MACHINE LEARNING FOR BIG PHARMA PORTFOLIOS

Dylan Curran & Sandip Das & Sunny Pamidighantam & Leah Caldwell
Department of Computer Science
Georgia Institute of Technology
Atlanta, GA 30313, USA

ABSTRACT

Pharmaceutical companies are marked by unique stock patterns, driven by long R&D cycles, regulatory shocks, and public perception. We present a unified, end-to-end pipeline that couples classical technical indicators with social-media sentiment to forecast the short-to-medium-term direction of 25 "Big Pharma" share prices. Historical price data are enriched with rolling-window features (SMA, RSI, MACD, Bollinger Bands, ADX, ...) and 1 000 Reddit posts per company, whose sentiments are quantified with VADER and augmented by post-count and "has-sentiment" flags to mitigate sparse windows. Rigorous leakage controls—most critically, a one-step shift between indicator windows and target returns—prevent look-ahead bias, while model capacity is regularized through architecture pruning and L_2 penalties to counter over-fitting on the 200–2 500 sample windows.

We benchmark four learners—logistic regression, random forest, a shallow feed-forward classifier, and a regression network—under stratified train/test splits. The random-forest classifier attains the strongest out-of-sample performance, achieving 60% accuracy, an F1 score of 0.70, and ROC–AUC ≈ 0.61, outperforming naïve baselines and demonstrating a measurable edge despite the notoriously noisy nature of the stock market. Feature-importance analysis highlights both momentum-based indicators and sentiment density as consistent decision drivers.

These results underscore (i) the value of sentiment data when properly contextualized, (ii) the importance of data-engineering safeguards in financial Machine Learning, and (iii) the competitiveness of ensemble methods relative to deeper networks on small datasets. We release our code and data-collection tools to foster reproducibility and invite future exploration of alternative information streams—e.g. FDA filings or macroeconomic releases—for pharmaceutical stock forecasting.

1 PROBLEM

1.1 MOTIVATION

Pharmaceutical companies exhibit distinctive stock behaviors, shaped by long R&D cycles, regulatory shocks, and public sentiment. Unlike fast-moving tech stocks, Big Pharma equities often respond slowly to new information, making them prime candidates for medium- to long-term trading strategies.

However, much of a pharmaceutical company's success hinges not just on fundamentals but on perception. In an era of increasing public scrutiny and skepticism towards "Big Pharma," sentiment in the media can play a pivotal role in shaping market behavior. A sudden wave of optimism or backlash on social platforms can sway retail investors and, in turn, affect short-term price movements.

With this in mind, our goal was to evaluate whether incorporating sentiment data into traditional financial indicators could improve the predictive performance of stock movement models. Specifically, we set out to create a machine learning pipeline that could forecast stock direction for 25 major pharmaceutical companies by combining technical signals with Reddit sentiment

This dual approach also allowed us to explore a secondary but crucial question in financial modeling: does alternative data—such as public sentiment—provide tangible forecasting power when engineered and integrated?

1.2 Assets and Data Collection

As mentioned above, we hypothesized that the general sentiment surrounding a company could predict whether its stock will increase or decrease, so we knew we wanted to obtain some sentiment from the internet. To do so, we created a Reddit Scraper to pull around 1000 reddit posts using Reddit's API. each post was processed using VADER sentiment analysis to generate sentiment scores, and mark which time window the post fits in to match it with other historical data.

```
def get_reddit_sentiment_praw(keywords, subreddits, time_filter="month", limit=250):
    all_sentiments = []
    total_mentions = 0

    for sub in subreddits:
        subreddit = reddit.subreddit(sub)
        query = " OR ".join(keywords)
        for post in subreddit.search(query=query, sort="new", time_filter=time_filter, limit=limit):
            text = f"{post.title} {post.selftext}"
            if any(keyword.lower() in text.lower() for keyword in keywords):
                score = analyzer.polarity_scores(text)["compound"]
                weight = max(post.score, 1)
                all_sentiments.extend([score] * weight)
                total_mentions += 1
```

Figure 1: A snippet of the code used to scrape Reddit sentiment

We aimed to use sentiment analysis to compliment existing technical indicators, such as SMA, Bollinger Bands, etc. We hypothesized that the average of these technical indicators over a fixed window would help determine the behavior of the stock AFTER this window: even in the noisiness of the Stock Market, maybe our model can help identify patterns.

```
def compute_rsi(series, window=14):
    """Compute Relative Strength Index (RSI)."""
    delta = series.diff()
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)
    avg_gain = gain.rolling(window=window, min_periods=window).mean()
    avg_loss = loss.rolling(window=window, min_periods=window).mean()
    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))

    """When the code runs, this print statement will indicate that this function has run correctly """
    #print("\n RSI Calculated Successfully \n")

    return rsi
```

Figure 2: A snippet of the code used to fetch traditional stock data

With both sentiment and technical indicators in hand, we established a solid baseline dataset. This included standard stock features like SMA, RSI, and ADX, alongside sentiment scores derived from Reddit posts. The final step in our data pipeline involved applying feature engineering techniques to derive additional attributes that we believed could improve model performance.

One thing we noted is that, sometimes, our reddit sentiment scores would yield zeroes not because of neutral sentiment, but simply because there was no data to collect. We wanted a way for our model to differentiate between the two cases, so we introduced a Has_Sentiment feature, which took on a 1 if the scraper was successful in obtaining data, and a 0 otherwise.

Overall, this gave us 15 data points. We used...

9 Vanilla Stock Indicators: (SMA, RSI, MACD, MACD Signal, Bollinger Upper Band, Bollinger Lower Band, OBV, ATX, ATR)

6 Sentiment Indicators: (Total Sentiment, Average Sentiment, Std Deviation, Max Sentiment, Number of Posts, Has_Sentiment)

1.3 Features and Input Length

For each data point, we either took the value as a floating point number, OR took its average value over the window to get a single numerical value. This resulted in each feature being represented by a single, floating-point number. Thus, each feature only had one value, and each input had the above 15 total features.

1.4 Data Quality, Limitations, and Biases

In terms of the validity of our data, we remained confident that we were collecting clean data. We obtained credible, accurate Vanilla-Stock Data using Yahoo-Finance and Python Functions. Then, on the sentiment side of things, we obtained Reddit Data from the official reddit API, and the Sentiment Score was calculated on a credible pretrained model. Taking this in combination, we were confident that our data was of high quality.

This does not mean that our data was without limitations and biases though. One key problem we faced was in relation to our project scope: Long-Term Trading makes it hard to do many windows when collecting data. Moreover, the Reddit API can only scrape up to 1000 posts. This may lead to model bias, which we discuss later in this paper.

2 Methodology

2.1 Machine Learning Justification

Our modeling approach was centered around machine learning due to its ability to uncover unintuitive, nonlinear relationships between variables—particularly between public sentiment and stock price behavior. In the highly volatile pharmaceutical sector, where small shifts in public perception or speculative buzz can significantly impact stock performance, we believed that traditional rule-based methods would fall short. Machine learning, by contrast, allows for the flexible combination of technical indicators and sentiment data, enabling models to detect subtle patterns that might not be immediately apparent through conventional analysis.

2.2 Machine Learning Methods

To bring robustness to our project, we trained four variants of models to help in our trading: A Classification Neural Network which simply predicted whether a stock will go ”up” or ”down;” A Regular Neural Network which predicts how much a stock goes up or down; a Random Forest Model, and a plain Logistic Regression model.

Each model was trained using stratified k-fold cross-validation to mitigate overfitting, especially considering the relatively small number of time windows per stock. We also took care to avoid data leakage by ensuring that no future information was present in the input features for any given prediction, which we discuss more in the ”Results” section.

Ultimately, our goal was not just to make accurate predictions but to determine whether sentiment data meaningfully enhanced prediction performance when used alongside traditional market indicators.

2.3 Challenges and Model Design

There are three main problems that often come up when it comes to model design and data: Cherrypicking, Data Leakage, and Overfitting.

Avoiding cherrypicking proved not too difficult. Our data scrapers obtained data from each company in consecutive windows, so it didn't "skip over" any data. The primary problems we faced came from Data Leakage and Overfitting. We cover Data Leakage in the "Results" section of the paper.

Thus, this leaves us with Overfitting. Especially when our datasets range from around 200 datapoints to 2500 (smaller than we would prefer), we knew we had to be extra careful. If our model was too complex, we faced the risk of it "memorizing" the training data, and blowing up on the output data.

Our original parameters fared too ambitious, with 3 hidden layers, going from 16 to 8 to 4 neurons. Although this may not appear like a lot for larger models, trained on a smaller dataset creates the risk of overfitting.

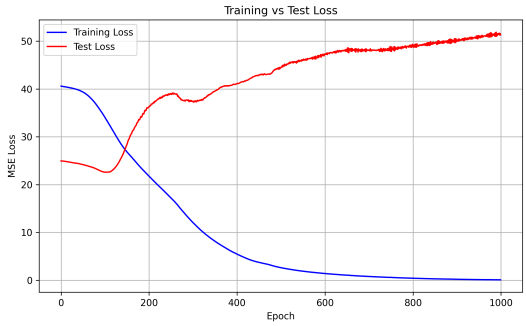


Figure 3: Model Overfitting

To fix this, we implemented two methods that helped reduce model complexity and penalize overfitting. Firstly, we reduced our model parameters to 3 hidden layers, but with 8, 4, then 2 neurons. Next, we introduced L_2 regularization into our optimizer and played around with different values of λ until our results appeared more reasonable.

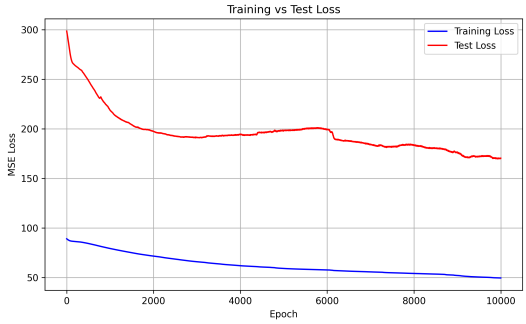


Figure 4: Regularized Loss over Epochs

Although the model has a better trend, it still performs relatively poorly. This is to be expected though, as the input data it is trained on are technical indicators from past windows. Thus, the model is assuming that the past stock trend dictates the future, not accounting for politics, sentiment, and

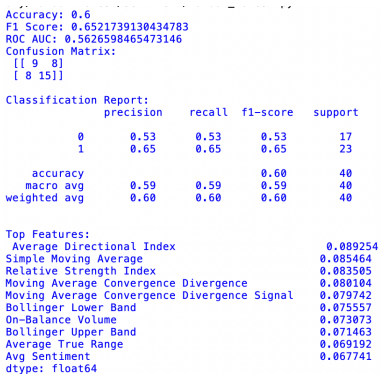
the noisiness/randomness of the stock market.

next we introduced sentiment data, which we hoped would allow our model to determine how the people will buy. We elaborate more on this in our "Results" section.

3 RESULTS

3.1 METRICS

To ensure the robustness of our evaluations, we incorporated a variety of evaluation metrics beyond simple accuracy.



```
Accuracy: 0.6
F1 Score: 0.6521739130434783
ROC AUC: 0.5626598465473146
Confusion Matrix:
 [[ 9  8]
 [ 8 15]]

Classification Report:
              precision    recall  f1-score   support

           0       0.53      0.53      0.53        17
           1       0.65      0.65      0.65        23

    accuracy                           0.60        40
   macro avg       0.59      0.59      0.59        40
weighted avg       0.60      0.60      0.60        40


Top Features:
 Average Directional Index                     0.089254
Simple Moving Average                          0.085464
Relative Strength Index                        0.083505
Moving Average Convergence Divergence          0.080104
Moving Average Convergence Divergence Signal   0.079742
Bollinger Lower Band                           0.075557
On-Balance Volume                              0.073073
Bollinger Upper Band                           0.071463
Average True Range                             0.069192
Avg Sentiment                                  0.067741
dtype: float64
```

Figure 5: Random Forest Performance Metrics

The ROC AUC score was particularly useful as it provided a baseline comparison against random guessing (AUC = 0.5). Although our model only slightly exceeded this threshold, achieving any edge in the inherently noisy and volatile stock market is a meaningful result.

Our model reached an accuracy of around 60, which is promising given the stochastic nature of stock prices. The F1 score also indicated that the model performed reasonably well despite some class imbalance—highlighting that it was able to balance precision and recall to a fair degree.

A deeper look into precision and recall per class revealed that the model struggled more when identifying negative stock movements. This is not entirely surprising, as downward movements tend to be less frequent in long-term stock data.

We also analyzed the model's top contributing features. This gave insight into which indicators the Random Forest relied on most—providing a valuable window into how sentiment and technical features interplay in the prediction process.

Finally, the model exhibited a noticeable bias toward predicting upward movements (class 1). We attribute this to the broader trend of the stock market, which generally leans toward growth over time. As a result, the model likely learned to favor upward predictions, not solely due to input features, but also due to the skewed nature of the training data.

3.2 OTHER RESULTS

As mentioned in the Methodology section, one challenge we faced directly was in relation to data leakage. Initially, we trained a Neural Network over vanilla stock data (Average SMA, RSI, etc) to predict how much a stock would go up or down, and we obtained the following results).

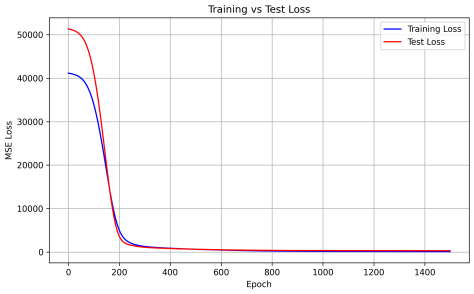


Figure 6: Loss over Training Epochs

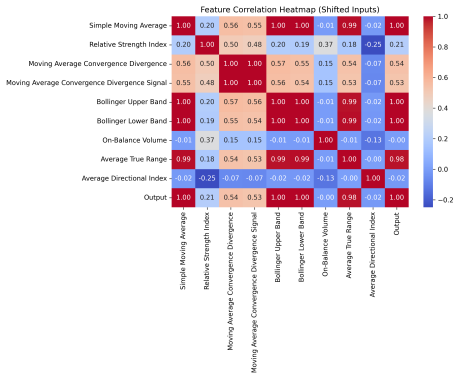


Figure 7: Correlation Between Features

Although the remarkable accuracy of the model is a red flag in its own right, the correlation heatmap highlights the main problems. There was essentially perfect correlation between SMA, Bollinger Bands, and the Outputs.

This makes sense, as both SMA and Bollinger Bands are functions of past and future prices over a window, so if they are calculated in the same window as the Output, they essentially ”give away” the output due to the way they were calculated.

To fix this, we shifted our input windows back by 1, which would ensure the output price isn't captured by these technical windows. The correction can be seen in the code below.

```
#Shift to Prevent Leakage
df[features] = df[features].shift(1)
```

Figure 8: Shifting Data to Prevent Leakage

After which, our model provided more reasonable results.

Additionally, following up on the Sentiment Analysis mentioned in the Methodology portion of this report, we think it is important to analyze the results of our model once Sentiment Data was added to the training set.

As a recap, to capture sentiment data we introduced a Reddit Scraper, which scraped 1000 posts from Reddit, and assigned them to the datapoints from the same time window. With more information, the hope was that our model would gain more predictive pattern, but it seems that the introduction of these datapoints was extremely noisy.

Figure 9: Loss over Epochs w/ Sentiment Data

The issue, we soon realized, came from the 1000 post cap. Having around 200 data points, this left room for, at most, 5 posts per data point that comprised its corresponding sentiment score. Such a small sample size introduces a lot of bias and noise. One overly enthusiastic post could lead our model to believe everyone was raving about the company, when in reality, it may have only been one person who contributed nothing to the change of the stock.

To account for this, we decided to incorporate more features to inform our data how many posts the sentiment data was derived from. Moreover, if a sentiment score came out as 0 because there were no posts, we wanted a way for our model to distinguish this from the case where the posts just had neutral sentiment. To do this, we added a feature called "Has_Sentiment" which would tell the model whether the score came from no posts, or just a lot of posts with neutral or average sentiment.

This helped improve model performance.

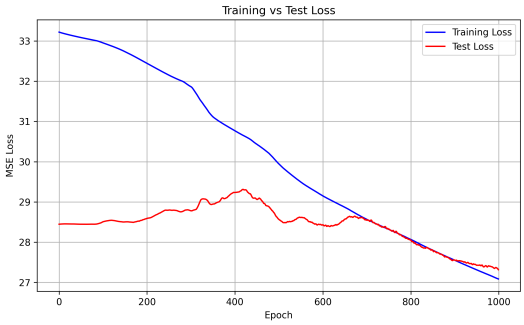


Figure 10: Loss over Epochs w/ Sentiment After Engineering

4 Conclusion

4.1 Summary

Before we reflect on the process as a whole, it is useful to recount the entire pipeline of this project to see where it has been.

This project began by scraping vanilla stock data from the Top 25 Big Pharma Companies. Such indicators include Average SMA, RSI, Bollinger Bands, etc. Then, in a separate script, we pulled stock data from a window after the technical data was obtained to determine how much the stock went up.

We then trained a Neural Network with this data, but faced challenges (overfitting, too little data, data leakage). We tried to reduce the problem to that of a classification problem.

When doing so, we trained a Classification Neural Network (1 output layer with sigmoid), a vanilla Logistic Regression model, as well as a Random Forest Model to perform the task of classification, each with their own set of challenges.

Lastly, we build a script that scraped reddit posts, and used VADER sentiment analysis to create more input features based on the general sentiment surrounding the stock in common subreddits. When we inputted it into our model, we found that it (at first) introduced a lot of noise, but with the help of feature engineering and informing our model how many posts it was dealing with, we were able to get a more realistic trend.

Ultimately, our model allows us to create a profitable trading strategy by predicting whether a stock will go up or down. Moreover, model confidence (i.e, a 0.8 prediction vs a 0.6 prediction) can inform the user how much stock they should invest.

4.2 Successes

The main successes of the project came in our identification of shortcomings, and what new things we should try to bring success to our model. For example, when we realized the task of predicting exactly how much the stock would go up or down was too challenging, reducing it to a classification problem allowed us to fine-tune the model and obtain better results. Another example came in our identification of Data Leakage and overfitting. Our solutions, as shown in the diagrams above, were able to mitigate these problems.

Another success was our model accuracy. Our Random Forest Classifier proved most successful, and even though 60 percent accuracy may not seem like a lot, given the noisiness of the stock market and the limited free data available to us, achieving this was a success.

4.3 Learning

Overall, the main takeaways from this project come from the Data Collection. On social media, there is a great meme pasted below

Figure 11: Data Science Meme

I think through this project we learned that there is a lot of truth in this image. When we train models, the data is all the model knows. It is important to collect good data, and try different kinds

of data.

One thing professor Zhao said at the beginning of the semester was something along the lines of ”Machine Learning is allowing the data to speak for itself.” We want the model to pick up on patterns in the data, and doing so may involve bringing in different data points, seeing if there is any correlation, and trying something new.

When we saw some improvements in our model after bringing in sentiment analysis, it helped reinforce this idea. Overall, our biggest takeaway is the importance of data preprocessing in the realm of Data Science.

4.4 Improvements and Future Work

Areas for improvement mainly reside in the sentiment analysis side. The main theory behind our work was the idea that Pharmaceutical stocks should mainly be driven by public sentiment, thus the lack of data through Reddit proved troublesome.

I think our model could have benefited from different sources. Incorporating perhaps combined sentiment scores from different platforms (Such as X, Instagram, etc) may have given us more data and helped reduced the noisiness of only having 1000 pieces of data.

Another avenue to consider would be different kinds of data. We mainly focused on sentiment analysis and technical indicators, but it may be interesting to explore other avenues such as political papers, macroeconomic reports, or company-specific news releases (e.g., earnings reports, FDA approvals, or M&A announcements). These sources could provide more context-driven signals that are less reactive and more predictive, potentially improving the model’s foresight and robustness.

GitHub Link (Click Me!)

or copy and paste...

https://github.com/dylanjcurran/ML-for-Big-Pharma-Stock.git