Ava Ginsberg, Cole Mitchell, Benjamin Gregory
Prof. Lopez-Bencosme
CS 424: Introduction to Machine Learning
2025-05-14

Introduction to Machine Learning Executive Report: Exploring Stock Prediction with LSTM and Financial Sentiment Analysis

Project Overview:

Our team set out to answer the question of whether or not machine learning could be used to accurately predict stock prices. To this end, we used the Reddit API to pull posts from the popular subreddit r/wallstreetbets. We then leveraged the seventy billion parameter open source llama3 large language model to assign stock tickers for each post that referenced a stock. Using this data source, we devised two strategies for stock investment. The first strategy was a machine learning approach which involved developing and testing an LSTM model. This model used the past 30 days of stock history to determine the price of the stock on the following day. The other method was a control for evaluating the performance of the LSTM model. We decided implementing a random algorithm for buying and selling stocks was ideal as a control. The results indicate that the LSTM performs marginally better than the random algorithm on average (both algorithms lost money overall); however, this was an expected result because past stock performance does not necessarily indicate future results, meaning creating a profitable model was not feasible.

Limitations:

For this project, there were three notable portions, which each included notable limitations and restrictions, eventually stunting the final viability of the project findings. While attempting to collect a large corpus of textual data from the r/wallstreetbets sub-reddit page, network bandwidth and API restrictions forced the group to reevaluate the potential usability of reddit data. The free reddit API only allowed users to collect data as far back as a couple of days. This prevented us from collecting data spanning similar time frames unlike the financial data collected from yahoo finance and Nasdaq. Further exploration into alternatives led to us collecting data from kaggle and torrent pages, in an attempt to supplement the limited api-based collection. In the end, we were unable to assemble a large enough corpus for a viable sentiment analysis application. Once the limited reddit text corpus was compiled, regular expressions and the open source llama3 model were used to attribute stock tickers to elements of the corpus data. Unfortunately, this was not enough to provide accurate attribution to most data points, despite several attempts at prompt engineering. In turn, the limited data source and limited stock

attribution further restricted our application of sentiment analysis. The group was able to employ VADER sentiment analysis through the natural language toolkit for python, giving some indication of short term financial sentiment. This sentiment was not meaningfully used throughout the model training process for the same reasons mentioned previously. The final sentiment analysis is only used to provide some overall indication of market sentiment while simulating stock trades. In addition, the limited number of potential stocks derived from the reddit data restricts the generalizability of the final trained model. With that in mind, efforts to expand all data collection is necessary to further expand upon this project.

Data Acquisition:

For this project, we used a multitude of data sources with the aim of acquiring data that could capture closing stock prices and some indication of public sentiment over these stocks. We used the Yahoo Finance API and public data sources to gather the stock prices. Among the large public finance data sources, we used CSV formatted stock history from the Nasdaq site in addition to stock lists, lifted from Github. The accumulation of stock-related data was split into the training, validation, and test phases of the modeling process. The Nasdaq datasets are also used for the final evaluation of the LSTM model. The reason we chose the Reddit page r/wallstreetbets over other subreddits is because it hosts a large community where users actively discuss personal finance and investment strategies, largely focused on United States markets. To pull data from this subreddit, we used a free Reddit API authorization, provided through a Reddit user account. We believed with nearly 19 million users as of May 10th, 2025, discussions in r/wallstreetbets would likely provide an accurate representation of the general distribution of US financial market data (based on tickers). Using a number of Python scripts and Jupyter notebooks, we employed public libraries to facilitate an easy-to-manage connection to the Reddit API. The final Reddit discussion data was compiled into parquet format over a few files. Due to the limited scope and time constraint for this project, we supplemented our collected Reddit data with existing Reddit collections from Kaggle. These Kaggle datasets reflect the same r/wallstreetbets forum since 2020.

Sentiment Analysis:

Using the Python NLTK (Natural Language Toolkit) library and its native sentiment analysis feature, positive, neutral, negative, and composite scores are calculated and assigned for each submission. The sentiment scores provided through the basic NLTK sentiment analysis are derived through the VADER (Valence Aware Dictionary and sEntiment Reasoner) algorithm. It uses a lexicon of words and phrases with pre-assigned sentiment ratings, and also considers grammatical and syntactical cues to determine the overall sentiment of a passage. The sentiment scores were calculated using either the title, body, or both texts when available for all Reddit discussion elements. Considering that certain discussions do not include a body text, using both

body and title text for analysis was not always possible. This project recognizes that sentiment scores for short corpus documents can be individually inaccurate; however, through a large sample size, it is assumed that the cumulative scores will generalize and reflect actual market sentiment overall.

Data Normalization and Data Split:

After viewing several other machine learning projects that attempt to predict stock prices, we concluded that the closing price was the best indicator of stock performance for our project. Since stock prices can widely vary, we used sklearn's min max scaler module to normalize the data to values between 0 and 1.

We decided to partition our dataset into three different sets, training, validation, and testing. The normal guidelines for data partitioning is that the validation and test sets should come from the same distributions, while the other pairings of sets are allowed to come from different distributions. However, after extensive research into the subject (other projects on Github and Kaggle) we determined that the best split for the LSTM was sequential. This meant that the training and validation sets would comprise the first 90% of the data for each five year period of data collected for each stock ticker. The final 10% would be the test set, which the model would only be evaluated on once after the hyperparameters had been tuned. The 90% of the data was split by 85% training data and 15% validation data. The way the data was organized was by X sets which were made by passing a sliding window of 30 days up until each cutoff point (the beginning of the validation set, and the beginning of the test sets). The day after the 30 day sliding window served as the actual value for the stock which was the y counterpart set. This meant we ended up having 6 datasets at the very end: X_train, y_train, X_validation, y_validation, X_test, y_test.

LSTM Model Architecture:

We chose the long short-term memory architecture over other machine learning architectures because of its ability to to capture and retain long-term dependencies in sequential data. This model was well-suited to our problem, as the previous month's stock prices provide strong predictive signals for the stock's behavior on the target day. We decided to also add several dense layers after the LSTM layers in an attempt to have our model learn additional non-linear features in the stock data. We had one output neuron at the end which returned an estimate for the normalized stock price. It was also feasible to take a binary classification approach, where the output would simply be a prediction of whether the price will go up or down. This was never integrated because we wanted to be able to visually compare the actual test set plot versus the predicted one, which required magnitude from a regression approach. The

only requirement to run the notebook (model_notebooks/stock_prediction.ipynb) that contains the code for this model is moving the updated_tickers.txt file from the data folder in the root of the repository to the Google Collab runtime environment.

Hyperparameter Tuning:

The first hyperparameter that we changed to improve the performance of our model on the validation set was changing the optimizer from SGD to Adam. While it is impossible to pinpoint the exact reason why Adam outperformed SGD, we speculate that Adam is better suited for the complex architecture of long short-term memory models. The most effective hyperparameter that we tuned was the number of epochs. We began with only 5 epochs, and noticed from the learning curve that the training loss was decreasing gradually, but the validation loss was decreasing slowly with multiple instances of divergence (we used mse for the loss). Once we increased the number of epochs to 15 we noticed that the validation was consistently decreasing with more mild divergence. This problem is extremely hard to model so despite our best efforts at tuning the hyperparameters, there is still some overfitting evident from the large gap between the validation and test set loss.

Model Evaluation (Results):

To evaluate the model, we wrote a custom algorithm to test the LSTM model versus the random strategy. We used the same base algorithm for both strategies with slightly different logic to create an accurate measure of performance. The first key simplification for the algorithm was deciding that stocks can only be bought in one hundred dollar increments. The second simplification was deciding that once a model buys a stock it only sells it at the very last day in the sequence (or when the LSTM model predicts the price will go down to curb losses). If a stock is predicted to go down by the LSTM the selling price for the hundred dollars is reduced proportionally to the stock change. Both the LSTM and random models have the same number of opportunities to buy and sell to maintain precision. After testing both strategies over several trials, we observed that the LSTM on average made seven hundred dollars more than the random selection of buying and selling options. This method of model evaluation was added in addition to the initial analysis because in a real world setting how much money the model could potentially make is paramount to a company/customer. The code for the model evaluation can be found in model_notebooks/model_evaluation.ipynb. To reproduce our results, the test_stocks directory inside of the data directory must be added to the environment. In addition, the h5 file for the LSTM model inside of the saved_models directory must be added to Collab's runtime environment.