Team Name: NoPlan

Team Members:

- Riya Berry | USC ID: 7828642721
- Benjamin Lu | USC ID: 8406742793
- Ryan Tung | USC ID: 2887643877

Github Link: https://github.com/18rberry/560_labs_no_plan/tree/main

1) Algorithm Development

a) References

Data Source: YFinance

- YFinance: library that allows us to easily download financial data from Yahoo Finance
- For this lab, wanted to monitor stocks of the following companies (also known as tickers in this context) from 01/01/2024 to now:
    - Apple (AAPL), Microsoft (MSFT), Google (GOOGL), Tesla (TSLA), IBM (IBM), Oracle (ORCL), Amazon (AMZN)
- I scraped the following columns:
    - **Open**: First traded price of stock at the start of the time interval
    - **High:** Highest price the stock reached during that interval
    - **Low:** Lowest price the stock reached during that interval
    - **Close**: The final price traded before the interval ended
    - **Volume:** total number of shares traded during the interval
- I ran data-cleaning/pre-processing and validation:
    - check for anomalies: flag and remove negative values in the price columns
    - remove duplicate timestamps for the same stock
    - backfill null values with data from the previous day
    - sort by time (and ticker, or stock) ascending
- Then, I conducted feature engineering to create features for downstream algorithm usage
    - Short Term Moving Averages (SMA)

$$SMA = \frac{A_1 + A_2 + \ldots + A_n}{n}$$

$$\textbf{where:}$$

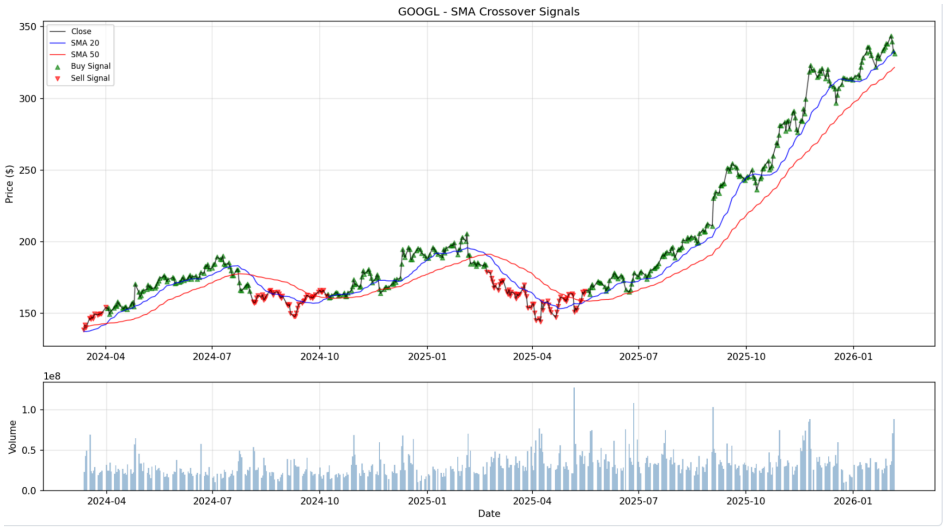
$$A = \text{Average in period } n$$

$$n = \text{Number of time periods}$$

        - used to capture recent price trends (average of last 20 days of closing prices, and then average of last 50 days to capture more long term trends)
        - help us identify the direction of a stock (upwards or downwards)
        - use a simple mean of prices over a timespan – to mitigate the impacts of random/short-term fluctuations on stock prices
        - 50 day MA-figures are widely used by investors and traders – considered to be important trading signals
        - **References: Investopedia - Moving Average Strategies**
    - Exponential Moving Averages (EMA):
        - $$EMA_t = \left[V_t \times \left(\frac{s}{1+d}\right)\right] + EMA_y \times \left[1 - \left(\frac{s}{1+d}\right)\right]$$
          where:
          $EMA_t = \text{EMA today}$
          $V_t = \text{Value today}$
          $EMA_y = \text{EMA yesterday}$
          $s = \text{Smoothing}$
          $d = \text{Number of days}$
        - exponential moving averages are more responsive to price changes than SMA (which is why some traders prefer EMA over SMA)
        - Under the hood, EMA gives more weight to recent price changes. We first calculate the SMA over a particular period, then calculate the "smoothing factor" or multiplier for weighting the EMA
        - **References: Investopedia - Moving Average Strategies**
    - Trading logic:
        - *if short term MA crosses above long term MA → signal is "Buy"*
            - (because we have upward momentum): also known as bullish crossover)
        - *if short term MA crosses below long term MA → signal is "Sell"*
            - (because we have downward momentum): also known as bearish crossover
        - *if short term EMA crosses above long term EMA → signal is "Buy"*
        - *if short term EMA crosses below long termE MA → signal is "Sell"*
    - Features for LSTM usage:
        - *daily returns:* more stationary than prices, and easier for LSTMs to learn from
        - *lag features*: close price from previous day gives LSTM recent history
        - *Volume ratio:* shows unusual trading activity
        - *Rolling volatility:* captures changing market conditions
        - *Day of Week:* we can extrapolate patterns from temporal data (i.e. stock price effects on Monday vs Friday)

b) Algorithm Implementation

First, we decided to implement a baseline model utilizing **simple moving averages (SMAs)** to be evaluated in the trading environment as well. As Professor Cho said in class, we should include baseline models based on domain knowledge (in this case, the domain knowledge that larger companies follow trends and are less volatile relatively) to enhance performance when more advanced models fail. SMAs are widely used by traders in technical analysis to help make buy and sell decisions. Once we have established this baseline, we can then explore more complex models like LSTM, ARIMA, etc for forecasting stock prices ..



Algorithm: LSTM

Rationale: I wanted to see how a Long Short-Term Memory neural network would perform on the 7 technology stocks we picked. Common sense dictates that a simple moving average would most likely have good performance given that these are large, blue-chip type companies. However, I wanted to see if the neural network's ability to learn complex, non-linear patterns in stock price movements would capture any meaningful signal and result in better profitability. Specifically, I thought that LSTMs memory gate approach (forget gates, input gates, output gates) would potentially allow us to filter out "noise" as I know ML based approaches often fail due to the noise being learned.

Additional notes: I implemented a 30% dropout rate layer to again address noise and overfitting, in hopes that only the "true" signals would remain. The output layer consisted of 3 classes for this classification problem on next-day price movement: Sell (-1), Hold (0), and Buy (1)

Training Notes:

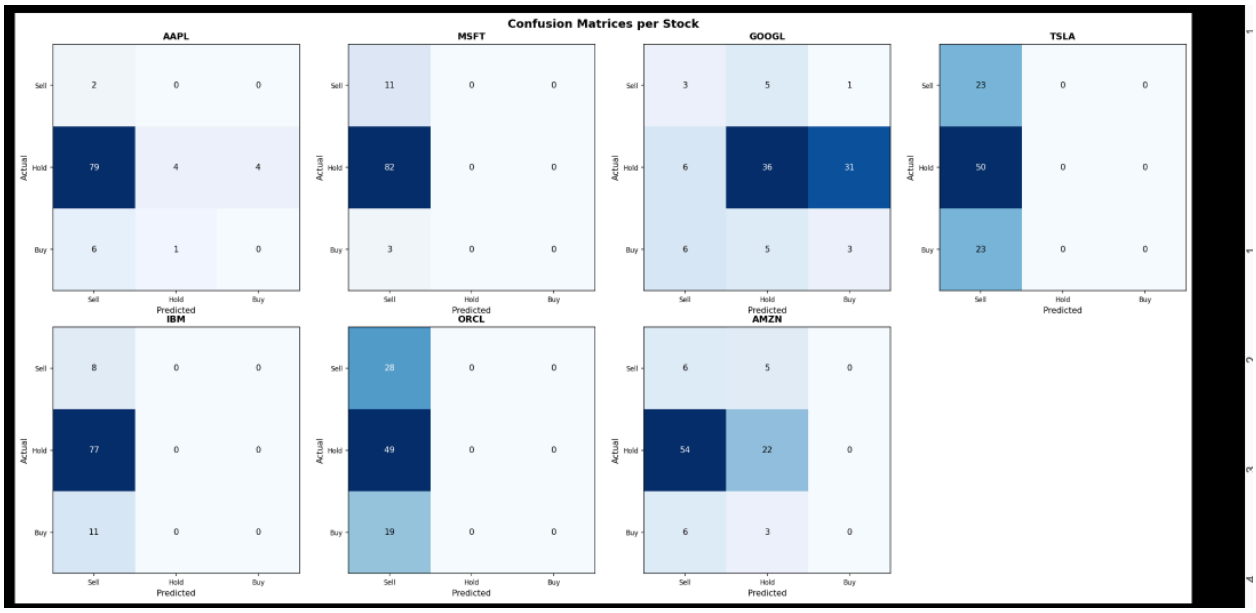
- *Training/Validation/Test Split*
    - We used a time-based split because stock data is a time-series, so random splits would not make sense. Using future data to predict the past is by definition data leakage and "cheating"
- *Early Stopping/Learning rate scheduler:*
    - We implemented this techniques to stop overfitting, ensuring we hit the right trade off where we improve accuracy without gaining too much noise into our predictions
- *Model for each stock:*
    - We trained a model for each stock, because we thought each stock/company has unique volatility patterns and market behavior due to different company fundamentals

Results (Model Performance Metrics from test set)


```
{
 "AAPL": {
  "accuracy": 0.0625,
  "precision": 0.7254789272030653,
  "recall": 0.0625,
  "f1": 0.07974067741410194
},
 "MSFT": {
  "accuracy": 0.11458333333333333,
  "precision": 0.013129340277777776,
```

```
  "recall": 0.11458333333333333,
  "f1": 0.023559190031152647
},
"GOOGL": {
  "accuracy": 0.4375,
  "precision": 0.626358695652174,
  "recall": 0.4375,
  "f1": 0.5013786764705882
},
"TSLA": {
  "accuracy": 0.23958333333333334,
  "precision": 0.05740017361111111,
  "recall": 0.23958333333333334,
  "f1": 0.09261204481792717
},
"IBM": {
  "accuracy": 0.08333333333333333,
  "precision": 0.006944444444444444,
  "recall": 0.08333333333333333,
  "f1": 0.012820512820512822
},
"ORCL": {
  "accuracy": 0.2916666666666667,
  "precision": 0.08506944444444446,
  "recall": 0.2916666666666667,
  "f1": 0.13172043010752688
},
"AMZN": {
  "accuracy": 0.2916666666666667,
  "precision": 0.5909722222222221,
  "recall": 0.2916666666666667,
  "f1": 0.34647349505840075
}
}
```

Confusion matrices from each model:



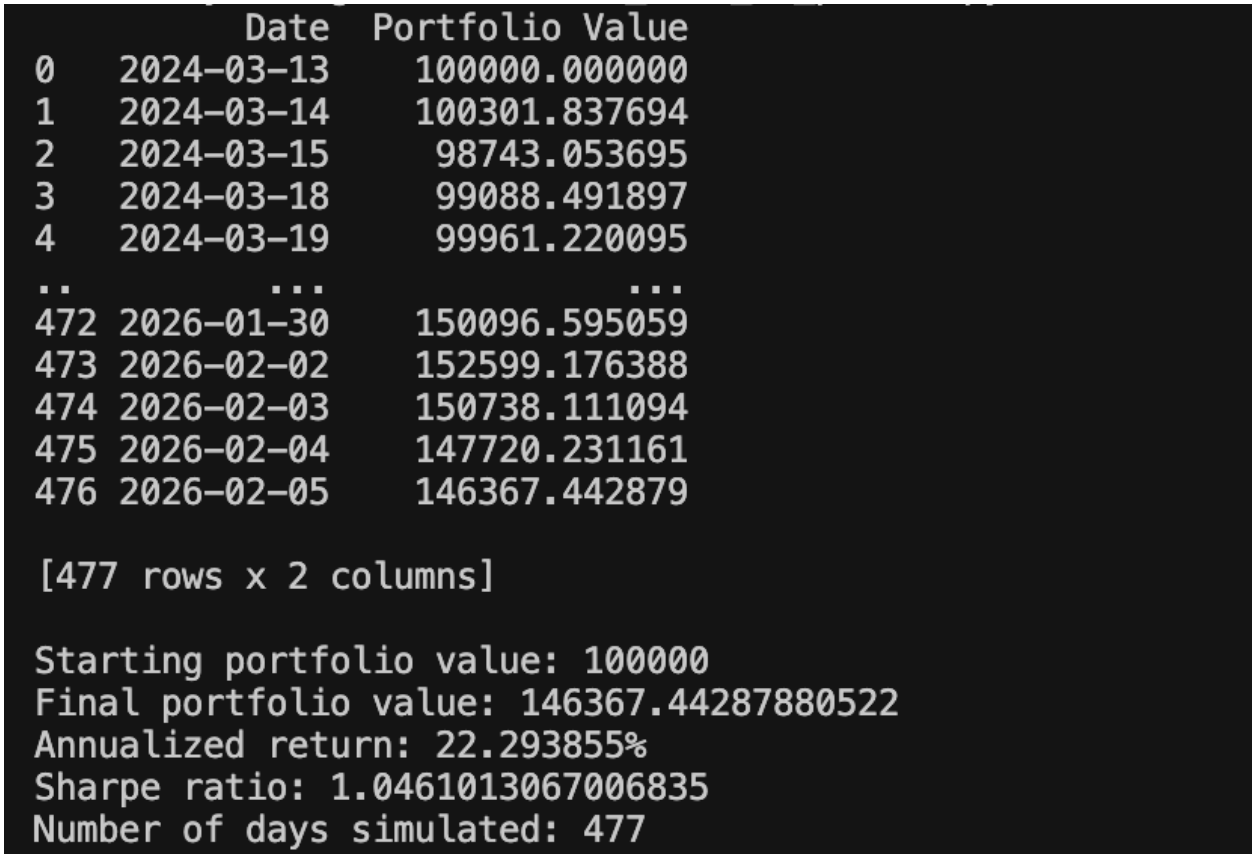
Takeaways:

From these results, the accuracy is clearly poor. Some models perform worse than random guessing (33%). Based on our data and my research, LSTM models typically need more training samples than we did for deep learning to provide benefit. Our best performer was the GOOGL model according to the accuracy metric, perhaps due to the fact that Google has been on a consistent positive trend recently. Some potential improvements are incorporating more data, implementing a hybrid approach, and also incorporating more feature engineering – sentiment analysis and other forms of data other than traditional and strict price/time/trading data.

2) Mock Trading Environment Ryan Tung

Using the signals produced in part 1.b, I created a mock trading environment that automatically bought, sold, and held stocks within a hypothetical scenario. The portfolio began with $100,000 in starting cash. It was then fed streams of daily stock data, consisting of stock close prices, as well as the BUY/SELL/HOLD signals produced by our LSTM and baseline models. For stocks designated with BUY, the algorithm automatically sold all shares that day. For all stocks designated with SELL on a given day, the algorithm allocated the portfolio's remaining cash to those stocks, proportional to the confidence level of our SMA and LSTM models.

The results of this simulation can be seen below. Our trading environment based on our baseline model traded for 477 days, culminating in a final portfolio value of $146,367. This translates approximately to a 22% annual return rate and 1.05 Sharpe ratio.

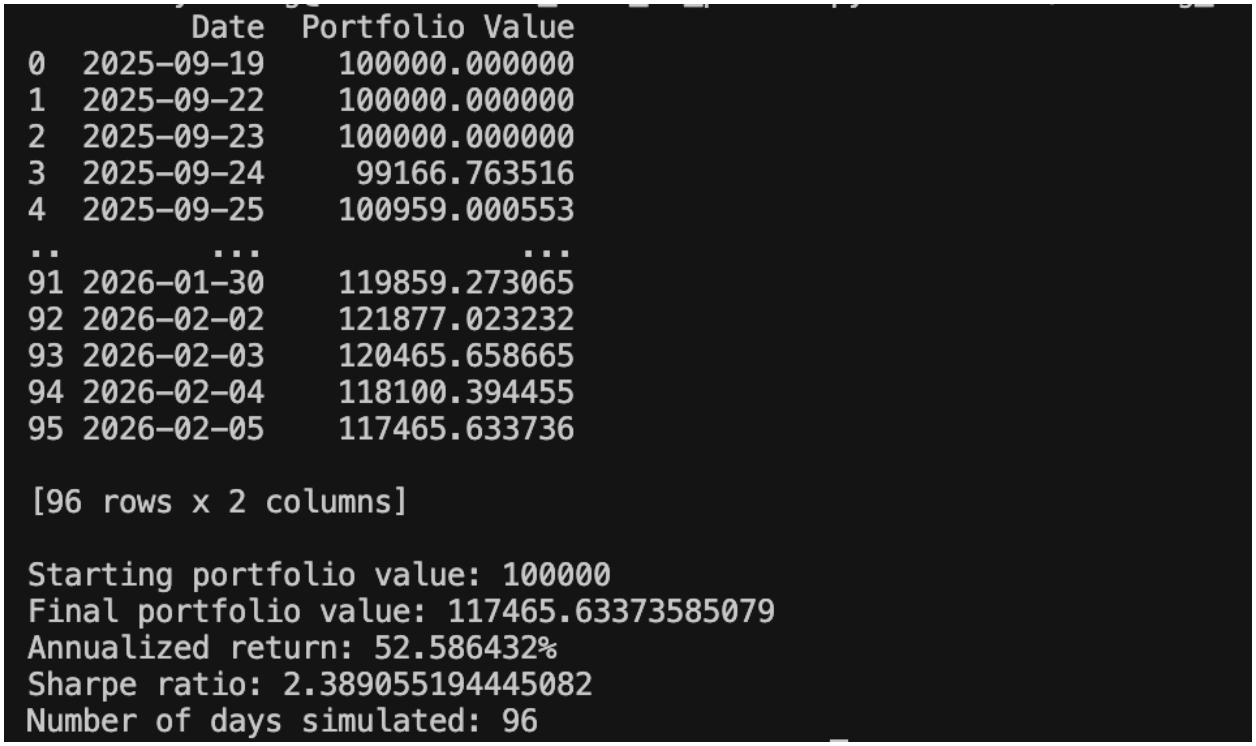


```
          Date  Portfolio Value
0   2024-03-13    100000.000000
1   2024-03-14    100301.837694
2   2024-03-15     98743.053695
3   2024-03-18     99088.491897
4   2024-03-19     99961.220095
..         ...              ...
472 2026-01-30    150096.595059
473 2026-02-02    152599.176388
474 2026-02-03    150738.111094
475 2026-02-04    147720.231161
476 2026-02-05    146367.442879

[477 rows x 2 columns]

Starting portfolio value: 100000
Final portfolio value: 146367.44287880522
Annualized return: 22.293855%
Sharpe ratio: 1.0461013067006835
Number of days simulated: 477
```

Our trading environment based on our LSTM model performed even better. Despite only trading for 96 days, the portfolio ended with a value of $117,465, which corresponds to a 53% annual return rate and 2.39 Sharpe ratio.



The code for this simulation can be found in *trading_environment.py.*

Meeting Minutes

Date	Summary
Thursday: 2/05/2026	We met from 2PM - 3PM on Thursday. We talked about how to divide up the work, and how each of the parts sort of built on each other. We also discussed which algorithms/models to use (SMA, EMA, LSTM) and why.
Friday: 2/06/2026	We met earlier on Friday to discuss more about the references Professor Cho had provided in the report. We discussed how the algorithms work and what factors are responsible for stock market trends. We also decided which companies stocks we wanted to monitor and focus on.
Saturday: 2/07/2027	We met in the afternoon to discuss compiling all of our work into a comprehensive report. We talked through what results and visualizations we wanted to show, and how we could clearly

	articulate our thought/research process for this project.