**Introduction**

Our project focused on combining two broad categories of stock analysis, Fundamental Analysis and Technical Analysis, to learn correlations between news reports and stock market action as a means to predict next-day stock price direction given prior day news. We assumed that good news correlates to upward market movement, and bad news correlates to downward market movement. To quantify good news and bad news we used sentiment analysis and for predictive ability we used a neural network (we assume the reader has foundational knowledge of both topics).

**Methodology**

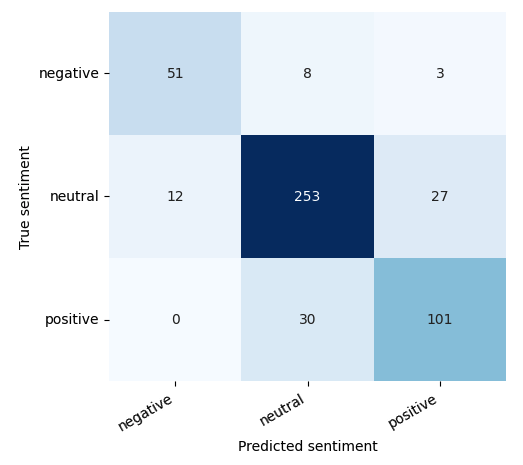
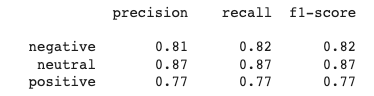
We used an ETL process to develop our predictive model. Data was procured from Kaggle, an online community of data scientists and machine learning practitioners, and analyzed through several statistical measures, including feature aggregates, density of news data per stock, date ranges, uniqueness, and sentiment distributions.

Our project used data for 50 stocks with the largest density of news data over a ten-year timeframe. As we’ll cover below, data was engineered to derive implicit stochastic relationships within time series stock and sentiment data. Price direction labels were generated using close-over-previous-close stock prices. An ensemble sentiment classifier was used to generate positive, neutral, and negative news sentiment labels for over 100000 text headlines. Aggregate sentiment features were joined with stock data on stock symbols and dates to generate neural network 24-node input vectors. Input vectors were divided into randomized datasets and used to train, validate, and test 16 neural network models to determine the best predictive model of next-day stock price directionality.

**Sentiment Classification**

To train sentiment classifiers, we used a Kaggle training set of 4845 financial headlines with known sentiment labels. Initially, we trained a TF-IDF vectorizer with Stochastic Gradient Descent (SGD) model. TFIDF vectorizer assigns weights to words based on their frequency in a document and inversely proportional to their frequency across the corpus. After hyperparameter tuning, our best model yielded a training accuracy of 97.8% and testing accuracy of 74.9%. To reduce error propagation to the neural network, we decided to use an ensemble sentiment classifier. We also trained a spaCy vectorizer with a linear SVC model. SpaCy vectorizer are based on word embeddings which are pre trained by spaCy to capture semantic information. The best model for spaCy vectorizer yielded training accuracy of 99.8% and test accuracy of 73.8%.

Additionally, we trained a transformer based model, BERT, which was created by Google. This model captures the contextual information of the text, such as word order and sentence structure. The best model yielded training accuracy of 96.2% and test accuracy of 88.5%. This shows that the BERT model performed better than the other two classifiers. Some of the information on the model can be found below.



(Fig. BERT model analysis) (Fig. BERT confusion matrix)

Looking at the train and test accuracy for the models, it can be seen that the TFIDF and spaCy vectorizer are overfitting on the data since the training accuracy is very high and test accuracy is low. This could be due to a multitude of things but one main factor is probably due to the training dataset that we used being imbalanced. The training data that we used had the distribution of the labels to be 1363 positive, 604 negative, and 2879 neutral. It can also be seen in the BERT confusion matrix that most headlines are predicted to be neutral, which makes it harder for the models to predict the correct label on new data sets.

Ensemble classifiers typically produce more accurate predictions than single model classifiers by taking the majority vote of predictions from mutually exclusive classifiers. Thus, we combined the sentiment prediction of the three models and picked the sentiment label with the most votes for each headline. The three models give the prediction from different angles, where TFIDF vectorizer is good for considering domain specific vocabulary, spaCy vectorizer is good for capturing generic semantic information, and BERT is good for capturing contextual information. By combining the results, we hoped to improve the accuracy of the final prediction of the sentiment classification.

**Data Engineering**

Raw data from Kaggle provided basic news and stock information. In particular, our news dataset provided headlines, stock symbol, and dates. Our stocks dataset provided dates, stock symbol, stock prices (open, high, low, close), and trading volume. We separated 50 stocks to train for our dataset, giving us roughly 115,000 rows of data.

We first transformed our news dataset, so each headline was classified as either a positive, neutral, or negative sentiment. We then grouped the data by date and stock symbol and aggregated the sum of positive, neutral, and negative sentiments, and then merged the resulting dataset with our stocks dataset.

We then added some more features to help extract meaningful trends in our data. For each stock, we calculated its 1, 2, 3, 4, 5, 10, 20, and 30 day % change in closing price. We also added the (% change - % change in S&P500 ) for the same 1, 2, 3, 4, 5, 10, 20, and 30 time periods, because we this shows how our stock did compared to the market average, which we deemed to be useful for our classification. We then dropped the stock symbol and date, leaving our final dataset with 24 attributes and 1 label. We then shuffled our dataset and created a [60%, 20%, 20%] split for our Training (69,460), Validation (23,154), and Testing (23,154) sets.

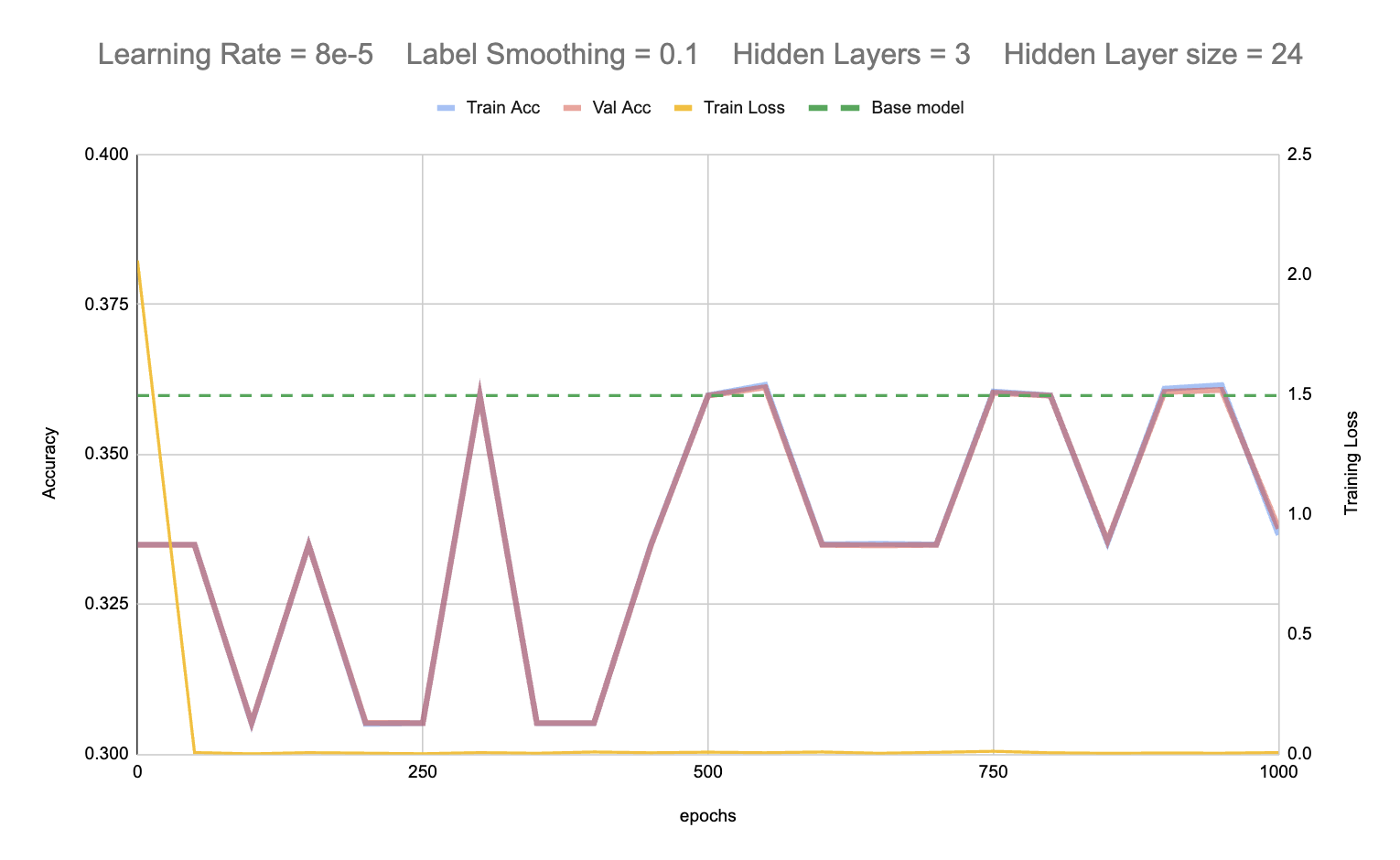
**Neural Network**

Our neural network uses a rectified linear unit (ReLU) activation, label smoothing, cross-entropy loss, and softmax classification. It has an input layer with 24 nodes, some variable number of hidden layers with variable sizes (which we optimized for in our hyperparamter tuning), with a 3 node output. 0 indicates tomorrow’s close will be less than 99% of today’s close, 1 indicates tomorrow’s close will be within 1% of today’s close, and 2 indicates tomorrow’s close will be greater than 101% of today’s close.

Optimization: We used a batch size of 1024 with the ADAM optimizer to train our modesl. We experimented with every combination of hyperparameters below, training a total of 24 models over 1000 epochs each.

Hyperparameters

* Learning Rate = {3e-5, 5e-5, 8e-5 }
* Label Smoothing = {0, 0.1 }
* Size of hidden layers = {12, 24}
* Number of hidden layers = {2, 3}

We found that our best model with the following parameters had the best validation accuracy. [ Learning rate: 8e-5. Label smoothing: 0.1. Size of hidden layers: 24, Number of hidden layers: 3 ]. This model achieved a training accuracy of 0.3616, validation accuracy of 0.3607, and testing accuracy of 0.3598. 

Unfortunately, the distribution of our labels is roughly 33.3% per category, so a model that guesses randomly is expected to have 33.3% accuracy in this prediction task, meaning our model performed slightly better than random guessing. This could be due to a wide range of reasons, but we believe the two primary reasons our model did not perform well are that the stock market is in general very difficult to predict and that we simply did not have sufficient data to extract any insights for our predictions. It is very difficult to predict the change in the price of a stock in 24 hrs with only today’s low, high, open, and close, and even with all the features we engineered, we simply did not have enough data points for any day to be able to accurately predict the change in price for the next day, so our model did not perform much better than random guessing in the end.

**Statistical Analysis**

While we wanted to classify the movement of our stocks using sentiments, we thought it would also be interesting to try to find statistical trends of stock movements based on the sentiments. For each day of movement, we counted the sentiments for the previous 5 headlines (ie {positive:1, neutral:3, negative:1}) and noted the % movement in stock price. Then, for each possible count of sentiments, we calculated the mean, median, and histogram of its stock movements:

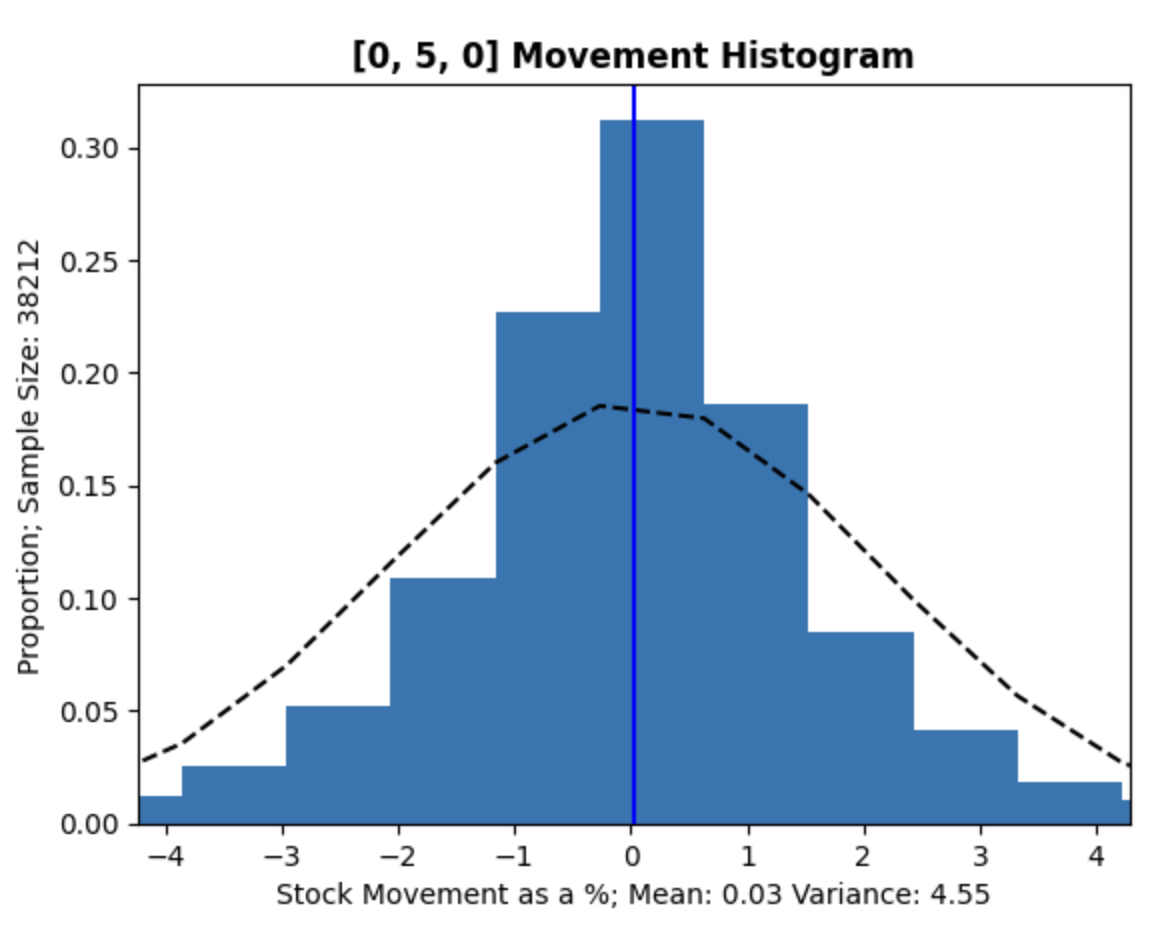
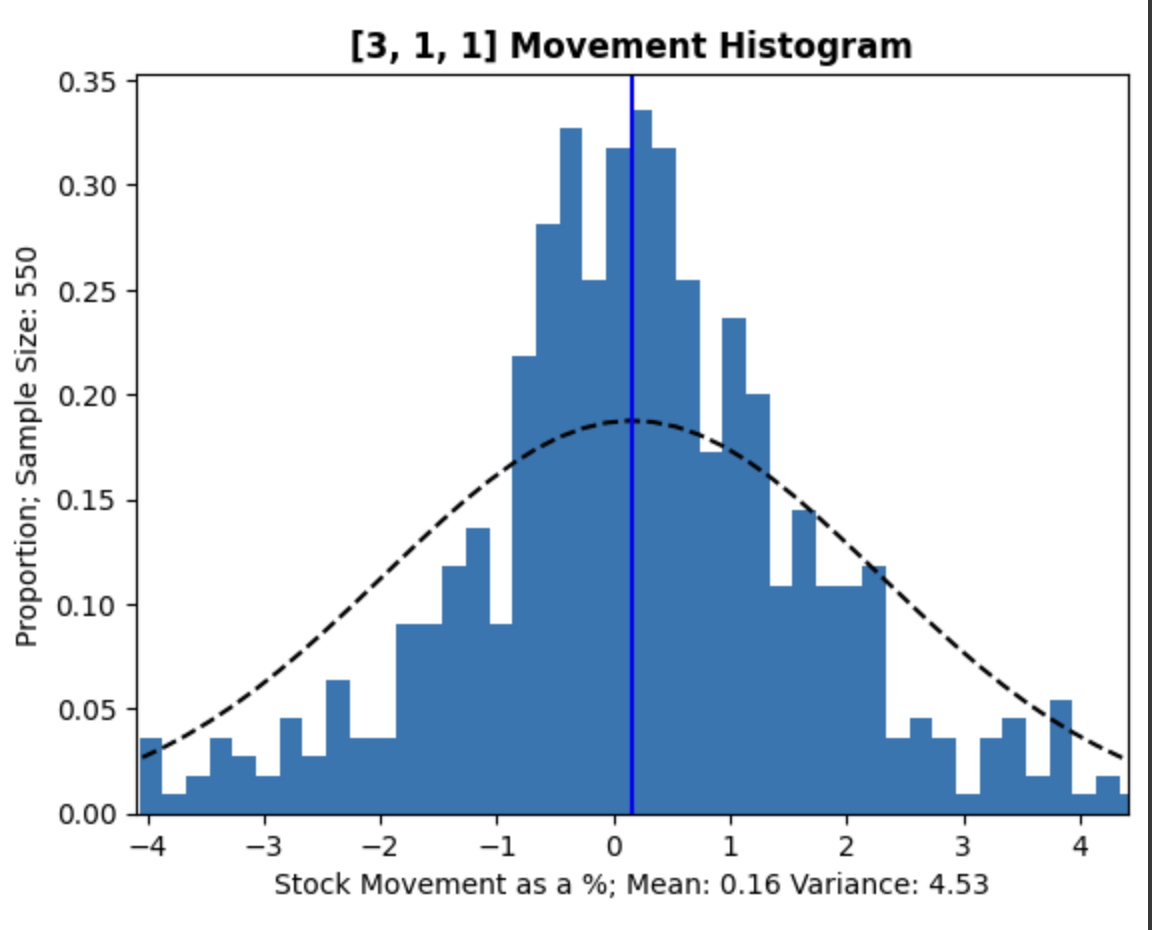
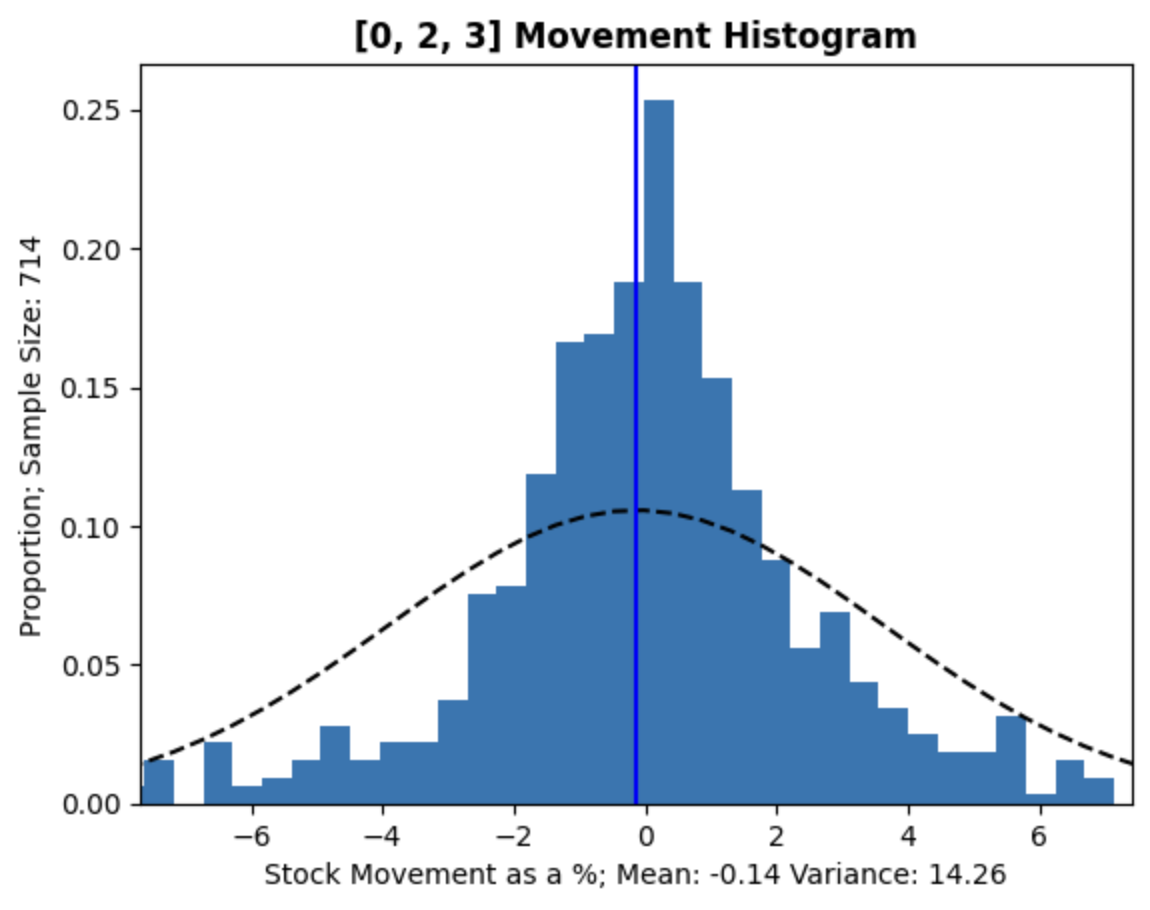


Fig.: Histograms of next-day price movement (in %) for 3 different headlines counts. The counts are formatted as [positive, neutral, negative]

One trend to note that’s not in the figure is that the majority neutral counts had the highest sample size. That is, for most days, the previous 5 headlines had at least 3 neutral headlines. It was much rarer to have a majority positive or negative count.

Generally speaking, the mean movements are generally what we’d expect with these sentiment counts (majority positives means upward average movement, neutrals means no movement, negatives means downward movement). However, it’s hard to say that this is a definitive trend, because the variance is so high compared to the means (at least 100x for counts with a decently high sample). This means that there’s a wide spread of movements for each count, instead of, say, [3,1,1] correlating strongly with a small increase in stock price. This high variance tracks with the weaker result of our neural network, and can also be because of the general volatility of the market.

**Conclusion**

Our best neural network model yielded 36% accuracy for training, validation, and testing datasets which is only 3% better than random prediction. While prediction is an inherently difficult task, the underwhelming results we achieved are in part a result of sparse sentiment data, minimal sentiment classification and data engineering, and neural network simplicity.

Sparsity of sentiment data could be addressed using data from multiple sources as opposed to a single source. While obtaining the volume of data required to learn strong correlations between sentiment and price movement was infeasible given our project's runway, a few sources to incorporate might include major news outlets, social media outlets like reddit and twitter, and company press releases. More classifiable textual data would likely have lent itself to more consistent daily stock sentiment coverage.

While this project used fine-grained sentiment classification to determine polar categories (negative, neural, positive), perhaps we could have expanded our ensemble classifier to include aspect-based, emotion detection, and intent analysis classifiers. Such an enhanced ensemble on a larger total dataset would have allowed for richer input sentiment features. Further, in addition to generating stock price percent changes over varying timeframes and comparison to the S&P, we might have also incorporated …

**Contributions:**

Richard Khillah -

Kazuma Endo -

Kia Afzali -

Miguel Roque - Sourced Data, Sentiment Classification (TDIDF), Statistical Analysis