Reading Earnings Reports To Predict Long Term Stock Movements, CS 584

Cade Cermak, ccermak@stevens.edu **Charles Booth,** cbooth1@stevens.edu

1 Introduction

The main idea of this project is to perform sentiment analysis on company quarterly earnings reports, so that binary classification and prediction can be made on the movement of the stock over the next quarter. Companies produce their Quality of Earnings reports to show the progress of the company during the quarter. Analyzing these reports may lead to prediction of the company's performance over the following quarter. Leveraging this information, we can classify the stock price as movement upward or downward over the following quarter, deciding if the stock should be bought or sold. A major obstacle in this project involved obtaining all the necessary documents for the analysis, which has been successfully obtained. The data for this project contains over 13,000 rows of 10-Q documents from every S&P 500 company in the last couple of years, more specifically the Part 1, Item 2 portion of the 10-Q form which contains analysis of the company's performance during the given quarter. It was important that text mining only highlighted this section, as the vast majority of 10-Q filings have a lot of information not necessary to perform any sort of sentiment analysis. Along with sourcing the text, the necessary movement in stock price on the day the 10-Q was released and 3 months afterward is included as an important column in our dataset. Another major obstacle was in collecting the mass quantity of the stock price data. Many APIs, such as Yahoo Query, has limits on API requests that are much smaller than the necessary amount needed for this project. As a work around, a different API, Yahoo Query, was utilized. When implementing the transformers for this project, two papers were helpful, the first being the FinBERT paper [1], that created a pre-trained transformer trained on financial documents, directly from the original BERT model. A second paper that proved as a guide, created by Oussama Fadil [2], preformed sentiment analysis on 10-Q and 10-K documents as well, instead predicting hedgefund holdings rather than the stock prices of the companies themselves. The data for this project goes back to about 2020, and all of the pre-2025 documents are used for training, utilizing the other documents dated to 2025 as our validation set. Another major challenge that proved to prevent progress was the neutrality of sentiment from the Management discussion and Analysis sections, preventing the model from learning the targets. On top of that, the stock prices did not necessarily move relative to the sentiment of these documents, which is a major critical issue that will explain the rest of the results in the paper. During training, the first step was to utilize the FinBERT transformer to predict the binary targets of the stock prices, which already was limited by the problems explained previously. The next step after starting with the FinBERT transformer was to downsize the corpus to only documents that had major percentages in positivity or negativity from the sentiment found in each document, using XGBoost to predict the movement in stock prices. All of the models are evaluated by Validation accuracy, F1 score, precision, recall, and AUC.

2 Problem Formulation

The main aspects of this machine learning problem is to leveraging sentiment analysis to predict a binary class. The input of our data will be the earnings report itself. The target is the binary classification of the stock movement over the quarter. With the stock price at the date of the report as S and the stock price the day before the next quarterly report as F, the classifier will be calculated as:

$$\frac{(F-S)}{S} > 0$$

3 Methods

The main technology used in this project for sentiment analysis is the FinBERT transformer. Quality of Earnings reports can be lengthy, so using a machine learning method that can handle long-term memory is important. This will provide better sentiment analysis for the entire report compared to an RNN or a variation such as LSTM RNN. The sentiment for each of the 10-Q documents are separated into 3 classes, positive, negative, and neutral, as proposed in the FinBERT paper mentioned previously. These three classes are analyzed by the FinBERT transformer and provide a score for each category in the form of a percentage of the sentiment for each document. The first method that was used to predict the stock price data was directly tuning the transformer to learn the sentiment of the 10-Q documents and predict stock movements. The second method after this was to analyze the sentiment of each document and store it in a tabularized fashion, then using XGBoost to use these scores as a predictor for stock movements. Fine tuning the FinBERT transformer is helpful here because the MD&A sections of the documents are incredibly long, and self-attention as a feature allows the transformer to weigh each of the words such that the context is learned properly. XGBoost was introduced later to make the process of learning the stock prices faster and more efficient, which introduces ensemble learning, combining thousands of weak learners to reduce bias and variance in the model.

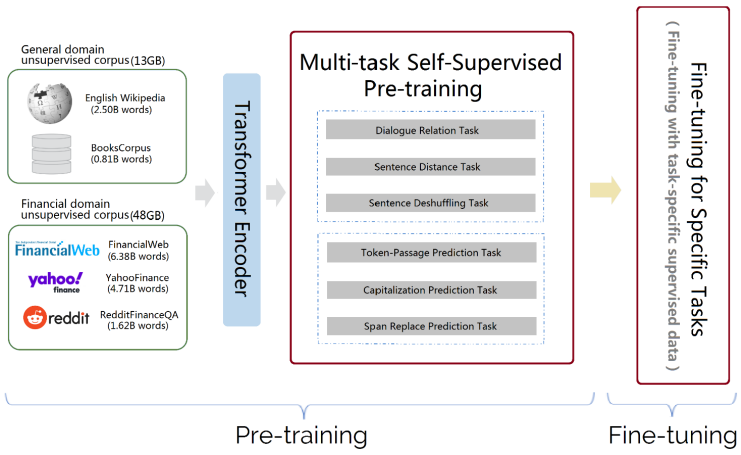

Figure 1: FinBERT architecture [3]

4 Dataset and Experiments

The main datasets that were used are the SEC's Electronic Data Gathering, Analysis, and Retrieval (EDGAR) system and Yahoo Query. The EDGAR system is a public database of filings by publicly traded companies in the United States. This is where the Quality of Earnings reports will be found for each quarter. Yahoo Query sources the stock price for each stock on a given day. When sourcing all of the quarterly reports, the S&P 500 proved to be a strong index and basis for all the necessary documents. All of the pre-processing was done using pandas and requests, where the EDGAR API was used to find the corresponding 10-Q documents for each Central Index Key or CIK of companies in the S&P 500. After all of the documents were found, the next step involved parsing all of the text in the .html docs retrieved from EDGAR, and particularly looking at the Part 1, Item 2 section, which contains valuable information regarding the analysis of the company's performance during the specified quarter. The python library beautiful soup allowed for streamlined text extraction of this section for every document, and after that was complete, all of the text found was cleaned and tokenized, removing all stop-words and non-latin characters. For the target data, the input data was first cleaned by removing any reports that were filed in the last 3 months. Then for each ticker, the stock history from the first to last report was taken from Yahoo Query. For each report in those tickers, the price on the report date, and the price on the date 3 months later (or closest to) were added to the data. Finally, the change in price was calculated for each report, and the binary target was formed.

tickers	companyName	accessionNumber	document	filingDate	url	content	price_today	price_3mo	price_change	direction
MMM	3M CO	0000066740-25-000063	mmm-20250630.htm	2025-07-18	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	153.229996	152.639999	-0.003850	0
MMM	3M CO	0000066740-25-000039	mmm-20250331.htm	2025-04-22	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	136.330002	151.199997	0.109074	1
MMM	3M CO	0000066740-24-000101	mmm-20240930.htm	2024-10-22	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	131.729996	149.119995	0.132012	1
MMM	3M CO	0000066740-24-000080	mmm-20240630.htm	2024-07-26	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	127.160004	124.750000	-0.018953	0

Figure 2: Tabularized 10-Q filing data, along with stock price delta and binary classifier

The corpus that was mined from the SEC documents provided a mass amount of data, both helping and hurting the transformer in terms of analyzing sentiment while also providing context. The first transformer experiment below had a split of $\sim$60% Class 1 (upward movement in stock price data) and $\sim$40% Class 0. This was later fixed and may provide a reason as to the performance initially.

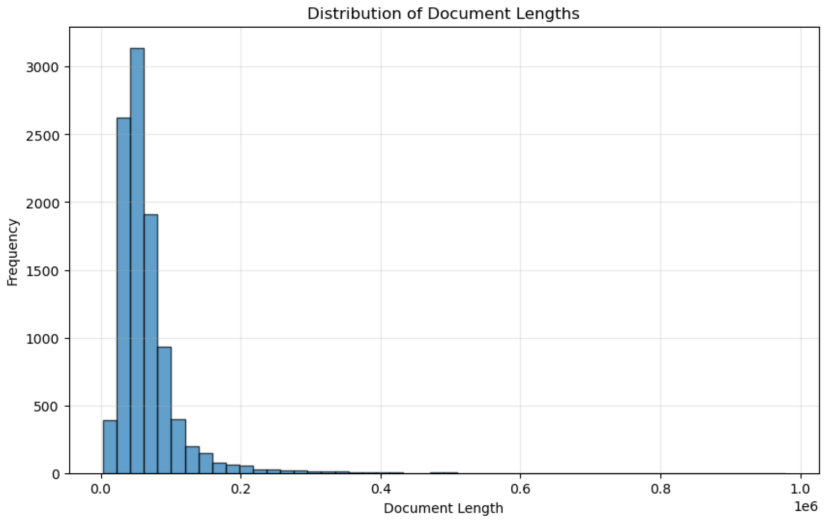

Figure 3: Document length distribution in characters

For the Stock Price Data, a more common library like Yahoo Finance was unable to be used as it could not handle the quantity of requests that were required for our dataset. Due to this, a different API, Yahoo Query, was used to better handle the needed data. In the dataset, there were several reports for each company, so to collect all of the data, Yahoo Query would obtain the stock price from the date of the first report, to the date of three months after the last report. Due to this, all reports filed within three months of collection were discarded. Once all of the prices were collected, the price on the day of the filing and price three months later was added to the dataset. The final direction counts looked as such:

Table 1: Class Distribution in the Dataset

Type	**Count**
Price Went Down	5195
Price Went Up	8127
Price Stayed The Same	17

Due to the limited quantity of stock prices that stayed the same, exclusion of those from the dataset was done to limit the target to binary classification and decrease overfitting on a third, unnecessary class.

4.1 FinBERT transformer experiments

The first experiment that tried to leverage the sentiments was done through fine tuning the FinBERT transformer. Sentiment analysis by itself was incredibly slow, so the Google Cloud T4 GPU was utilized to employ CUDA for faster token processing and training. As stated previously, the training set for the transformer was a split of documents prior to 2025, which was a majority of the dataset. The validation simply contained everything in 2025. The reason for splitting the data as such was to avoid data leakage, as the 10-Q reports may contain information regarding stock movements from previous quarters that would cause the transformer to memorize the and learn the weights rather than properly predict the target and learn why. When constructing the transformer itself, the classification, as noted in the introduction, was changed to binary. Another method to increase the speed of the model was to use truncation so that the large tokenized documents could pass into the input sequence of the transformer. Mini-Batch gradient descent was also used to provide a faster optimization. These few aspects of the training process provided a significantly faster speed-up from the vanilla FinBERT transformer sentiment classification. We defined the training arguments as follows:

- Learning rate: 0.00002
- Weight decay: 0.01
- Epochs: 30
- Training batch size: 16

Unfortunately, we did not have enough computing power to make it to 30 epochs, and the results were as good as default accuracy. Default accuracy as a metric can be defined by the accuracy of predicting one class repeatedly, so in our case, the most basic prediction would be predicting the market goes up every time, resulting in $\sim 60\%$ accuracy.

Table 2: Training and Validation Metrics Across Epochs

Epoch	Training Loss	Validation Loss	Accuracy	Precision	Recall	F1	AUC
1	0.6737	0.6770	0.5939	0.5939	1.0000	0.7452	0.4926
2	0.6681	0.6758	0.5939	0.5939	1.0000	0.7452	0.4724
3	0.6382	0.6776	0.5939	0.5939	1.0000	0.7452	0.5081
4	0.6812	0.6931	0.5939	0.5939	1.0000	0.7452	0.4850
5	0.6714	0.6761	0.5939	0.5939	1.0000	0.7452	0.4943
6	0.6693	0.6756	0.5939	0.5939	1.0000	0.7452	0.4896
7	0.6744	0.6772	0.5939	0.5939	1.0000	0.7452	0.5012

One major contributing factor to the performance of the model is likely because we used an imbalanced dataset at the time. As noted previously, around $\sim$60% of the targets had a value of 1, and the rest 0, which made it slightly difficult for the model to learn properly. On top of this, market data is not directly tied to the sentiment of the quarterly reports, the quarterly reports already overwhelmingly neutral. These results informed our process for the next step, which was to simply compute the sentiments of all the documents and retain the documents with the greatest percentages of positivity or negativity.

4.2 XGBoost experiments

As stated previously, the classification is done with XGBoost. The sentiments are already pre-calculated and fed into the model architecture, which preforms binary classification.

tickers	companyName	accessionNumber	document	filingDate	url	content	price_today	price_3mo	price_change	direction	positive	negative	neutral
MMM	3M CO	0000066740-25-000063	mmm-20250630.htm	2025-07-18	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	153.229996	152.639999	-0.003850	0	0.132291	0.303619	0.564090
MMM	3M CO	0000066740-25-000039	mmm-20250331.htm	2025-04-22	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	136.330002	151.199997	0.109074	1	0.095642	0.234331	0.670027
MMM	3M CO	0000066740-24-000101	mmm-20240930.htm	2024-10-22	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	131.729996	149.119995	0.132012	1	0.059686	0.325514	0.614801
MMM	3M CO	0000066740-24-000080	mmm-20240630.htm	2024-07-26	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	127.160004	124.750000	-0.018953	0	0.082315	0.346704	0.570981
MMM	3M CO	0000066740-24-000053	mmm-20240331.htm	2024-04-30	https://www.sec.gov/Archives/edgar/data/000006...	Item 2. Management's Discussion and Analysis o...	96.510002	126.750000	0.313335	1	0.117315	0.241281	0.641404

Figure 4: Updated dataframe with sentiments provided prior to training.

With this in mind, the training is faster, given that the main load of work was analyzing sentiment. When constructing the model, the following hyperparameters were used:

- Number of estimators: 200
- Learning rate: 0.05
- Max depth: 2
- Subsample: 0.5
- Subsample ratio of columns: 0.7
- Evaluation Metric: AUC

These hyperparamters were chosen with the intent of limiting overfitting. This is particularly evident with a max depth of 2, that is not very common in most XGBoost models as it cannot find many trends with a depth so low. On previous models with high depths, the model only learned trends present in the training data. This overfitting was never met with any increase in validation or test AUC increases. Despite this, the model could not learn from the data again. Here are the results.

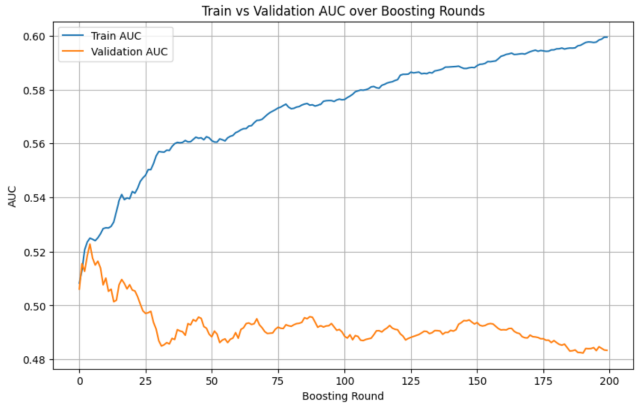

Figure 5: XGBoost AUC report

The XGBoost model obtained an AUC-ROC score of 0.4862 on the test set, which is below the baseline of 0.5 expected from random guessing. This sub-baseline AUC, combined with the classification report showing 0% precision and recall for Class 0 (Down), indicates that the model exhibits severe class imbalance issues. The model predicts Class 1 (Up) for 99% of samples (recall = 0.99), effectively ignoring the minority class entirely. This behavior results in an AUC below 0.5, as the model systematically misclassifies Class 0 samples while correctly identifying most Class 1 samples. The poor AUC performance suggests that the sentiment-derived features from FinBERT may not provide sufficient discriminative power for predicting stock price movements.

Table 3: XGBoost Classification Performance on Test Set

Class	Precision	Recall	F1-Score	Support
0	0.00	0.00	0.00	251
1	0.59	0.99	0.74	367
Accuracy		0.59		618
Macro avg	0.30	0.50	0.37	618
Weighted avg	0.35	0.59	0.44	618

Overall Metrics: Test Accuracy = 0.5890, Test AUC = 0.4862

The next experiment will involve a better split of data, using 50% of class 0 and 50% of class 1.

4.3 XGBoost on balanced data

Same architecture, data is now split evenly so that there are equal parts Class 0 and 1. Here are the results.

Table 4: XGBoost Classification Performance on Test Set

Class	Precision	Recall	F1-Score	Support
0 (Down)	0.49	0.39	0.43	251
1 (Up)	0.44	0.55	0.49	225
Accuracy			0.46	476
Macro avg	0.47	0.47	0.46	476
Weighted avg	0.47	0.46	0.46	476

Overall Metrics: Test Accuracy = 0.4622, Test AUC = 0.4430

Sub-default accuracy, began to learn the negative patterns.

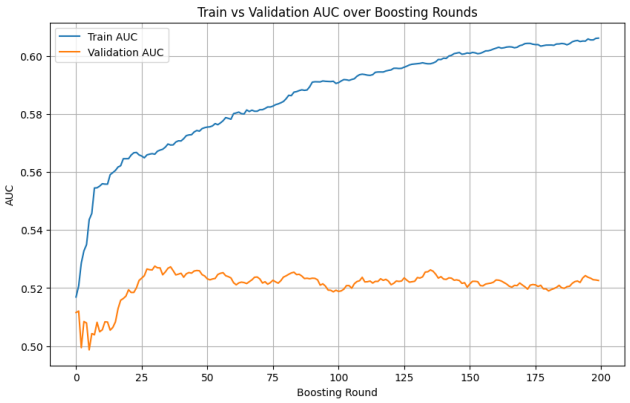

Figure 6: Balanced XGBoost AUC report

The results came back incredibly quickly which is a good sign of the optimizations of the model working properly, however it failed to learn the targets. An even split did prove to show that the model was capable of learning slightly more than previously, so this was a idea to experiment with.

Ultimately the decision to not optimize the XGBoost models with something like Optuna was made as it became clear that there was no patterns for the model to learn with any amount of optimization.

5 Conclusion

The initial problem was to provide a way of predicting stock market movements by just using the language in 10-Q reports. Unfortunately, these reports do not provide enough alone to solve this problem, as exhibited through our tests. Using XGBoost provided a faster way of doing these predictions, and the FinBERT transformer was instrumental in providing proper quarterly report sentiment analysis, which was slightly helpful. CUDA and the usage of Google Colab also provided a significant optimization to this project, which helped both of us collaborate and contribute the code to this project. Another finding through this project is that long term stock markets have uncontrollable variables, and despite the sentiment provided in the quarterly reports, these variables cannot be properly accounted for unless other metrics are provided. To improve this project, it would likely require more than just 10-Q reports, such as form 8-Ks, which talk about major events regarding businesses. These forms might provide more impact in terms of the stock price movements, creating more of a correlation. Although the results are disappointing, predicting stock movements by simply using the sentiment of 10-Q reports was unlikely to yield any significant results. Prediction of long term stock price movement is a trillion dollar market and is a problem that is tried and failed by many. The movement of stocks is influenced by much more than the earnings and reports of companies, such as many macroeconomic factors. In early 2025, the tariffs placed by the federal government was responsible for many shocks to the market and is something impossible to predict by a company's earnings reports. In the end, this project has concluded that the sentiment of earnings reports have little to no impact on the movement of the markets.

6 Project Management

The two members of the group were Charles and Cade. Charles was responsible for collecting the reports from the EDGAR system, and Cade was responsible for collecting and forming the target variable. Cade primarily worked on the development of the FinBERT transformer, along with the XGBoost model. Charles aided this by providing cleaner text data and evenly splitting the targets later on during the project.

References

[1] Dogu Araci. Financial sentiment analysis with pre-trained language models. 2019.

[2] Oussama Fadil. Predicting hedge fund holdings from 10k/q text analysis. 2021.

[3] Zhuang Liu, Degen Huang, Kaiyu Huang, Zhuang Li, and Jun Zhao. Finbert: A pre-trained financial language representation model for financial text mining. 2020.