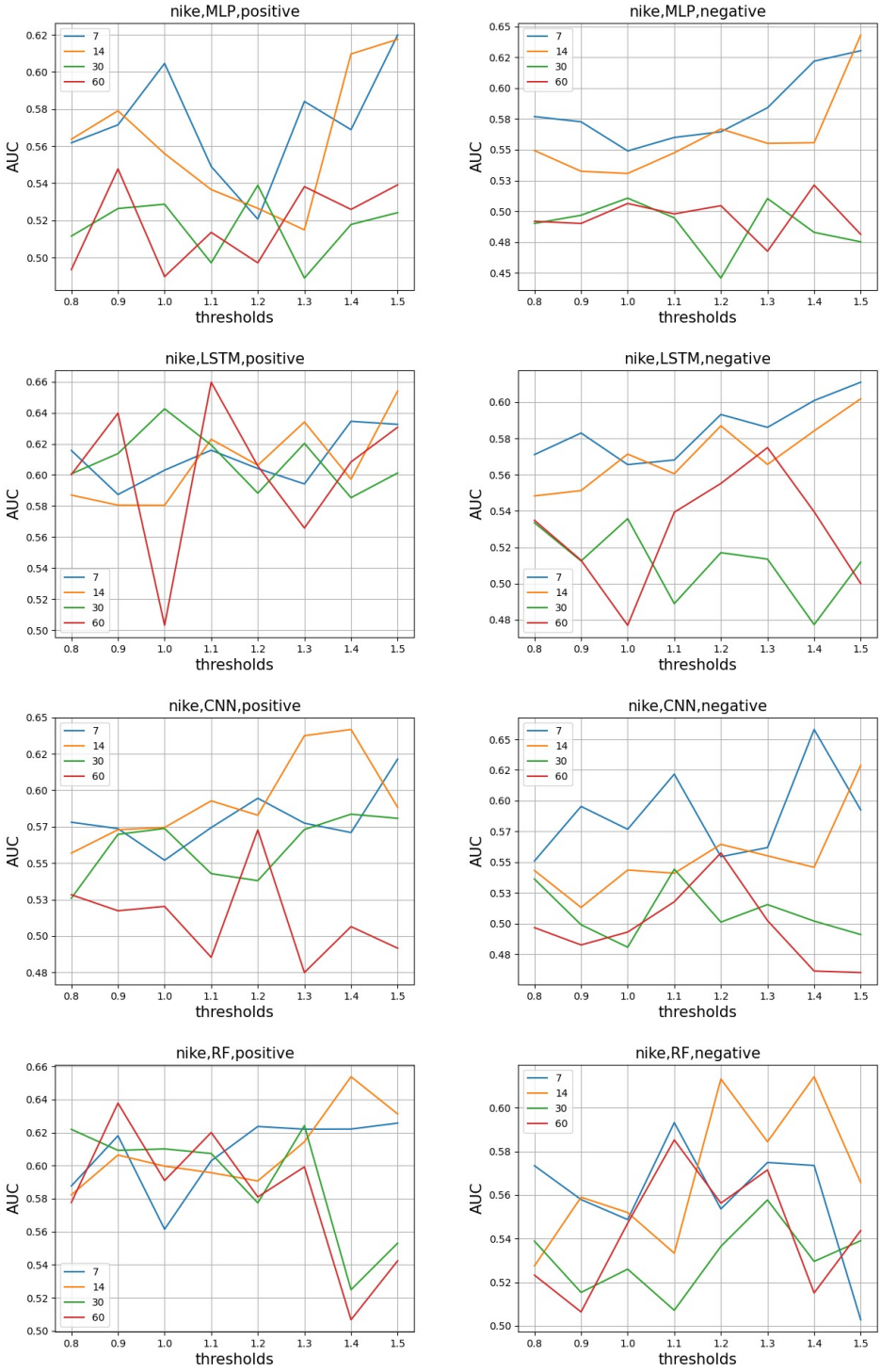


Figure 5: The Coca-Cola Company.

Predicting significant positive and negative returns using various number of prior days: 7, 14, 30, 60. AUC measures the performance of each model. The RF model using a 7-day lookback period achieves the highest result.

Figure 6: Nike Inc. Predicting significant positive and negative returns using various number of prior days: 7, 14, 30, 60. AUC measures the performance of each model. The results are mixed with different models under different parameters.



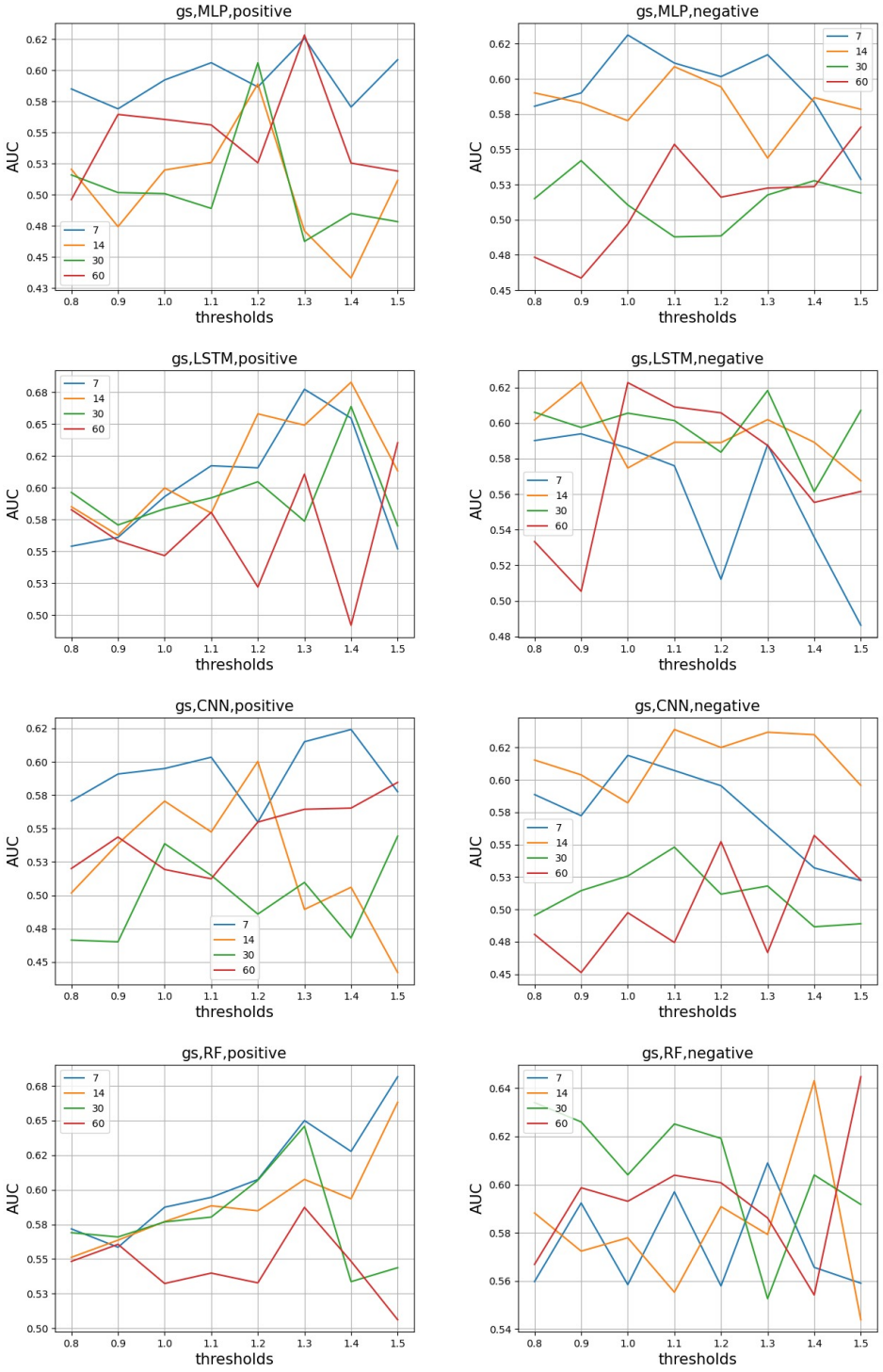


Figure 7: The Goldman Sachs Group. Predicting significant positive and negative returns using various number of prior days: 7, 14, 30, 60. AUC measures the performance of each model. The results are mixed with different models under different parameters.

References

[1] Srivastava N. , Hinton G. , Krizhevsky A. , Sutskever I. , and Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15:1929–1958, 2014.

[2] Zhang C. , Bengio S. , Hardt M. , Recht B. , and Vinyals O. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[3] Heaton J. B., Polson N. G., and J. H. Witte. Deep learning for finance: deep portfolios. *Applied stochastic models in business and industry*, 33(1):3–12, 2017.

[4] Szegedy C., Vanhoucke V., Ioffe S., Shlens J., and Wojna Z. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016.

[5] Matthew F. Dixon. *Machine Learning in Finance From Theory to Practice*. Springer International Publishing, Cham, 1st ed. 2020. edition, 2020.

[6] Fama E.F. and French K.R. Permanent and temporary components of stock prices. *The Journal of political economy*, 96(2):246–273, 1988.

[7] Kamalov F. Forecasting significant stock price changes using neural networks. *Neural Computing and Applications*, 32(23):17655–17667, 2020.

[8] Dudek G. Multilayer perceptron for GEFcom2014 probabilistic electricity price forecasting. *International journal of forecasting*, 32(3):1057–1060, 2016.

[9] Yao L. and Guan Y. An improved lstm structure for natural language processing. In *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, pages 565–569, 2018.

[10] Judy Nduati. Introduction to neural networks. https://www.section.io/engineering-education/introduction-to-neural-networks/.

[11] Gergaud O. and William T. Z. Great investors: Their methods, results, and valuation. *Journal of portfolio management*, 38(4):128–, 2012.

[12] Shwartz-Ziv R. and Tishby N. Opening the black box of deep neural networks via information, 2017.

[13] Arpit D. , Jastrzundefinedbski S. , Ballas N. , Krueger D. , Bengio E. , Kanwal M. S. , Maharaj T. , Fischer A. , Courville A., Bengio Y. , Lacoste-Julien S. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 233–242. JMLR.org, 2017.

[14] Xiong W., Wu L., Alleva F., Droppo J., Huang X., and Stolcke A. The microsoft 2017 conversational speech recognition system. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5934–5938, 2018.