

Spring 5-4-2021

Machine Learning in Stock Price Prediction Using Long Short-Term Memory Networks and Gradient Boosted Decision Trees

Carl Samuel Cederborg

Follow this and additional works at: <https://digitalcommons.spu.edu/honorsprojects>



Part of the [Artificial Intelligence and Robotics Commons](#)

Recommended Citation

Cederborg, Carl Samuel, "Machine Learning in Stock Price Prediction Using Long Short-Term Memory Networks and Gradient Boosted Decision Trees" (2021). *Honors Projects*. 132.
<https://digitalcommons.spu.edu/honorsprojects/132>

This Honors Project is brought to you for free and open access by the University Scholars at Digital Commons @ SPU. It has been accepted for inclusion in Honors Projects by an authorized administrator of Digital Commons @ SPU.

MACHINE LEARNING IN STOCK PRICE PREDICTION

by

CARL CEDERBORG

FACULTY MENTORS:
DR. CARLOS ARIAS

HONORS PROGRAM DIRECTOR:
DR. CHRISTINE CHANEY

A project submitted in partial fulfillment of the requirements
for the Bachelor of Arts degree in Honor Liberal Arts
Seattle Pacific University
2021

Presented at the SPU Honors Research Symposium
Date: 5/22/21

ABSTRACT

Quantitative analysis has been a staple of the financial world and investing for many years. Recently, machine learning has been applied to this field with varying levels of success. In this paper, two different methods of machine learning (ML) are applied to predicting stock prices. The first utilizes deep learning and Long Short-Term Memory networks (LSTMs), and the second uses ensemble learning in the form of gradient tree boosting. Using closing price as the training data and Root Mean Squared Error (RMSE) as the error metric, experimental results suggest the gradient boosting approach is more viable.

Honors Symposium: ML is an unbelievably powerful tool, and the application of ML must be subject to our biblical calling as stewards. As technology progresses to make us increasingly productive, we must direct what we produce towards ends that glorify God. Just as importantly, we must be vigilant to the great temptation to become lost in decadence. ML has wildly successful applications in the financial world that far surpass the scope of this paper, but we cannot lose sight of He who provides. A firm grounding in scripture and a healthy understanding of Providence should be enough to keep those of us who pursue the blessing of technology from becoming lost in our own grandeur.

Machine Learning in Stock Price Prediction

Carl Cederborg
College of Arts and Sciences
Seattle Pacific University
Washington, United States of America
cederborgc@spu.edu

ABSTRACT

Quantitative analysis has been a staple of the financial world and investing for many years. Recently, machine learning has been applied to this field with varying levels of success. In this paper, two different methods of machine learning (ML) are applied to predicting stock prices. The first utilizes deep learning and Long Short-Term Memory networks (LSTMs), and the second uses ensemble learning in the form of gradient tree boosting. Using closing price as the training data and Root Mean Squared Error (RMSE) as the error metric, experimental results suggest the gradient boosting approach is more viable.

Keywords

Machine Learning (ML), Deep Learning, Ensemble Learning, LSTM, Decision Trees, Gradient Boosting, RMSE

1. INTRODUCTION

The interest of quantitative analysis has always been to make connections between sets of data and increase accuracy of predictions. Different tools, systems, and mathematical models have been applied to this problem over the years, and as of late, ML has become increasingly popular among investment funds. ML is the practice of programming computers so they can be fed data to learn to solve a problem. There are many different types of ML with different strengths; this paper will focus on two in particular: deep learning in the form of LSTM networks and ensemble learning in the form of gradient tree boosting.

Stock prices and the market in general are extremely dynamic systems, and as such, are difficult to predict. There are countless variables that affect stock price, ranging from quantitative indicators to feeling/opinion, also known as market sentiment. A reason ML has become popular in this field is that ML is able to draw connections between data points in sets, often providing insight into causation. However, there must be caution because machine

learning will find patterns in a data set even if there are none, as the famous quote attributed to economist Ronald Coase goes, “if you torture the data enough, nature will always confess.” [1]

Neural networks, of which LSTMs are a subset, have been directed at market sentiment due to their success with natural language processing, so they may properly discern the attitude of news articles [2]. Their success comes from their ability to “remember” data, allowing them to effectively process things like context in language and long-term dependency in time series. Memory allows them to be directly applied to price prediction using time series data. As shown in section four, the LSTM network used for this paper takes in time series data.

The fundamental building block of gradient tree boosting is a decision tree, which is used for both classification and regression. The latter is used for the sake of stock price prediction. Decision trees are convenient because they do not require feature scaling before training, reducing the amount of preprocessing of data that needs to be done.

It is important to note that a large part of ML is data science, and not simply algorithm design. This is partially because of an influential paper in 2001 [3] that showed many different algorithms performing about the same once given enough data, and it was further solidified by a paper in 2009 suggesting the same [4]. This realization that data was essentially as effective as a good algorithm is important because it has defined ML for the past couple decades. In practice, data and feature engineering ends up being the major focus of work because the heavy lifting of building models is supplied by libraries.

Section 2 explains the models and training techniques used. Section 3 lists a series of related works on machine learning. Section 4 explains the methodology of the tests. Section 5 displays the results. And section 6 is an explanation of the results and what could be done to improve.

2. BACKGROUND

LSTMs were made to solve the problem basic recurrent neural networks (RNNs) had with remembering data from too many time steps in the past. With a basic neuron containing only an activation function, a simple RNN had trouble learning how to hold on to relevant information. An LSTM corrects for this by altering the inside of the cell, taking in and therefore outputting an additional state to each subsequent cell. Among the alterations are a forget gate that determines which information to keep from the previous state, a weighted combination of what information to add to the current state, and finally an activation function of the updated state to form the new output.

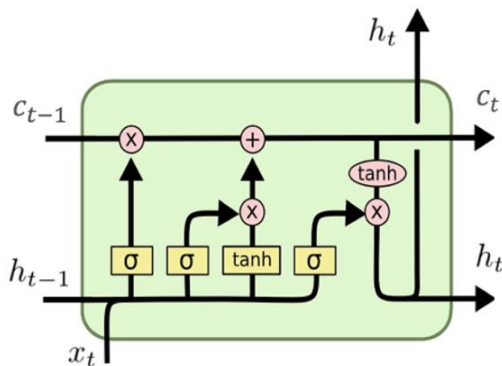


Figure 1. An LSTM Cell

Figure 1 shows the insides of an LSTM cell. The yellow boxes are neural network layers that use a weighted input plus bias in an activation function to produce output, in this case with sigma or tanh as activation functions. Sigma stands for a sigmoid function that takes in numerical input and outputs a value ranging from 0 to 1. Tanh performs a similar task but output ranges from -1 to 1. The circles are pointwise operations on the vector input, which can be simple like multiplication or addition, or it can be an activation function like tanh. The cell takes as input a previous state, c_{t-1} , a previous output, h_{t-1} , and a current data input, x_t . In the path of the first neural network layer is the implementation of the forget gate. Here, x_t and h_{t-1} are concatenated and put through a sigmoid layer, producing a series of values ranging from 0 to 1 for each value in the state c_{t-1} . When the series is multiplied by c_{t-1} , a value of 0 fully drops or “forgets” previous values in the state, while a value of 1 fully keeps or “remembers” the previous values in the state. The second section in the LSTM processes what values are to be added to the state (produced by the tanh layer) and scaled by some amount (produced by the sigmoid layer). After the new values are scaled, they are added to the state produced by the forget gate, creating c_t , the current

state. In the final section, c_t is put through a tanh function (not a neural network layer) and scaled by a sigmoid layer of h_{t-1} and x_t ; this produces h_t , the output of the cell. Both c_t and h_t are passed to the next layer in the network; if it is an LSTM layer, the process is repeated.

With the addition of the continuous state and the forget gate, an LSTM is able to select which information it wishes to keep moving forward through time. This allows connections to be drawn between data points that previously were lost to the mangling of simple activation functions. An LSTM is good with time series data for that reason; it can understand that price from X number of steps ago affects the current output. Put simply, deep learning is a neural network that contains multiple “hidden” layers, or layers that are neither the input nor output layer.

A decision tree is a type of supervised ML that can be used for both classification as well as regression. They work by splitting the data set into subsets in such a way as to minimize a cost function. In the case of regression, often this is mean squared error. Unrestrained, decision trees are prone to overfitting, as they will split the data all the way down to the individual points. Normally, parameters are set to specify how many times the tree can split the data, how many points must be in each split, etc. Given the right parameters, a decision tree will produce a well-fit model with good predicting power.

Gradient tree boosting is a form of ensemble learning, in which many predictors are aggregated to increase accuracy. This is based on the law of large numbers; if there are enough predictions with even only 51% accuracy and the majority opinion is selected, there is a significantly higher than 51% likelihood of it being correct. Boosting is an ensemble method that trains the predictors sequentially, attempting to correct the previous predictor’s errors. Gradient tree boosting uses decision trees as the predictors, and it trains each following predictor on the residual error of the one prior. In this way, having more predictors accounts for the errors of the previous predictors. The model in this paper uses the optimized Python library, XGBoost [5, 6], which has found recent success in algorithm competitions.

In machine learning models, there are some common problems to be aware of. The overarching issue in training is the Bias/Variance tradeoff. If the model is too simple (highly constrained/low degree of freedom), then it is unable to adapt to the data and is prone to underfitting. If the model is too complex (high degree of freedom), then it adapts too well to the data and is prone to overfitting. The goal before and during training is to minimize these two

conflicting sources of error so the model can generalize well. One factor in the complexity of the model is the number of parameters being measured. Another is the type of model; decision trees are easily capable of overfitting due to the lack of assumptions made about the data.

There is a tendency to anthropomorphize machine learning, but what either of the models are actually doing under the hood often differs greatly from a human's perception of the problem. The split here is referred to as a white or black box approach. Neural networks are black boxes; it is unclear why they make the decisions they do after training. Decision trees are white boxes; their method can be broken down easily into what factors they consider, how important they are, and more.

Coding either of these machine learning methods used to take much more work. Fortunately, creating these networks has been trivialized by many modern libraries, namely Keras, Tensorflow, and XGBoost, so more emphasis can be placed on feature selection and fine tuning the parameters.

3. RELATED WORKS

Neural Nets have been applied to time series data even before the advent of the LSTM, just not as well. LSTMs are mainly used in prediction or sequence classification [2, 7] by themselves, while different neural nets like convolutional neural networks can be used in more complex graph structures [8]. For more complicated structures like multi-task RNNs, in which attention-based neural nets are used, LSTMs have shown to not be as effective [9].

Gradient boosting was first published in 1997 [10] improving on the concept of boosting, which had existed for a short while. In 2016, Tianqi Chen published his paper [6] that established XGBoost, a system based on extreme gradient boosting.

Other kinds of machine learning have also been directed at finance. Support vector machines (SVMs) are a popular, supervised approach that have had some amount of success in prediction [11, 12]. Reinforcement learning (RL) is similar in that it can use neural networks as their decision policy, but the machine learning is the agent that acts on the market, not just a predictor. RL has found real financial success as automated trading bot [13].

4. METHODOLOGY

These tests were run on an Intel Core i7-7700HQ CPU at 2.80GHz with four cores, eight logical processors, and 16GB of RAM. The language used was Python version 3.8, utilizing the Keras, Tensorflow, and XGBoost libraries, and the work

was done in a mix of Atom/Command Line and Jupyter Notebooks.

The data was pulled from TradingView [14], a chart tracking and data website that can integrate with brokers to track the market. The data used in these tests was a comma-separated value (.csv) file of AAPL stock from 2015 to 2020 in two hour (2h) candles. The Pandas [15] library, which provides matrix and database functionality, was used to process and sort the .csv data. Each model was trained on the closing price data.

The LSTM was constructed from the Keras library using three LSTM layers followed by one Dense layer containing one neuron for the output. A Dense layer is a fully connected layer, where each neuron (in this case, just one) receives input from every neuron in the previous layer. The model used in these tests consisted of three LSTM layers of 50 neurons each; the number of neurons for each layer is selected with both functionality and training time in mind. It is not clear that having more neurons strictly means a better result, but too few neurons does increase error. The loss function used was mean squared error. The input shape of the first LSTM layer was [100,1] because the model was being trained taking in the previous hundred days of data and making a prediction of the next day. For most of the test runs, dropout layers were used in between the LSTM layers, which weaken the impact of certain layers by reducing their effects by a percentage.

The gradient tree boosting model using XGBoost used regression with squared loss as the learning objective, and it had estimators set to 1000, meaning there were 1000 gradient boosted trees contributing to the final output.

The data was split at 75/25 percent for the LSTM training/test data. Out of 7474 data points of closing price, the training set size was 5605, and the test set size was 1869. The training set for the LSTM consisted of two arrays, `x_train` for the input and `y_train` for the expected output, which is supervised learning. `X_train` contained a sequence of the 100 prior data points for each point in the training data, and `y_train` was each 101st data point as the expected output. Once the data is cleaned and split, the model is fitted and validated using the Keras `fit()` and `predict()` methods. The `fit()` method's input parameters such as dropout, epochs, and batch size were tested at different levels, as shown in Results. The output of the model using `predict()` is compared to the `y_train` and `y_test` (the data points in the test set) arrays to compute the RMSE.

The XGBoost model was also trained on the entire AAPL data set (7474). The process for training and fitting the model was similar, splitting the data this time at 80/20 percent for train/test set. Decision trees

also use supervised learning, so there was an input, x_{train} , and expected output, y_{train} . The x_{train} array for this model consisted of the current closing price, and the y_{train} array consisted of the following day's closing price. Once the data was cleaned and split, the model was fitted using XGBoost's `fit()` method and validated using a method called walk forward validation. Walk forward validation is a process in which predictions are made, but the model

is retrained on the "new", real data of the test set every step. This way, the model is kept up to date. The "expanding window" method was used, in which new test data points are added to the old set and none are removed.

Run/Input	Dropout: #layers (amount)	Epochs	Batch Size	Train RMSE	Test RMSE
1	None	25	64	39.47	88.65
2	3 (.2, .2, .2)	25	64	39.26	86.22
3	3 (.2, .2, .2)	20	128	38.75	85.17
4	3 (.8, .5, .5)	25	64	39.3	81.53
5	3 (.8, .5, .5)	25	256	39.07	83.06

Table 1: LSTM Results

5. RESULTS

Keeping in mind that the validation methods were different, the results of both models were vastly different. The LSTM model was fitted with different parameters, but they all produced similar results, as shown in Table 1. Utilizing substantial dropout in the training process achieved the best test RMSE at 81.53 dollars. Figure 2 shows the graph of the last 768 data points in the test set for Run 4.

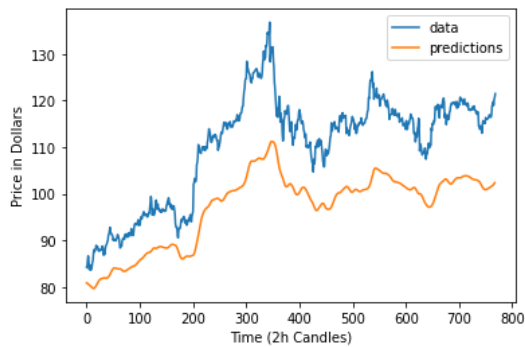


Figure 2: LSTM Predictions for Run 4

While it may have had the lowest Test RMSE, the graph shows a bit of underfitting. Figure 3 shows the graph for Run 1, which fits the data better but has a higher RMSE.

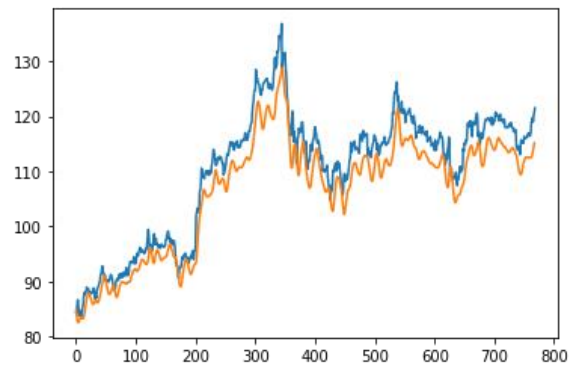


Figure 3: LSTM Predictions for Run 1

The results for the XGBoost model were substantially better, with an RMSE of 1.28 over the test set. Figure 4 shows the last 768 data points in the test set for the XGBoost model.

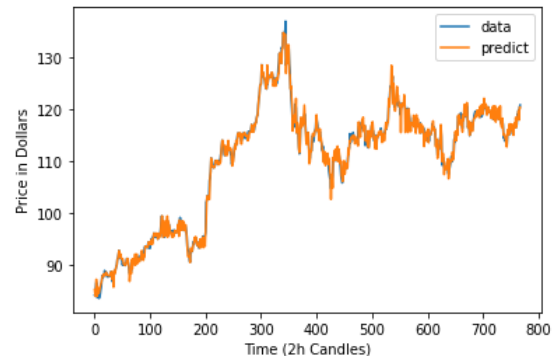


Figure 4: XGBoost Predictions

A zoomed-in perspective of Figure 4 is shown in Figure 5, which is the last 50 data points in the test set. And Figure 6 shows a point where the XGBoost model can have high variance.

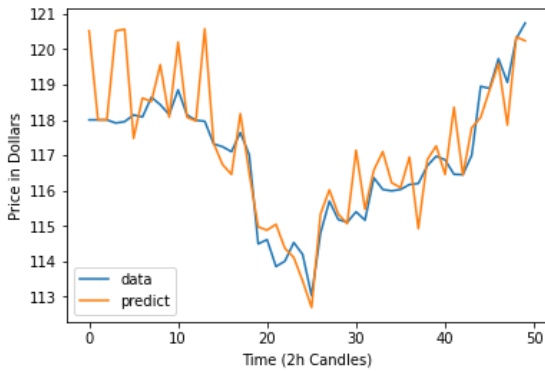


Figure 5: Last 50 Points of Figure 4



Figure 6: High Variance XGBoost

6. CONCLUSION

From the data observed, it seems to suggest that gradient boosted decision trees are superior to basic LSTMs when it comes to time series prediction. There are multiple flaws in the comparison and methodology that could be improved upon. These improvements include, but are not limited to, lining up the dataset, fleshing out the model by testing different layer structures, and using multivariate analysis instead of univariate. The validation method for each model differs as well, which can skew results if interpreted incorrectly. RMSE is in the units of what is being measured, which in this case is dollars. Lower RMSE is better, but depending on the context, an RMSE of 30 could be acceptable. If the stock price is 1000 dollars, an RMSE of 30 would be good. However, in this case, AAPL only gets up to ~120 dollars, which leads me to conclude that my simple LSTM did not perform well, despite showing a promising graph at times.

The market viability of these models was not the initial goal, but even so it is questionable for multiple reasons. Back testing itself is full of pitfalls and never guarantees future results. Referring back to the black box nature of neural networks, it is difficult to determine what throws off the results. The LSTM

model does not seem to be grossly underfitted, but it is underfitted to some degree. On the other hand, there is high variance in the XGBoost model that may speak to some overfitting. An increase in complexity, namely updating from univariate to multivariate analysis (more features) may help both models perform better.

While these two methods were just predictors, it would be interesting to develop a RL method that trades on its own. Machine learning is often used as an aid to discretionary trading, but RL agents act on their own. Further research to be pursued in the way of RL would be applying the deep neural net I created and use it to update a policy.

REFERENCES

- [1] R. Coase, Lecture, Topic: "How Should Economists Choose?" American Enterprise Institute, Washington, D.C., 1981.
- [2] Huy D. Huynh, L.Minh Dang, and Duc Duong. 2017. A New Model for Stock Price Movements Prediction Using Deep Neural Network. In *SoICT '17: Eighth International Symposium on Information and Communication Technology, December 7–8, 2017, Nha Trang City, Viet Nam*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3155133.3155202>
- [3] Michele Banko and Eric Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics (ACL '01)*. Association for Computational Linguistics, USA, 26–33. DOI:<https://doi.org/10.3115/1073012.1073017>
- [4] Alon Halevy, Peter Norvig, and Fernando Pereira. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems* 24, 2 (March 2009), 8–12. DOI:<https://doi.org/10.1109/MIS.2009.36>
- [5] "XGBoost Documentation," *XGBoost Documentation - xgboost 1.5.0-SNAPSHOT documentation*, 2020. [Online]. Available: <https://xgboost.readthedocs.io/en/latest/>. [Accessed: 14-Apr-2021].
- [6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. DOI:<https://doi.org/10.1145/2939672.2939785>

- [7] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock Price Prediction via Discovering Multi-Frequency Trading Patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. Association for Computing Machinery, New York, NY, USA, 2141–2149. DOI:<https://doi.org/10.1145/3097983.3098117>
- [8] Pratik Patil, Ching-Seh Mike Wu, Katerina Potika, and Marjan Orang. 2020. Stock Market Prediction Using Ensemble of Graph Theory, Machine Learning and Deep Learning Models. In *Proceedings of the 3rd International Conference on Software Engineering and Information Management (ICSIM '20)*. Association for Computing Machinery, New York, NY, USA, 85–92. DOI:<https://doi.org/10.1145/3378936.3378972>
- [9] Chang Li, Dongjin Song, and Dacheng Tao. 2019. Multi-task Recurrent Neural Networks and Higher-order Markov Random Fields for Stock Price Movement Prediction: Multi-task RNN and Higher-order MRFs for Stock Price Classification. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '19)*. Association for Computing Machinery, New York, NY, USA, 1141–1151. DOI:<https://doi.org/10.1145/3292500.3330983>
- [10] L. Breiman, “Arcing the Edge” UC Berkeley Statistics Department, Berkeley, CA, Technical Report 486, 1997
- [11] Carson Kai-Sang Leung, Richard Kyle MacKinnon, and Yang Wang. 2014. A machine learning approach for stock price prediction. In *Proceedings of the 18th International Database Engineering & Applications Symposium (IDEAS '14)*. Association for Computing Machinery, New York, NY, USA, 274–277. DOI:<https://doi.org/10.1145/2628194.2628211>
- [12] Hongming Wang. 2020. Stock Price Prediction Based on Machine Learning Approaches. In *Proceedings of the 3rd International Conference on Data Science and Information Technology (DSIT 2020)*. Association for Computing Machinery, New York, NY, USA, 1–5. DOI:<https://doi.org/10.1145/3414274.3414275>
- [13] Alexander A. Sherstov and Peter Stone. 2004. Three automated stock-trading agents: a comparative study. In *Proceedings of the 6th AAMAS international conference on Agent-Mediated Electronic Commerce: theories for and Engineering of Distributed Mechanisms and Systems (AAMAS'04)*. Springer-Verlag, Berlin, Heidelberg, 173–187. DOI:https://doi.org/10.1007/11575726_13
- [14] “TradingView” 2021. [Online]. Available: <https://www.tradingview.com/>. [Accessed: 2-Apr-2021].
- [15] “Pandas API Reference” 2021. [Online]. Available: <https://pandas.pydata.org/docs/reference/index.html>. [Accessed: 3-Apr-2021].

Honors Symposium Presentation

The interest of quantitative analysis in finance has always been to make connections between datasets and to increase the accuracy of predictions. Different tools, systems, and mathematical models have been applied to these endeavors over the years, and as of late, machine learning has become increasingly popular among investment funds. Machine learning is the practice of programming computers so they can be fed data to learn to solve a problem. There are many different types of machine learning with different strengths, and my project focused on two in particular: deep learning in the form of Long Short-Term Memory networks and ensemble learning in the form of gradient tree boosting.

(Slide Change)

When we say a machine can “learn” from data, we mean that data is put through an algorithm, an output is measured with a cost function, and the machine attempts to minimize that cost function. Cost functions and the method the machine takes to minimize them (known as the learning method) differ from task to task and across different machine learning architectures. One way learning methods can be categorized is by the amount of human supervision involved, called supervised, unsupervised, semi-supervised and reinforcement learning. Both types of learning implemented here use supervised learning methods, meaning the input data comes with a corresponding label which shows the expected output. This is the most direct form in the sense that the programmer is giving the algorithm the answers in the hope it will learn to generalize. The tasks a machine learning algorithm is expected to perform generally fall into two categories: classification, such as identifying pictures, and regression, such as time series prediction. As the name suggests, stock price prediction is most directly represented as a time series prediction problem (although there are other ways to conceptualize it).

Stock prices and the market in general are extremely dynamic systems, and as such, are difficult to predict. There are countless variables that affect stock price, ranging from quantitative indicators to feeling/opinion, also known as market sentiment. A reason machine learning has become popular in this field is that it is able to draw connections between data points in sets, often providing insight into causation. However, we must be cautious because machine learning will find patterns in a dataset even if

there are none, as the famous quote given in a 1981 lecture by economist Ronald Coase goes, “if you torture the data enough, nature will always confess.”

(Slide Change)

The first model I created is an LSTM, which is a type of recurrent neural network. To get to LSTMs, the fundamentals of neural nets should be explained. Based originally on a conceptual representation of the brain, the terminology has remained, but the similarity in function has mostly faded. A neural net is composed of layers of “neurons”, or cells, that contain what is called an activation function, which typically sorts values between an easier-to-handle range of 0 and 1 or -1 and 1. These activation functions take as input the weighted sum of the previous layer’s outputs and they output the result to one or more of the neurons in the next layer until the output layer is reached. These connections have a weight that represents the strength of the connection, and deep learning is simply when there are layers in between the input and output layer, called hidden layers. Neural nets train through a process called Gradient Descent, in which partial derivatives and the chain rule are used to determine how much each input is responsible for the output, thereby indicating which weights should be adjusted to help minimize the cost. This is often represented in an analogy of a climber lost in the mountains trying to find his way down to the valley. A possible solution is to repeatedly go downhill in the direction that is steepest. Eventually, when no direction has a “downhill”, the climber has reached the bottom, also known as a minimum. However, this may only be a local minimum, such as between two mountains but not yet in the valley. There are multiple methods to avoid this, one of which for regression is the shape of the mean squared error function. Because mean squared error is a convex function, it implies that there is only one minimum, which makes it global. Additionally, the learning rate is set higher at the beginning of training and decays over time to settle into a minimum.

(Slide Change)

LSTMs were made to counter what is called the vanishing gradient problem. Due to the shape of activation functions like sigmoid or hyperbolic tangent, the gradient can be vanishingly small, sometimes to the point of stopping updates to weights altogether. LSTMs corrected this by changing the contents of

the neuron from a simple activation function to also contain a memory state that is passed to subsequent cells. This means that LSTMs are able to remember patterns for much longer than a vanilla RNN because they can choose what values to add to and forget from the state. (Quick Description if time is needed)

(Slide Change)

The second model uses decision trees, which are a bit simpler than LSTMs, but they are quite powerful. They work by splitting the dataset in sections as to minimize some cost measurement, which in the case of regression is often mean squared error. The number of splits determines the depth of the tree, which is specified by a hyperparameter, the parameters we give the machines that alter their learning process. This graph is showing the likelihood of kyphosis after spinal surgery given the vertebrae in which it started and the age at which surgery was performed.

(Slide Change)

The second model is not just a single decision tree, but rather many, utilizing what is called ensemble learning. Ensemble learning is any learning method that combines weak learners, learners that are barely better than random guessing, into a strong learner. This operates on the statistical principle of the law of large numbers, which essentially says that as the number of trials increases, the average results trend closer to the expected average. This means if you combine enough predictors that guess correctly 51% of the time and take the majority result, you can expect accuracies much higher than 51%. Boosting is a process of training the learners sequentially in order to improve results, each time compensating with weights for what the previous learners missed. Gradient boosting is a subset that fits the following learners on the previous residual error, rather than changing weights. The library used, XGBoost, implements gradient boosted decision trees, so the weak learners of the ensemble method are decision trees.

The implementation of both methods was done using libraries available to the public that have trivialized the creation of ML. Keras, tensorflow, were developed and are used by Google. XGBoost was developed by Tianqi Chen in 2016. The models were trained using AAPL two hour closing price data from 2015-2020. Around 75-80% of the data is set aside to be used in the training set. The remainder is

used as the test set, data the model has not seen in order to check its ability to generalize. The overarching issue in training is the Bias/Variance tradeoff. If the model is too simple (highly constrained/low degree of freedom), then it is unable to adapt to the data and is prone to underfitting. If the model is too complex (high degree of freedom), then it adapts too well to the data and is prone to overfitting. The goal before and during training is to minimize these two conflicting sources of error so the model can generalize well. One factor in the complexity of the model is the number of parameters being measured. Another is the type of model; decision trees are easily capable of overfitting due to the lack of assumptions made about the data, and if they had unlimited depth, they would make a split for every data point, losing generality.

(Slide Change)

On the left is the LSTM and on the right is the Gradient Boosted model. As you can see, these models likely lack any real financial viability. The LSTM, while it has a promising fit at times, had a root mean squared error of around 83 on the test set, which is terrible given a stock price ranging from 30-120 dollars over the time period. The XGBoost model is certainly more promising than the LSTM with a root mean squared error of only 1.28, but the high variance as shown in the bottom right is worrisome. In the future, I would improve the tests by putting more effort into feature selection; I would not only use closing price but other factors as well. It also would be fascinating to develop a reinforcement learning agent, which executes the trades itself, instead of just predicting.

(Slide Change)

However, in the case that they were financially successful, there would be more considerations at hand. Machine learning is an unbelievably powerful tool, and the application of machine learning must be subject to our biblical calling as stewards. As technology progresses to make us increasingly productive, we must direct what we produce towards ends that glorify God. Just as importantly, we must be vigilant to the great temptation to become lost in decadence. Machine learning has wildly successful applications in the financial world that far surpass the scope of this project, but we cannot lose sight of He who provides. A firm grounding in scripture and a healthy understanding of Providence should be

enough to keep those of us who pursue the blessing of technology from becoming lost in our own grandeur.

It is clear to me in the Parable of the Talents that God wants us to use our gifts maximally for the good. It is not good to bury the talent and become a “wicked and slothful servant,” (ESV, Matt. 25:26). In a similar vein, the wife of noble character in Proverbs 31 “does not eat the bread of idleness.” In these two passages, we see that our talents are good and should be used industriously for those around us, which is ultimately for the glory of God. Further solidifying the point, in both the Sermon on the Mount and in the Parable of the Sheep and the Goats, Christ shows that talk is not enough. It is not enough to simply say, “Lord, lord!” to enter the kingdom of heaven, but “whatever you did for one of the least of these brothers and sisters of mine, you did for me,” (NIV, Matt. 25:40). It takes action, dedication of resources, and self-sacrifice to glorify God and do his will. Importantly, the Bible also reinforces temperance, “It is good to grasp the one and not let go of the other. Whoever fears God will avoid all extremes,” (NIV, Ecc. 7:18). Though we are to be productive, we are not to ruthlessly drive ourselves into the ground for the sake of money or success.

It is a little late to say that our world is becoming increasingly controlled by technology; we are living in a digital age, and we must wield this tool responsibly. All advancements bring a wide array of new ethical concerns, and computer science is no different. Already showcased by the ethical standards of the ACM are lying, cheating, harm, and integrity. I would posit that a few more are serving political masters, seeking wealth above all else, and environmental concern, all three of which are interwoven with the aforementioned ACM standards.

I believe the political realm has still not recovered from the onset of the digital age. On one hand, the mass dissemination of information is good, but the state of our discourse has only grown more polarized, and we have grown isolated. I think of this as I develop a social media app in my senior capstone class for the sake of a grade, but I feel I would hate to be the person to inflame our current state of isolation sadly masked by faux connection. Additionally, it is easy to see how technology can be used to abuse and trample on rights. When such few people have acquired such overwhelming power to

silence speech, how do we have discourse? None of this even begins to mention the new security concerns with the development of cyberattacks. Whatever we produce in the tech realm should be done mindful of our civic duties.

Avarice is one of the oldest sins in the book, and a golden goose such as machine learning throws blatantly obvious warning signs. Personally, I have never struggled with an attachment to money. However, I am only human, and I can imagine the draw wealth could have on me, should God choose to bless me in my endeavors. Maybe I would begin to seek out a higher and higher salary in the name of security, or worse, because I think I deserve it. There should be charity in all things, and that includes the simplest, direct application, our resources. Tithing and an open-palm attitude towards money, the faith that there will again be manna tomorrow, are important factors in maintaining a healthy life with a tool like machine learning.

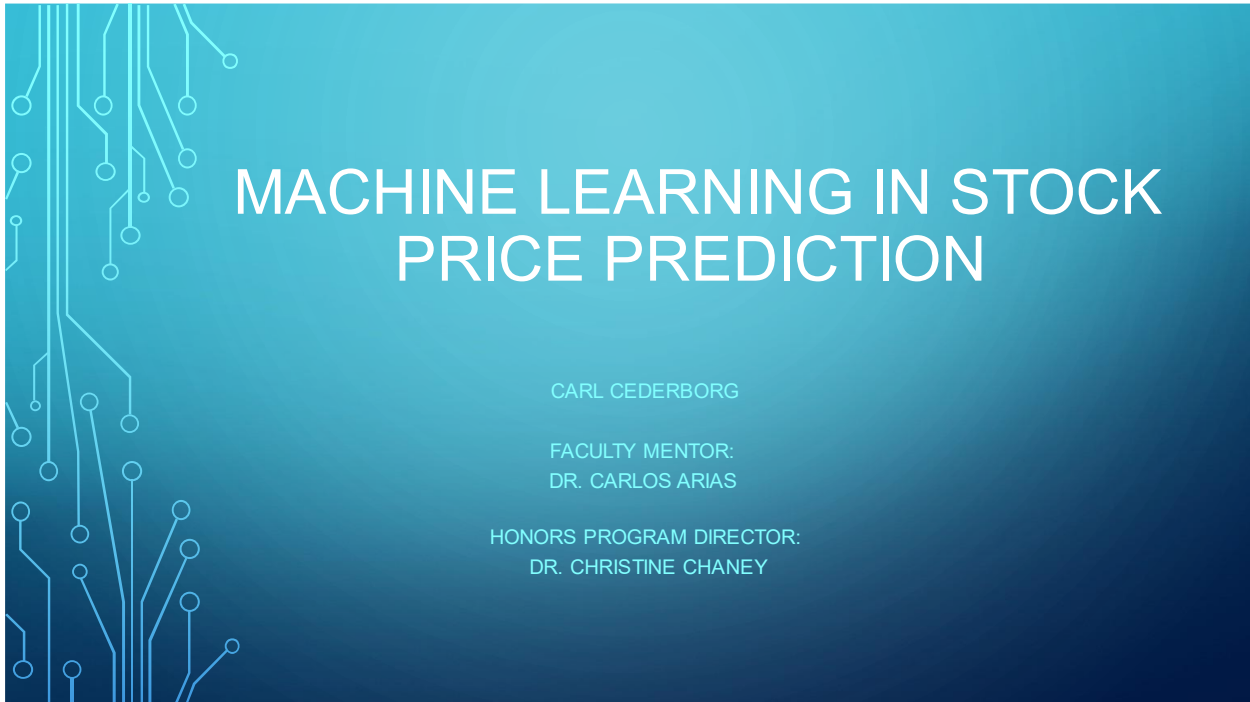
During my computer science classes at SPU, we have discussed the notion of green computing. The reality is that computers drain a lot of power, cost many expensive resources, and the understandable and cost-effective inclination to replace rather than repair can be extremely wasteful. Does this mean we cease production or somehow regress? No, I hardly think removing a tool that has helped lift millions out of poverty to be a good idea. I believe that the answer is found moving forwards. This does not mean pursuing more of the same wasteful endeavors; there should be a morality that rules above the market. The logical path of technology has been the path to efficiency, the path to cheaper, better, cleaner. Obviously, this is not always carried out, and I understand that there may need to be certain areas that are subsidized in order to incentivize those who can make change to make it, such as battery tech. But I truly believe that technology is the ally of the environment. Cleaner, safer energy like nuclear could help lessen our dependence on other forms of fuel. To decry tech and fossil fuels as evil is to be ignorant of the good they have done and continue to do in lifting people out of poverty. But I see it as perfectly reasonable to want greener energy, and how we apply technology can help further that goal.

I have always been a proponent of individual action as the route to bettering society. Here too, in being stewards, is the route forwards. Before our minds move on to grand, sweeping social change, we

should first consider and reflect on ourselves. How can *we* be the hands and feet of Christ? The path to macro change *is* micro change. How can we improve our social fabric? Be the one who builds community, gets involved in schools and churches. How can I further the kingdom of God with software? Do not be the one to build software that degrades life, and further than that, be the one to make software that honors him. This does not mean the software built has to be explicitly religious, or that we are all destined for a life in the clergy. But at all times, we are called to be the salt of the earth and a light to others. Just as a Christian artist brings glory to God by being the best artist they can be, a Christian software engineer brings glory by producing the best code they can. If it is God's will, the opportunities to larger things will be made apparent.

Us students at SPU are fortunate enough to be attending university in a free society. I hope we all have our minds set on how we can steward these gifts of both skill and opportunity. I am excited to be pursuing a startup after college with close, like-minded friends with hearts for Christ. Hopefully that way, I can turn these ones and zeroes into something beautiful, honorable, and glorifying to God.

Power Point Slides:

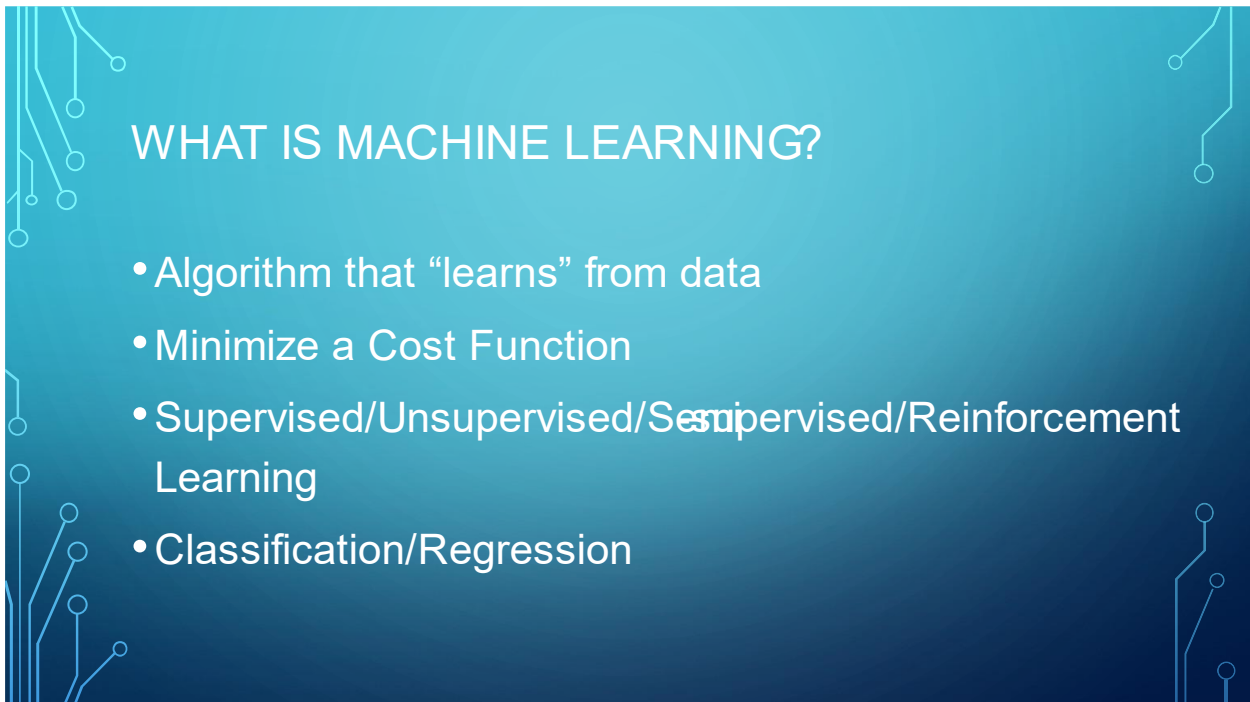


MACHINE LEARNING IN STOCK PRICE PREDICTION

CARL CEDERBORG

FACULTY MENTOR:
DR. CARLOS ARIAS

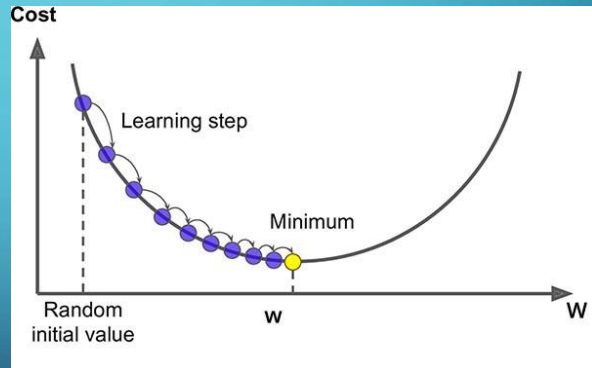
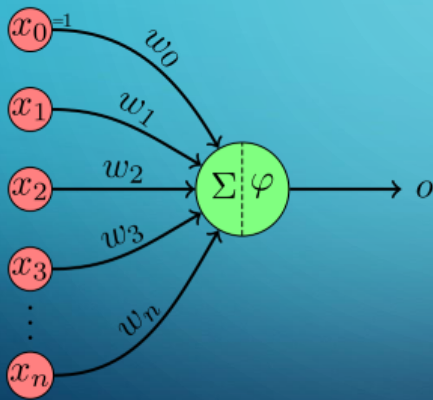
HONORS PROGRAM DIRECTOR:
DR. CHRISTINE CHANEY



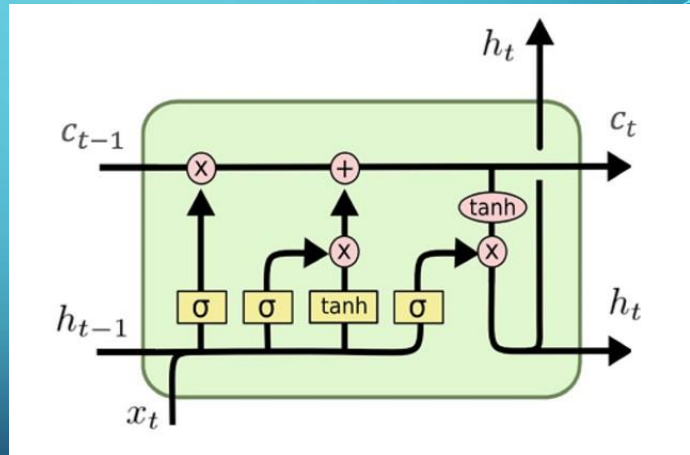
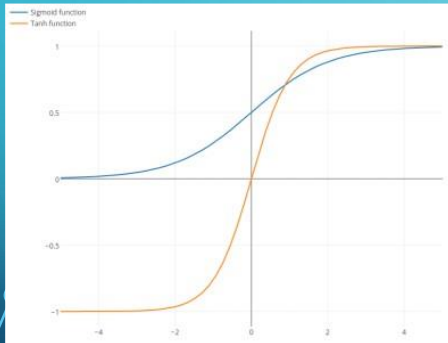
WHAT IS MACHINE LEARNING?

- Algorithm that “learns” from data
- Minimize a Cost Function
- Supervised/Unsupervised/Semi-supervised/Reinforcement Learning
- Classification/Regression

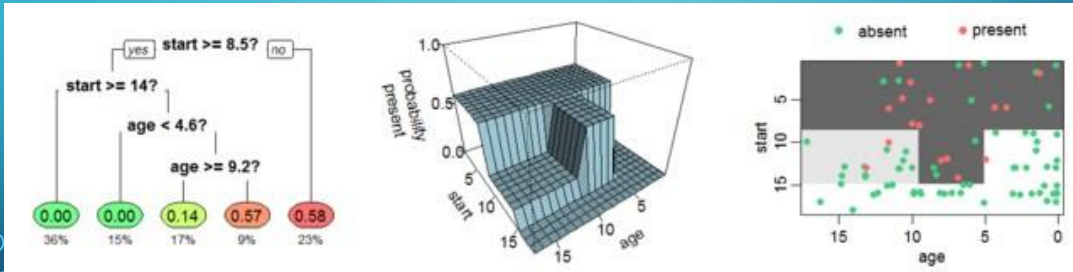
NEURAL NETS AND GRADIENT DESCENT



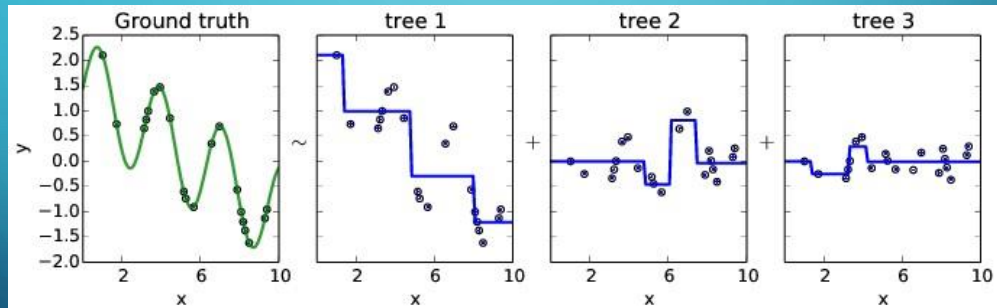
LSTMS



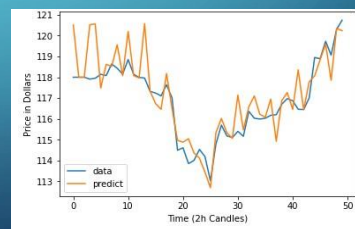
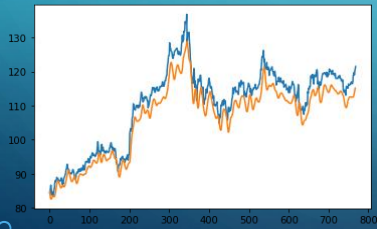
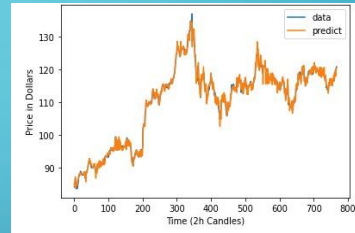
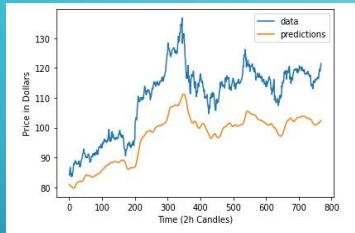
DECISION TREES



ENSEMBLE LEARNING AND GRADIENT BOOSTING



RESULTS



WHAT MORE?

- Serve one master