CS449 Final Project Report - Team JSSF

1 Group Members

Felipe Dimantas (fpd0719), Jack Daenzer (jtd4183), Sparsh Gautam (sgh1827), Stephen Savas (srs5143)

2 Abstract

Our final project seeks to use an LSTM, an MLP, and a Transformer model trained on stock data from all of the S&P 500 companies dataset to classify and predict different levels of stock changes (i.e. 0-5% growth, 5-10% loss, etc). The idea is to compare different models to see which works best in this scenario and attempt to create reasonable predictions.

3 Introduction

Classifying increases and decreases in stock prices is both an important part of trading stocks and can potentially be very profitable. At the least, being able to predict big downswings in securities can prevent significant losses in one's portfolio. We will use a combination of datasets, including historical stock prices, earnings reports, predictors, and other price indicators to train our models to input price indicators and output whether a price swing is on the horizon and in which direction.

4 Dataset

Our primary dataset is based on stock data from all of the S&P 500 companies. This data tracks the daily open, close, high, low, and volume for each stock dating from 01/01/2000 to 01/01/2023. This is all pulled from the Yahoo Finance API [2].

We also want to explore the use of fundamental data such as company specific information on earnings and balance sheets. This information comes from the Wharton Research Data Services stock data [4], which include quarterly fundamentals for every S&P 500 company in our time frame. We use the AI4Finance-Foundation repository to pull the specific data we need from WRDS. Given our goal of predicting changes in stock price, our data measures the changes in these financials, as opposed to standalone financial data.

We combined these datasets to get the stock price information on the date of each quarterly financial data release as found in the WRDS dataset. We then calculated the quarterly return for each company, and classified that return into the 0-25th, 25-50th, 50-75th, and 75-100th percentile losses and gains. This way, instead of predicting a specific return, we focus on a more general range of outcomes. This was done as a way to both make the model generalize better and make the data more balanced.

5 Methods

In this project we aim to predict large swings in stock prices of S&P500 companies with the help of Long-Term Short-Memory (LSTM), Multilayer Perceptron (MLP), and Transformer model architectures. These are the models we aim to use for our essential goals and we are interested in seeing how one performs better relative to another. We took inspiration from using these model architectures and comparing their performance from the paper [1] referenced below.

We mainly use Binary Cross Entropy (BCE) Loss as our initial goal is to get the models to correctly predict the directions of stock swings (up, down, steady) and general classification of magnitude by percentiles (as explained in section 4a). For such a purpose BCE loss is an apt choice because it can handle multi-label classification. We also intend to use Mean Square Error (MSE) loss as a check to verify that our loss values are low enough, but the focus is on BCE loss.

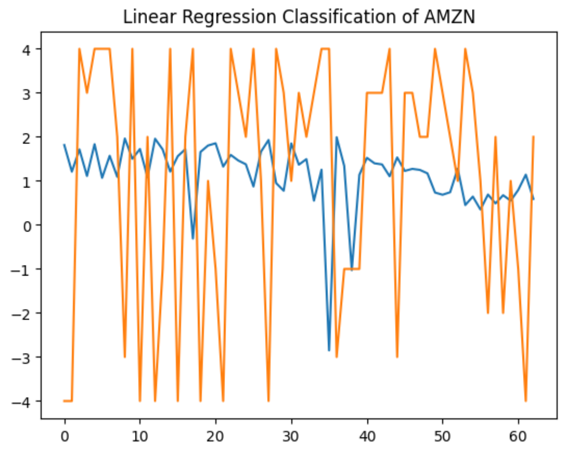
Finally we aim to evaluate our model performance using standard metrics such as accuracy and F1 score. We will also visualize model performance using learning curves; training accuracy & loss curves, validation accuracy & loss curves, and confusion matrices as these techniques allow us to gauge into areas of improvement visually (for the graphics) and also numerically guides us in the efficacy of our proposed methods.

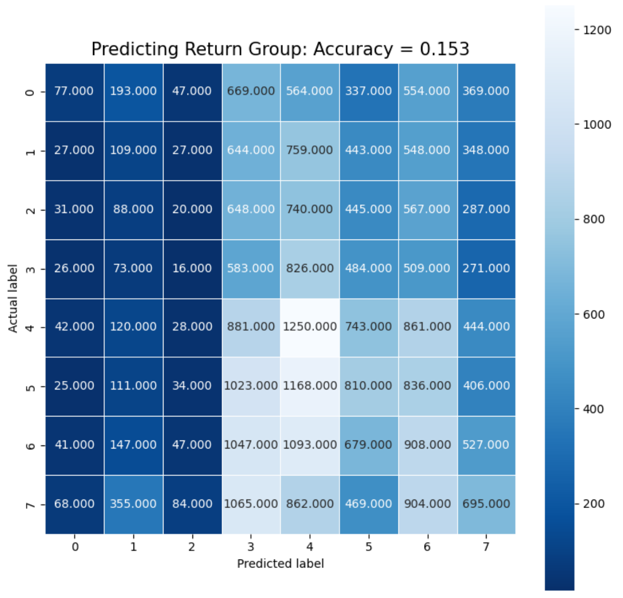
6 Goals and Discussion

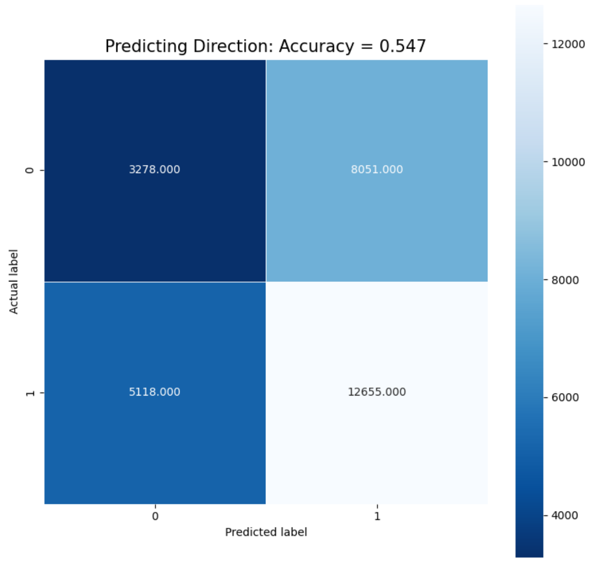
6.1 Essential Goals

Our data is put together by combining quarterly financial reports with stock data. Using the datasets mentioned in section 4, we matched quarterly data with its respective price based on a matching date. From there, we calculated each quartile of gains and losses to categorize from -4 to 4.

We use a logit model as a baseline to be sure that learning can happen on the data (and verify that learning is indeed happening). We completed this goal by dividing data into upswings and downswings, and within each classifying data by 0-25th, 25-50th, 50-75th, and 75-100th percentile gains and losses, respectively. As a baseline, we performed linear & logistic regression to gain a simple understanding of potential relationships between our data and goals. For the linear regression in particular, data was split by category ID (spcindcd) to create more generalizable plots for each sector. However, we deemed it impractical moving forward, as we would have to create too many sub-models to keep track of and there would be less data for each model. Initially, we aimed to predict returns as opposed to classify them, but as we switched to classification, we believed it would make more sense to compare even a linear regression's classification work. The linear regression itself performed poorly; however, our logistic regression had interesting results. It successfully predicted the return group over 15% of the time, better than the random guess of 12.5%, and predicted the direction over 54%, again better than the random guess. The trickiest thing for the logistic regression was the parameters, as it was difficult to select the proper class weights to balance the number of guesses for classes -4 to -1 and 1 to 4. Unfortunately, we could not strike a great balance, as the model heavily favors classifying returns from -1 to 4.





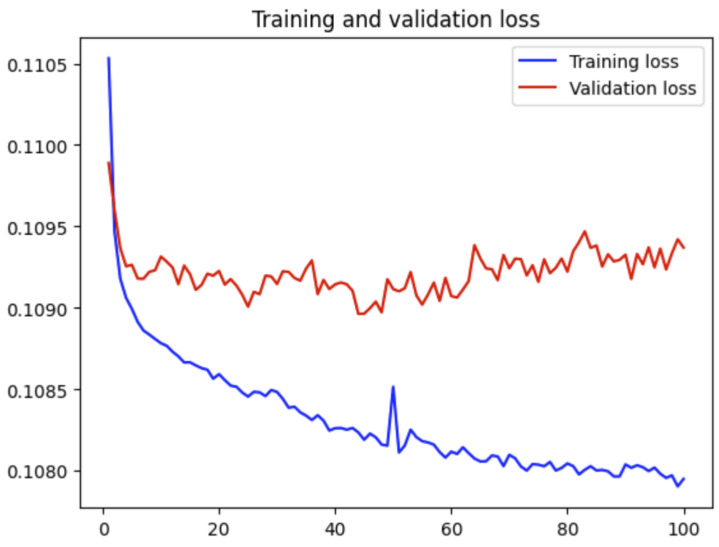


For our main models, we use Long-Short Term Memory (LSTM) and Multilayer Perceptron (MLP) to try and create more accurate predictions. These models use an 80/10/10 train-validation-test split to ensure we can generalize from our initial data to actual predictions.

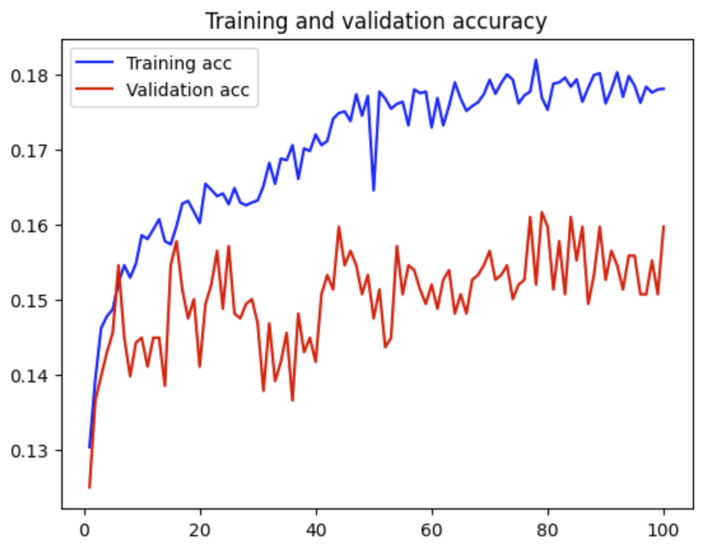
6.2 Desired Goals

Our main desired goals are to fine tune our LSTM and MLP models to be better than a guess. This can be compared in two ways. First is whether or not the model predicts the correct direction, positive or negative, where better than a guess means better than 50% accuracy. Second is the model's ability to predict the return profile, where better than a guess is better than 12.5%.

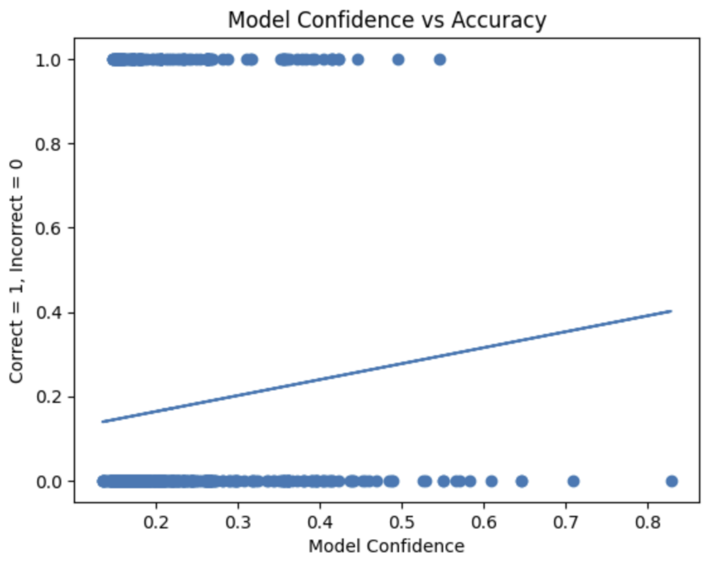
For the LSTM, after much fine tuning, we settled on the following parameters: learning rate of 0.002, 100 epochs, batch size of 64, and an Adam optimizer. Our LSTM looked back in time 10 quarters for each data point. In addition, we found that MSE loss actually edged out BCE loss for classification accuracy, due to MSE loss adapting better to the continuous nature of our classes. For our LSTM model, we also undersampled each class larger than the smallest, as imbalanced label distributions led to our model heavily favoring gains (see 6.1) and generally made accuracy results difficult to interpret. Our previous solution to this issue was to punish the model with much higher losses for failing to identify data points within the minority classes, but this just created a line between all predictions being the majority classes and all predictions being the minority classes depending on how this custom loss was weighted. With these parameters and changes, we were able to consistently achieve a test classification accuracy of 15.8%, and a test direction (gain vs loss) accuracy of 52%. Although these values are very similar to those of the logistic regression, the steps we took to balance the data makes these predictions much more credible, showing our model truly performs better than guessing. The graphs below show the training patterns of this model, as well as a Linear Regression on the model's accuracy versus its confidence. There is clear overfitting as training and validation losses and accuracies split, but this is in reality only to a small degree as the models were only able to achieve a small amount of accuracy gain and loss prevention, and is unavoidable due to the high number of features at each data point. The positive correlation between model confidence and accuracy gives us hope in this model's practical ability, as a trader informed by this model could cut "weak" predictions when making decisions.



LSTM Loss During Training

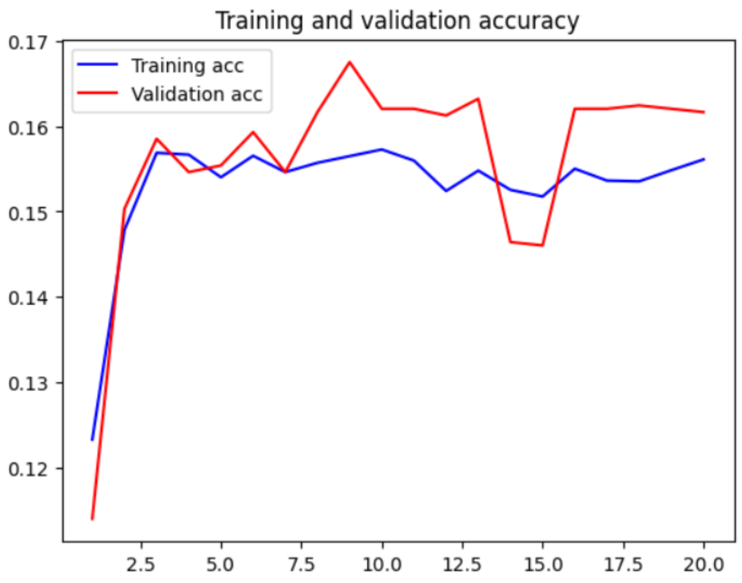


LSTM Accuracy During Training

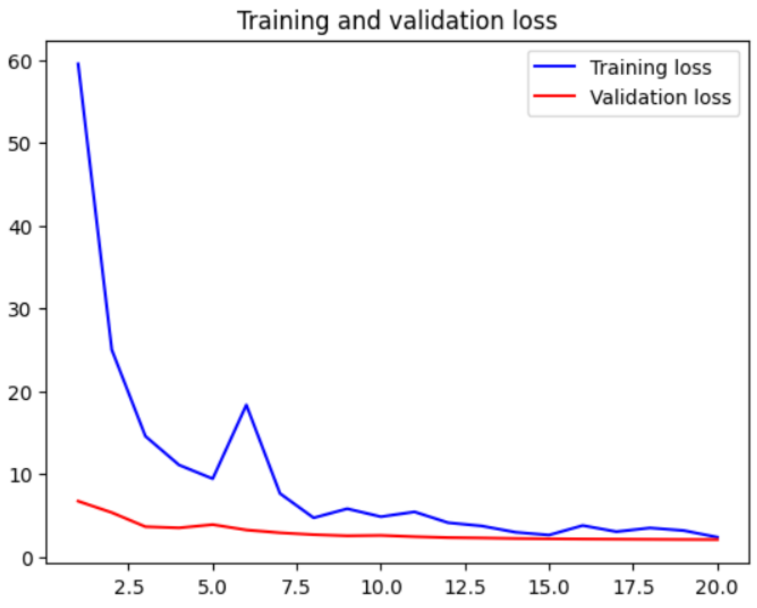


LSTM Confidence vs Accuracy

After fine tuning, we settled on a learning rate of 0.001, with Adam Optimizer which succeeded our stated thresholds, with a _ accuracy for our LSTM and 17% accuracy for our MLP when classifying the return profile. Fine tuning these models took a substantial amount of time, which ended with us creating our own specialized loss functions to heavily punish specific false signals to avoid large losses of capital. For both these models we tested all the possible hyperparameter combinations which could have yielded better accuracy results. We spent a great deal of time figuring out possible combinations of loss functions and other hyperparameters such as batch size, greater number of epochs etc but no solid accuracy results. A keen learning we got from this model training of MLP therefore was that our data may have not been as full proof enough to better gauge into the complexity of the stock market. The stock market is complex and dynamic and therefore such simplistic models (such as a basic multi-classification model) may not do justice to the overall strong prediction of degrees of swings in the stock market. Another key realization we had from this rigorous training process using MLP the issue may have been the data processing and may have not incorporated the holistic aspect of the stock market prediction. In general the MLP training process was a rigorous and long process and despite giving in the hours into tuning this model, we feel other changes would have yielded some stronger results. The plots below display the training and validation accuracy and loss plots respectively.



MLP Accuracy Plots



MLP Loss

Based on our model performance, we believe we can predict the direction of a stock with a reasonable degree of confidence. However, our exact return profile is not as strong (albeit better than a guess). As a result, we tested a trading strategy that revolves around buying stocks with positive classifications and shorting stocks with negative classifications. To compensate for risk management, we decided to set take-profit levels (sell once this price is reached) at our classification prediction with a trailing stop loss -- the point where we exit the position because we were wrong -- starting at the -1 quartile for buys, and the 1 quartile for shorts. Unfortunately, we were not able to execute our backtests due to data issues. Our models all train using a combination of price and fundamental data; however, we were not able to obtain this data for stocks outside our existing dataset, thus we could not find suitable data to feed our model over a long enough period of time to give us confidence in the level of returns.

6.3 Stretch Goals

One stretch goal is to train a reinforcement learning model to execute trades. The idea is to reward positive trades and punish negative trades -- a simple concept. In addition, we want to use our best existing model as input for the reinforcement learning to use to execute trades. However, we found that implementation of such a model would be too time consuming and technically difficult, so we decided not to pursue the project.

Our other main stretch goal is to perform social media sentiment analysis to gain a different datapoint for analyzing a stock's performance. However, given that none of our team members have experience in sentiment analysis or language processing, we would have to learn an entirely new skill set, so we dropped this goal as well. In addition, social media sentiment datasets only go back so far, so it may be difficult to use the fairly small amount of data to train a generalizable model.

7 Code and Documentation

7.1 Logit

This document was used to formalize our dataset by combining the wharton data with our stock data, in addition to classifying the return profiles by percentile. The name of the file is Logit2.ipynb
https://colab.research.google.com/drive/1FM1_BLvR2ZKZY_pEPyJP_dRq8ImtWVsP

7.2 Linear Regression

This document gives us our most basic linear regression model to try and establish a simple baseline. The name of the file is LinearRegression.ipynb
https://colab.research.google.com/drive/130e1_8ouHbOb7ZRi5HMLN5-s2oeuEwVT

7.3 Logistic Regression

Similar to our linear regression, our logistic regression provides another simple basis of comparison to evaluate our more complicated target models. The name of the file is LogisticRegression.ipynb
https://colab.research.google.com/drive/1lYsVF-PLMhAow9d3HaipzdAQfd8c75ft

7.4 LSTM

Our LSTM model is our first complex model that aims to use more historical data than simply our most recent inputs. The name of the file is LSTM_Real.ipynb
https://colab.research.google.com/drive/1OhcUQMlyeGJRL5GFHF--EQkz6ESXaV4S

7.5 MLP

The MLP model is our next complex model that aims to take a more detailed approach to classification than our regressions while still keeping the input data small. The name of the file is MLP_Model.ipynb
https://colab.research.google.com/drive/1ICqHzglppGBMt5eUe5rQHWGc0gYgbavf

8 Reflections

8.1 Interests

As a group, we are all very interested in modeling financial markets, so our topic was a natural fit. Particularly, going into this project, we knew that there were abundant resources that try to predict precise stock prices, but there were few that try to do what we set out to do at first: foresee big changes in price to avoid big losses (and maybe jump in on big winners). While that encouraged us to work more on the models, what made the project much more interesting and unique was experimenting with the models and parameters by actually designing them with minimal guidance and constraints. It is this trial by fire experience that helped us learn the most about how the models operate and what we can do to tailor the models to our needs. Unlike the homework assignments, real world data is very messy, so learning how to adapt the models to that was very rewarding.

8.2 Difficulties

We encountered a number of difficulties while doing the project.

The first hurdle was in data gathering and EDA: not all of our data had the same periodicity, and we researched and discussed extensively what the best approach would be to solve this issue. At first, we decided to interpolate quarterly data to make it daily, but there was a critical issue with that: fundamental company data does not follow a linear, polynomial, or even step function that we could use for interpolation. In addition, we realized that working with daily data might introduce unwanted noise to the model: since we are only interested in general trends of the stock and not the specific price, quarterly data should suffice, since if a fundamental change was driving the price the trend should last a period over one quarter. For these reasons, we decided to make the daily data – prices – quarterly instead.

The second main difficulty was in accounting for data imbalances. We initially thought that defining a good threshold for downswing could be accomplished by taking top and bottom percentile cutoffs. However, under this criterion, there were very few instances where the big downswing happened. Hence, the model could simply guess that there was no downswing every time, and most of the time, it would be right. We designed a custom loss function that penalized these false negatives to account for this, but we were unable to find a happy medium between all false negatives and all false positives. Although the model was more accurate than guessing, it was only because of the data imbalances. One of the solutions we used to get around this issue was to split the data into percentiles so that it would be more balanced for each return profile label. This worked to an extent, but it was still imbalanced as there were more gains than losses due to general upward market movement, especially among the S&P 500 companies we were working with. We ended up going even further by undersampling quarters with positive returns, allowing us to fully balance our data.

8.3 Moving Forward

If we were to spend another month on this project, we would look into more intensive performance testing. In particular, this would mean more robust backtests and actual live paper trading. In addition, we would hopefully spend some time exploring our stretch goals, reinforcement learning execution and online sentiment analysis. We would also look into getting more historical fundamental data than what we could get from Wharton, such as through the Bloomberg Terminal, which we tried to use at the start of the quarter but could simply not get to give us the data we needed. With this data, we could look further back when training and work with more complex market identifiers and macroeconomic information. Finally, if we received a huge sum of money to fund this project the main aspects we would spend the funds on would be high quality data acquisition, model research & development and cloud computing resources. High quality data would allow us to gauge all different sorts of trends in markets such as historical, market shocks, change of preferences etc. Furthermore, to get the best results for certain prediction problems, feasible models are crucial for the success of stocks prediction. Hiring specialist deep learning researchers in exploring the most feasible models and taming them to specific problems would allow huge project progress in this domain. Finally, cloud computing resources would enable us to be speedily and with little memory usage train models on vast amounts of datasets. This would make the environment extremely capable at handling deep learning loads.

9 Sources

[1] Yan, H., Ouyang, H. Financial Time Series Prediction Based on Deep Learning. Wireless Pers Commun 102, 683–700 (2018). https://doi.org/10.1007/s11277-017-5086-2
[2] “Yahoo Finance - API.” Yahoo! Finance, Yahoo!, https://finance.yahoo.com/.
[3] "Wharton Research Data Services." WRDS, wrds-www.wharton.upenn.edu/