Deep Buffett: Stock Market Prediction using Fundamental Data

Ritik Goyal ragoyal2@wisc.edu
Param Gandhi pgandhi2@wisc.edu
Eliot Kim ejkim23@wisc.edu

Abstract

The modern stock market is a critical foundation of the global economy. Making the most ideal decisions on when to buy, hold, or sell certain stocks are the key to earning returns on investments in the stock market. In order to make decisions, investors must know when a stock for a company is overvalued or undervalued compared to the company's true value. Analyzing the fundamental financial data of a company enables investors to determine a company's true value. Our aim for this project is to use fundamental financial data of corporations to train deep learning models which output decisions on whether to buy or sell a certain corporation's stock, based on whether the stock is overvalued or undervalued. We used 143 input features, all of which are fundamental data from financial statements of publicly traded companies in the U.S. The output of our aggregated network is a buy (1) or a sell (0) decision. We trained several neural network models using this data. Initial results for all models were poor, but with hyperparameter tuning and optimization, we were able to increase accuracy. We found that Multilayer Perceptrons with 3 hidden layers, ReLU activations and dropout after each layer work well. Further, we use multiple trained models to make a prediction in a method inspired by Random Forests, which improves accuracy and provides more reliable results for our end goal. Overall, the results seem promising but need further testing and validation on a bigger and better dataset, with different network architectures and with more models in our Random Neural Network Forest. However, the current outputs provide a general indication of whether a certain stock is undervalued or overvalued, which is a valuable piece of information for investors.

1. Introduction

In the modern world, the stock market is crucial for many sectors of society. Volatile fluctuations in stock prices impact the decisions of politicians, the success of major corporations, and the retirement funds of 100 million Americans [1]. The U.S. stock market is around $34 trillion, accounting for 43% of the world's stock market value [2]. In order to take advantage of the immense potential earnings the stock market offers, investors must make correctly timed decisions on when to buy, hold, or sell certain stocks. While the stock market is notoriously difficult to predict due to volatility, there are a multitude of methods investors can use to make somewhat accurate decisions.

In this project, we will study the method of fundamental analysis. Championed by stock market expert Warren Buffett, fundamental analysis assists investors in making decisions about certain stocks [3]. Based on the company's estimated intrinsic market value, fundamental analysis indicates whether a certain company's stock is undervalued or overvalued. The intrinsic value of a company is determined by cash flow, return on assets, capital management, and other items on a company's balance sheet and income statements [4]. If the intrinsic value is greater than the value of the company's stocks, the company is undervalued. Thus, investors would likely profit from buying stocks, because in the longer term the stock price would likely rise to match the company's true value. Similarly, if the intrinsic value is lower than the value of the company's stocks, the company is overvalued, and investors should sell stocks to avoid losing money. Lastly, if the intrinsic value is around the same as the value of the stocks, then the wise decision would be to hold onto the stocks. Thus, analyzing fundamental data to estimate the real value of a company is a useful method of making decisions on investments based on quantitative factors.

Our motivation for this project was to use the deep learning techniques we learned throughout the semester to create models that provide decisions on investments based on fundamental data. The outputs of the models we created indicate whether investors should buy or sell a certain company's stock based on inputs of the company's fundamental data. Thus, after training the models, the fundamental data of publicly traded companies in the U.S. can be used as inputs to

determine the optimal decision for investors. We will discuss related work, our models, and our results in the upcoming sections.

2. Related Work

The stock market has been a prime application for deep learning techniques due to the easily quantifiable nature of financial and stock data, as well as the profitability of an accurate prediction model. We studied several papers in the literature which applied deep learning models to stock prices and stock decisions.

In their 2015 paper, Liu et al. applied neural network models to the technical analysis of stock prices [5]. Technical analysis focuses on stock prices and volume as predictors for future investments, as opposed to fundamental analysis, which uses the financial data of corporations as predictors [6]. According to Liu et al., technical analysis is more useful for short-term investment decisions, while fundamental analysis is more robust for longer-term stock market movements. Their study included 17 input features, much less than the 143 input features used in our models. The main model used by Liu et al. is the Long-Short-Term Memory (LSTM) model, which is a type of recurrent neural network (RNN). Two types of predictions were made using this model: stock prices and the rate of change of stock prices. The rate of change predictions were generally more accurate, a result which has intriguing implications for model design.

A 2018 paper by Hiransha et al. reports on the application of four different types of neural network models to predict stock prices [7]. This paper used the unique method of training the model on the stock price of one company on the National Stock Exchange (NSE) of India then testing the model on several companies on both the NSE and the New York Stock Exchange (NYSE). The model produced surprisingly accurate predictions for companies on the NYSE, because the model learned similar underlying features present in both the NSE and NYSE's stock movements.

3. Proposed Method

We experimented with various neural network architectures for both Logistic Regression and Multi-layer Perceptron (MLP) models and found that a MLP with 3 hidden layers, ReLU activations, dropout after each layer and a cross-entropy loss had the best performance.

While we were not able to get great results using MLPs alone, we noticed that some instances of our model were better at classifying some class labels better than others while comparing their F1 scores [8] and confusion matrices. For example, one particular instance of the model was better at identifying label 0 whereas another instance was better at identifying label 7. This led us to the idea of training multiple networks with different hyperparameters and using all of them in a Random Forest-like fashion to make a prediction. We give an overview of this process as follows:

1. Train multiple MLP networks with different combinations of hyperparameters (14 models in our particular case, but increasing this number should increase the prediction capability).
2. Run the test dataset on each model to generate predictions.
3. For each model, calculate the F1 score for each class.
4. For each input row, aggregate the F1 scores class-wise corresponding to each model's prediction. For example, consider a case where models A and B predict label 0, models C, D and F predict label 3 and models G and H predict label 8 and that each model M has an F1 score M_x where $x \in [0, 8]$ and represents the F1 score corresponding to class x. In this case, each label's F1 score is as follows: $[A_0 + B_0, 0, 0, C_3 + D_3 + F_3, 0, 0, 0, 0, G_8 + H_8]$
5. Finally, add up the aggregated scores for labels 0 - 4 as Sell and labels 5 - 8 as Buy.
6. If Sell > Buy, output 0. Else output 1.
7. Further optimization: Sort in decreasing order of | Buy - Sell | and pick the top N predictions as they are likely more accurate because most of the models agree on the prediction.

4. Experiments

4.1. Dataset

4.1.1 SEC Dataset

We had to build our own dataset for this project since we could not find any reliable sources of quality fundamental financial data. Companies are required to file detailed financial reports with the U.S. Securities and Exchange Commission (SEC) every quarter. So, we started out by building various scripts in Python to scrape data from the SEC website, parse, clean and organize it. The raw data had thousands of features for each company, so we picked the top 1000 features which most companies had in common and obtained a dataset with 200K rows and 1000 columns. But, this dataset had various problems such as a lot of missing

data, sparse columns and different feature names across companies that should be the same, such as Revenue, Sales Revenue, etc. So, we decided to scrap this dataset and look for another source.

4.1.2 Financial Modeling Prep API

We discovered the FMP API (`https://financialmodelingprep.com/developer/docs`) which is a free API for fundamental and stock data. By examining some of their data, we found that it was very consistent with few missing values. We scraped various endpoints of this API using Python and aggregated the data to obtain a combined dataset with 160K rows and 140 columns. We dropped all rows with missing values to obtain a final dataset with 83k rows.

4.1.3 Data Preparation v1

1. In order to label the data, we created a target column by shifting the Market Capitalization for each company back by 1 so that each row's target is to predict the Market Cap of the company in the next quarter.

2. Next, we calculated the Deltas as the percent difference between the target and current market cap.

3. Created the labels as follows:

   (a) Label 0: $\Delta < -5\%$ (Sell)

   (b) Label 1: $-5\% < \Delta < 5\%$ (Hold)

   (c) Label 2: $\Delta > 5\%$ (Buy)

4. Standardized all the numerical data and split into train, validation and test sets as follows:

   (a) Test set: All the data from 2018 or later. Split this way to facilitate backtests and also to imitate real-world usage of the model as it will be used to predict stock prices in the future.

   (b) Train set: 95% of non-test data (randomized)

   (c) Validation set: 5% of non-test data (randomized)

4.1.4 Data Preparation v2

Our models failed to perform well on the v1 dataset, so we decided to inspect it for any problems. Upon manual inspection, we discovered several problems with the dataset:

1. Market cap data was not consistent, which explained the poor performance of the model as we used this data to calculate the labels. So, we changed the classification parameter from Market Cap to Stock Price.

2. There was a huge class imbalance (28, 27, 45). When the model couldn't learn well, it just ended up classifying most inputs as a label 2 as that was the most common label. That's why the model couldn't improve past 45% accuracy.

3. There were many outliers in the Target column with really high/low returns that were most likely errors. So, we removed all these outliers.

4. We increased the number of labels from 3 to 9, ranking them in order of stock performance from 0 (worst) to 8 (best). The classification boundaries are shown in table 1. These boundaries were chosen arbitrarily so that each class had about the same number of examples.

The final dataset can be found in the Github repository along with the code used for building the models and scraping the data.

Returns	Label
-100% to -15.5%	0
-15.5% to -7.75%	1
-7.75% to -3%	2
-3% to 0.7 %	3
0.7% to 4.3%	4
4.3% to 8.25%	5
8.25% to 13.1%	6
13.1% to 21.5%	7
21.5% to 100%	8

Table 1. Classification boundaries for stock returns

4.2. Software

The list of software packages used for the project is as follows:

- Python: 3.7.4
- PyTorch: 1.4.0
- Pandas: 0.25.1
- Numpy: 1.17.2
- Matplotlib: 3.1.1
- Scikit-learn: 0.21.3

Model No.	Dimension Of Hidden Layers	Learning Rate	Dropout	Weight Initialization	Batch Size	Test Accuracy (%)
1	50*75*25	0.001	0.3	xavier_normal	64	18.77
2	50*75*25	0.001	0.3	kaiming_normal	256	18.35
3	50*100*75	0.001	0.2	xavier_uniform	64	19.26
4	75*100*50	0.001	0.3	kaiming_normal	256	19.38
5	75*100*50	0.001	0.3	sparse_	128	19.25
6	100*100*100	0.001	0.5	nan	64	18.90
7	125*250*100	0.001	0.2	xavier_normal	64	18.22
8	125*250*100	0.001	0.2	xavier_uniform	64	18.86
9	125*250*100	0.001	0.3	xavier_normal	128	18.33
10	143*286*100	0.001	0.5	kaiming_uniform	64	17.37
11	150*300*100	0.001	0.5	orthogonal_	128	18.67
12	200*300*100	0.0001	0.4	orthogonal_	64	19.52
13	200*300*200	0.001	0.3	kaiming_normal	64	18.77
14	200*300*200	0.0001	0.4	kaiming_normal	64	18.47

Figure 1. Hyper-parameter Settings for various Models

4.3. Models

We decided that a Multilayer Perceptron would be the best model for the data that we collected. We tested different hyperparameter settings and table 1 shows the results.

Figure 2 shows the accuracy and cross-entropy loss plots for model 12, which was the best out of the 14 models.

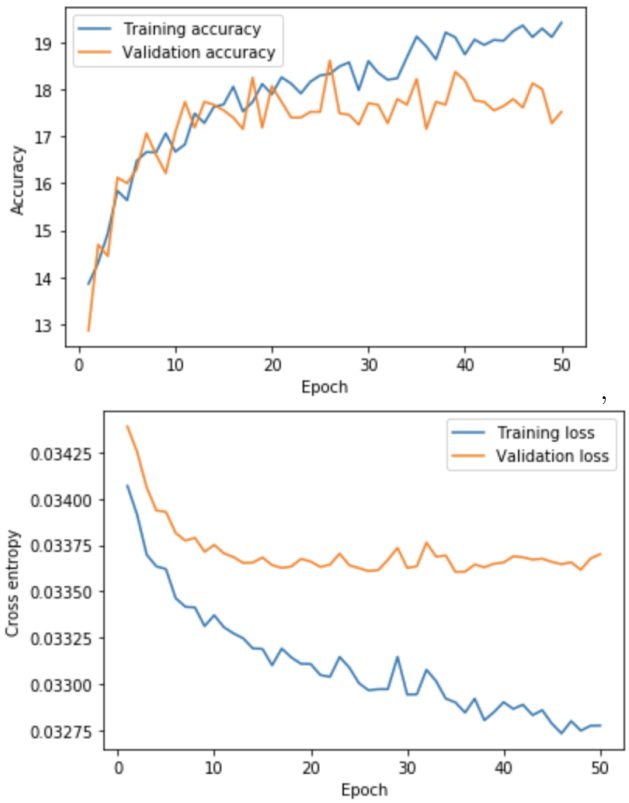

Figure 2. Accuracy and Cross-Entropy Loss plots for Model 12

5. Results and Discussion

5.1. MLP Models

We tried hundreds of different hyperparameter settings and made the following observations:

1. We found that generally, larger networks with more hidden layers or nodes converge faster and have slightly better accuracy, but overfit more than smaller networks. We used dropout after each layer to reduce overfitting.

2. Batchnorm layers had no noticeable impact on training performance.

3. Adam optimizer with very low learning rates worked much better than SGD.

4. The network continued to converge even after 500 epochs, but it's progress decreased significantly after the first 50 epochs. So, we were unable to train any of the networks past 20% test accuracy. We intend on trying out significantly longer training periods (>1000 epochs) on Model 12, which seems the most promising.

5. We tested the network's learning capacity by forcing it to overfit by using a small subset of the training data and 1000 epochs. The network was able to achieve around 85% train accuracy and continued converging, which leads us to believe that the network is capable of performing much better but just needs very long training periods that weren't feasible with the compute resources we had access to. Figure 3 shows the training results of the overfit test.

6. Weight initialization techniques like kaiming, orthogonal and xavier improved performance allowing the network to converge faster.

7. Adding more then 3 hidden layers generally hindered performance.

8. Training on raw data resulted in exploding gradients most of the time. Standardizing the data resulted in much better and stable training.

9. The network converged much faster with ReLU activations compared to Sigmoid. Sigmoid activations also resulted in vanishing gradients leading to very poor performance.

10. We also tried modelling the problem as a regression and ordinal regression problem. In both cases, we did not get good results as the network tended to 'smooth' out the outputs, i.e. output the mean of the labels every single time to artificially minimize the loss. Regularization techniques didn't improve performance significantly.

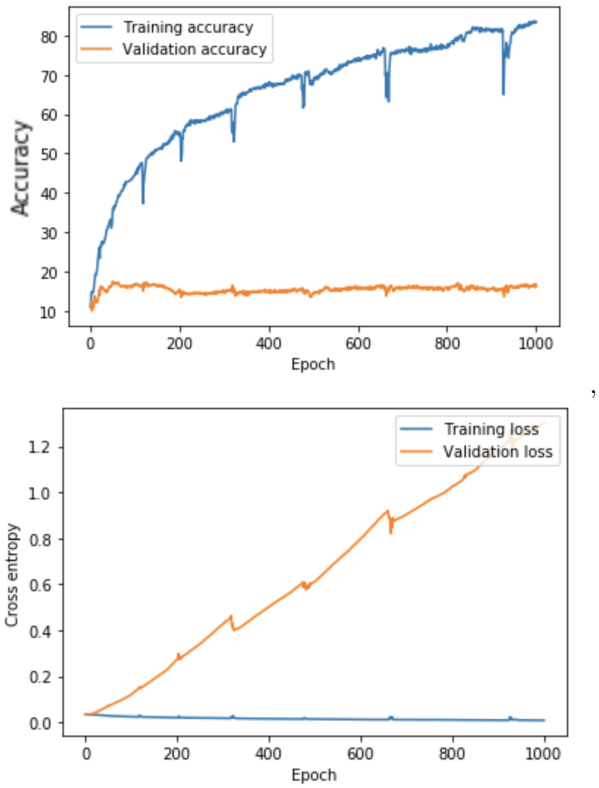

Figure 3. Training results of purposeful overfitting

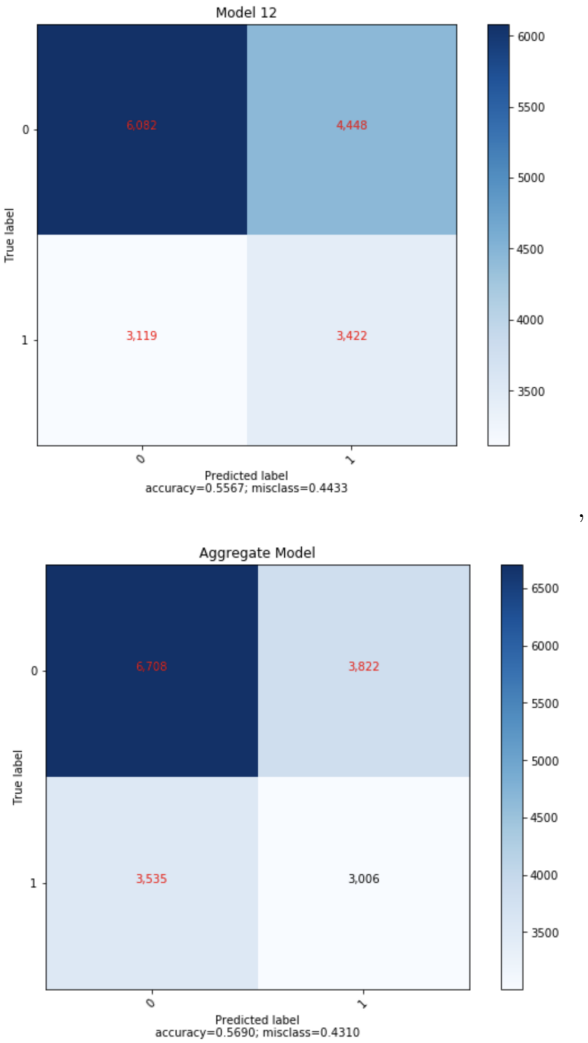

Figure 4. Confusion matrices for Model 12 and Aggregate Model

5.2. Aggregate Model

Table 2 shows the comparison between the best performing MLP model (model 12) and the Aggregate Model. Figure 4 shows the confusion matrices for the two models.

Model	Overall	Sell	Buy
Best MLP	55.67%	61.64%	47.4%
Aggregate Model	56.9%	64.5%	44.96%

Table 2. Performance comparison (Columns indicate accuracy)

We can see that the aggregate model performs slightly better than the best MLP model. It is especially better at identifying when a stock is overpriced and should be sold. But, we can get better predictions by only buying/selling the stocks with the highest confidence values. After sorting by the aggregate F1 scores generated by the algorithm and picking the top N stocks, the performance is shown in Table 3.

We can see that sorting by the aggregate score and selecting only a small subset of predictions gives us much better performance compared to any of the individual models. This is desirable because we only want to invest in a small number of companies which we are certain about. Increasing the number of original models or training the original models better will continue improving the aggregate model.

5.3. Estimated Portfolio Returns

The Aggregate Model seems to be especially good at picking out stocks that underperform the market quarter-to-quarter. This can be valuable in picking stocks for a short portfolio [9]. Our test dataset has an average return of 17.18% for long stocks (that increase in value over the quarter) and an average return of 10.48% for short stocks (that decrease in value over the quarter). Using these values, we can calculate estimated returns for the long and short portfolios generated by selecting the top 20 stocks from the aggregate model.

Average long stock return = $R_b = 17.18\%$
Average short stock return = $R_s = 10.48\%$

For Top N = 20,
Accuracy of buy decision = $\alpha_b = 0.5$
Accuracy of sell decision = $\alpha_s = 0.75$

Expected long portfolio returns
$= R_b\alpha_b - R_s(1 - \alpha_b) = 3.35\% = 14.08\%$ (annualized)

Expected short portfolio returns
$= R_s\alpha_s - R_b(1 - \alpha_s) = 3.49\% = 14.7\%$ (annualized)

(Note: The annualized returns are calculated by compounding quarterly returns by a factor of 4)

N	Overall	Buy	Sell
10	70%	50%	90%
20	62.5%	50%	75%
50	59%	42%	76%
100	57%	42%	72%
1000	57.65%	46.2%	69.1%

Table 3. Accuracy of aggregate model after picking top N predictions sorted by aggregate F1 score

Thus, we can see that the aggregate model outperforms historical market returns, which is around 10% [10]. These are just rough calculations based on the average stock returns from the test dataset and our estimated accuracy. We can conduct more thorough back-tests on the test data by using actual predictions made by the aggregate model in the future.

6. Conclusions

The goal of our project was to apply deep learning techniques to the problem of valuing an equity and making a decision to buy or sell a given equity based on fundamental financial factors alone. Our models provide promising results and need to be tested further, but we think we were successful in achieving our primary goal. While our models don't perform as well as other similar projects that have been cited, we think our approach to the problem was completely different and may provide better results in the future.

We aim to continue training the MLP models for longer training periods and add many more models to our aggregated model. We also were not able to experiment with Recurrent Neural Network architectures, which might be better suited for the problem since the data we have is sequential. But, implementing RNNs would require a bigger dataset since we don't have long enough sequences of data for companies currently. We are constantly looking for more sources of data and aim to try out other network architectures in the future.

We were able to improve the performance of our models by a significant amount just by doing minimal data cleaning. Gathering more data and extensively cleaning the data we have currently will likely improve performance further. One signficant step towards increasing the size of the dataset would be to train simpler Machine Learning models to perform data imputation instead of just dropping the missing values will essentially double the size of our dataset at most. Furthermore, the original SEC dataset had millions of data points that we had to drop due to missing values. Applying data imputation and other techniques to this dataset would increase the dataset size by an order of magnitude, allowing the networks to train better.

7. Contributions

All members of the group were very involved in this project. Ritik Goyal built the scripts to scrape, clean and organize the data for both the SEC and the FMP datasets. He also formulated the Random Forest inspired aggregate model. Eliot Kim worked on some of the earlier models, such as logistic regression. Param Gandhi worked on generating the 14 MLP models used in the final aggregate model. All members extensively tested and updated the different models used in the experiments. The project proposal, presentation and report were a team effort too.

References

[1] T. Segal, *6 surprising facts about retirement*, Apr. 2020. [Online]. Available: `https://www.investopedia.com/articles/retirement/110116/6-surprising-facts-about-retirement.asp`.

[2] R. Surz, *U.s. stock market is biggest & most expensive in world, but u.s. economy is not the most productive*, Apr. 2018. [Online]. Available: https://www.nasdaq.com/articles/us-stock-market-biggest-most-expensive-world-us-economy-not-most-productive-2018-04-02.

[3] I. Staff, *Warren buffett: How he does it*, Jan. 2020. [Online]. Available: https://www.investopedia.com/articles/01/071801.asp.

[4] T. Segal, *Fundamental analysis*, Mar. 2020. [Online]. Available: https://www.investopedia.com/terms/f/fundamentalanalysis.asp.

[5] J. Liu, F. Chao, Y.-C. Lin, and C.-M. Lin, "Stock prices prediction using deep learning models," *Journal of LaTeX Class Files*, vol. 14, no. 8, Aug. 2015. [Online]. Available: https://arxiv.org/pdf/1909.12227.pdf.

[6] A. Hayes, *Technical analysis*, Apr. 2020. [Online]. Available: https://www.investopedia.com/terms/t/technicalanalysis.asp.

[7] H. M, G. E.A., V. K. Menon, and S. K.P., "Nse stock market prediction using deep-learning models," *Procedia Computer Science*, vol. 132, pp. 1351–1362, 2018. DOI: 10.1016/j.procs.2018.05.050. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050918307828.

[8] *F1 score*, Apr. 2020. [Online]. Available: https://en.wikipedia.org/wiki/F1_score.

[9] J. Chen, *Understanding long/short equity*, Jan. 2020. [Online]. Available: https://www.investopedia.com/terms/l/long-shortequity.asp.

[10] J. Maverick, *What is the average annual return for the sp 500?* Apr. 2020. [Online]. Available: https://www.investopedia.com/ask/answers/042415/what-average-annual-return-sp-500.asp.