# Midpoint Progress Report

## Model Implementation

### 1 Logistic Regression as Baseline Model[1]

1.1 Overview of Logistic Regression

Logistic regression can model the probabilities of classification problems. As a discriminative model, it directly computes P(c|d). For this project, we used logistic regression as our baseline model to predict the closing stock price direction of the following day.

We used y = 1 to refer to the upward movement of closing stock price and y = -1 to refer to the downward movement of closing stock price. Given the historical stock prices as inputs, we modeled the probability of the output being 1 and being -1, as P(y = 1 | x) and P(y = -1 | x), which are calculated as below:

$$P(y = 1) = \sigma(wx + b) = \frac{1}{1+exp(-(wx+b))}$$

$$P(y = -1) = 1 - \sigma(wx + b) = \frac{exp(-(wx+b))}{1+exp(-(wx+b))}$$

1.2 Summary Statistics

We took Apple as our target company and looked at the pricing data from 2021-12-29 to 2022-04-28 (84 days). Within the 84 days, there were 36 days when the stock price went up and 48 days when the price went down. Detailed statistics related to models can be found in implementation parts below.

We chose not to consider a longer time period in our approach so that we can align the time period for historical pricing data with our StockTwits data. Obtaining posts from StockTwits for a longer period may not be in the scope of this project, as it takes prohibitively long to acquire this data.

1.3 Implementation (**without** technical indicators)

After taking 'High', 'Low', 'Open' as feature values and splitting the dataset into 80% for training data and testing data, we obtained 67 instances of training data and 17 instances of testing data. By implementing the logistic regression, we got the model coefficients below:

| | 0 | 1 |
|---|---|---|
| 0 | High | [0.26218930818632746] |
| 1 | Low | [0.15304821604380683] |
| 2 | Open | [-0.47929093522655974] |

A look at the classification report shows that the overall accuracy score is around 0.53, with precision of y = -1 as 0.64 and y = 1 as 0.33.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1 | 0.64 | 0.64 | 0.64 | 11 |
| 1 | 0.33 | 0.33 | 0.33 | 6 |
| accuracy | | | 0.53 | 17 |
| macro avg | 0.48 | 0.48 | 0.48 | 17 |
| weighted avg | 0.53 | 0.53 | 0.53 | 17 |

[1] Please refer to "Stockprice_Baseline_Models.ipynb" in GitHub for the code implementation

On one hand, the accuracy score of 0.53 is merely a bit higher than a random guess; on the other hand, the precision rate for prediction of y = 1 is extremely low, indicating a poor true positive rate especially of y = 1. The above result led us to consider ways in which we could potentially improve the accuracy of our logistic regression baseline model. One approach we took was to add more technical features as x inputs, which we describe in the part below.

1.4 Implementation (with **all** technical indicators)

We generated 86 technical features from Technical Analysis library[2] and 12 date features from fastai library[3]. We added those features to the model in 3.1.2 and ran a logistic regression. The classification report is as follows.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1 | 0.58 | 0.64 | 0.61 | 11 |
| 1 | 0.20 | 0.17 | 0.18 | 6 |
| accuracy | | | 0.47 | 17 |
| macro avg | 0.39 | 0.40 | 0.40 | 17 |
| weighted avg | 0.45 | 0.47 | 0.46 | 17 |

Now, the accuracy dropped to 0.47 because we included too many features (around 100), leading to a multicollinearity problem and overfitting. To address this, we decided to select the important features based on previous literature.

1.5 Implementation (with **selected** technical indicators)

We selected 12 technical features, which have been frequently used in previous studies[4]. We also dropped some of the date features that are irrelevant in short time prediction. A sample selection of summary statistics is displayed below:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Close** | 84.000 | 168.334 | 6.831 | 150.620 | 163.192 | 168.850 | 174.130 | 182.010 |
| **High** | 84.000 | 170.458 | 6.368 | 154.120 | 165.518 | 171.175 | 175.205 | 182.940 |
| **Low** | 84.000 | 166.218 | 7.139 | 150.100 | 161.342 | 166.600 | 171.573 | 179.120 |
| **Open** | 84.000 | 168.361 | 6.895 | 150.900 | 163.757 | 168.995 | 172.868 | 182.630 |
| **Volume** | 84.000 | 91000642.857 | 20564526.445 | 59773000.000 | 77108000.000 | 89238450.000 | 97243250.000 | 179935700.000 |
| **Year** | 84.000 | 2021.964 | 0.187 | 2021.000 | 2022.000 | 2022.000 | 2022.000 | 2022.000 |
| **Month** | 84.000 | 2.845 | 2.080 | 1.000 | 2.000 | 3.000 | 4.000 | 12.000 |
| **Week** | 84.000 | 10.452 | 9.361 | 1.000 | 5.000 | 9.000 | 13.250 | 52.000 |
| **Day** | 84.000 | 15.786 | 9.029 | 1.000 | 8.000 | 15.500 | 24.000 | 31.000 |
| **Dayofweek** | 84.000 | 2.036 | 1.384 | 0.000 | 1.000 | 2.000 | 3.000 | 4.000 |
| **trend_sma_fast** | 84.000 | 169.218 | 5.138 | 159.282 | 166.183 | 169.407 | 173.193 | 177.359 |
| **trend_ema_fast** | 84.000 | 169.054 | 4.632 | 159.106 | 166.137 | 169.601 | 172.030 | 177.464 |
| **momentum_stoch_rsi_k** | 84.000 | 0.408 | 0.351 | 0.000 | 0.101 | 0.345 | 0.747 | 1.000 |
| **momentum_stoch_rsi_d** | 84.000 | 0.411 | 0.337 | 0.000 | 0.106 | 0.346 | 0.729 | 1.000 |
| **momentum_rsi** | 84.000 | 49.049 | 9.363 | 31.549 | 42.509 | 48.299 | 55.351 | 67.483 |

[2] https://github.com/bukosabino/ta

[3] https://docs.fast.ai/tabular.core.html

[4] Peng et. al. (2021), Table 1 summarized technical indicators used in recent financial prediction studies. We selected technical features that have been used in more than 3 literatures and available in the library.

Similarly, we ran a logistic regression and produced a classification report.

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| -1 | 0.70 | 0.64 | 0.67 | 11 |
| 1 | 0.43 | 0.50 | 0.46 | 6 |
| accuracy | | | 0.59 | 17 |
| macro avg | 0.56 | 0.57 | 0.56 | 17 |
| weighted avg | 0.60 | 0.59 | 0.59 | 17 |

Note that overall accuracy increased to 0.59, which is higher than the models without technical indicators. However, the precision rate of y=1 is still below 0.5.

1.6 Incorporation of StockTwits Data

We scraped StockTwits for additional data. Previously, we only had posts made in the last month for 6 companies. We decided to narrow our focus to AAPL, MSFT, and NVDA, and we rescraped the site for all posts which tag these companies dating back to 12/31/2021. For now, we consider only posts with labeled sentiment (with "bullish" and "bearish" representing positive and negative sentiment, respectively).

We added the number of positive and the number of negative posts by day as a feature in our model. We also used the selected technical features from part 1.5. Then, we ran a regression and again obtained an accuracy of 0.59 on the test set. The model's predictive power did not increase with the addition of the new features because its predictions are largely determined by a subset of the features with comparatively massive coefficients. The features *volume, volume_obv,* and *volume_adi* had coefficients which were orders of magnitude larger than the coefficients of the other features. Therefore, the values of those features (mostly) determine the model's predictions, so the StockTwits sentiments could not influence these predictions. Simply standardizing features would not address this issue for a logistic regression. However, we will have to explore feature selection and feature normalization techniques to address this behavior of our model.

## Conclusion and Next-Step Focus

We observed that feature engineering and feature selection could improve the model's performance. In addition to using Logistic Regression as a baseline model, we have started to[5]:

1. implement Random Forest to select important features more effectively,
2. feed important features into an LSTM model and implement it, since LSTM is widely used in the latest stock prediction studies, and
3. continue to explore feature selection and normalization techniques.

We also plan to begin work on Sentiment Analysis of posts from Stocktwits to label the posts which are not pre-labeled and thus expand our ability to use this dataset.

Following these approaches, we will track accuracy scores produced by different models and design an algorithm with the highest accuracy score within our capabilities.

[5] Preliminary codes of these approaches are available in our GitHub Repo. Please notice that we will continue to work on and improve these models.