CPSC 393 Final Project Technical Report

Hayden Kaufmann, Timothy Pieschala, Andrea Fasoli, Luke Valerio

Introduction

The stock market is an incredibly volatile and difficult-to-predict system. There are countless variables that impact each stock and buying or selling actively impacts the overall behavior. Many individuals invest in surpluses in hopes of gaining a steady stream of income. We aim to make a tool that can use large amounts of data from stocks to make time-series predictions that can aid in investment decisions, increasing the chances of a positive outcome. We believe the model can be made to automatically take in recent data and retrain itself to keep it producing the most accurate results. This would allow us to build an efficient model that learns on current data and is updated as much as possible.

Analysis

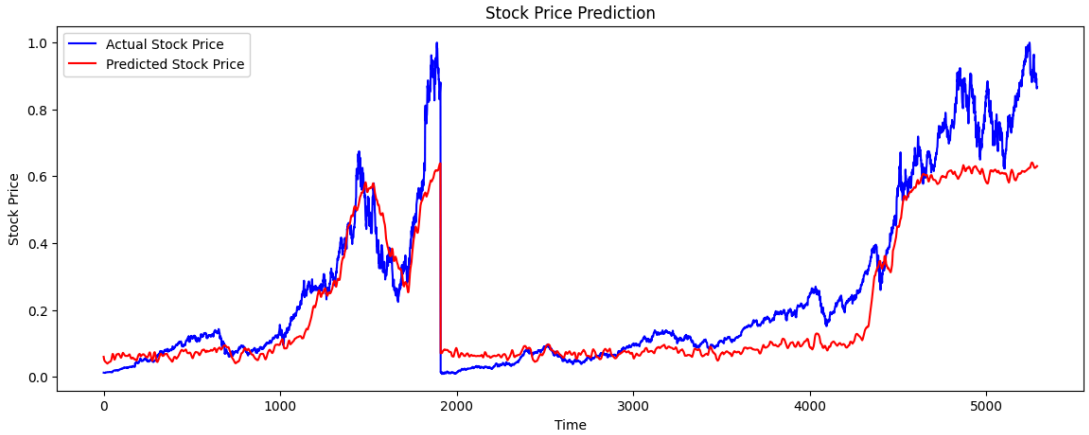
The specific data that we used was from the Yahoo Finance Python Library. Our final model drew 83 stocks across the tech industry (software and services, semiconductors and semiconductor equipment, technology hardware storage and peripherals, internet and direct marketing retail, and IT services). The given data was the price of specific stocks over a ten-year period with one sample per day. This was used to calculate various stock prediction metrics that we used as the features for our model. The following is all of the features of our dataset and a short description of each feature.

Features:
Close: Stock's closing price for the day.
SMA_10: 10-day Simple Moving Average.
EMA_10: 10-day Exponential Moving Average.
RSI: Relative Strength Index, measuring momentum.
Stoch_RSI: Stochastic RSI, normalizing RSI values.
TSI: True Strength Index, measures trend direction.
MACD: Moving Average Convergence Divergence, momentum/trend indicator.
MACD_Signal: Signal line of the MACD for crossovers.
CCI: Commodity Channel Index, measuring deviation from average price.
DPO: Detrended Price Oscillator, highlights short-term trends.
BB_High: Upper Bollinger Band, indicating overbought levels.
BB_Low: Lower Bollinger Band, indicating oversold levels.
ATR: Average True Range, measures volatility.
Ulcer_Index: Measures drawdown risk.
OBV: On-Balance Volume, tracks volume flow.
CMF: Chaikin Money Flow, tracks money inflow/outflow.

SPY_Close: Closing price of SPY (S&P 500 ETF).
VIX_Close: Closing price of the VIX (volatility index).
Year: Year of the data point.
Month: Month of the data point.
DayOfWeek: Day of the week of the data point.
Ticker_encoded: One-Hot encoding of each stock ticker.
Top Performer Label: Stocks were ranked daily by their future performance over a 5-day horizon.

Methods

For this task we decided to use a multi-layer, bidirectional LSTM model to predict whether the market is going to go up or down in the next coming days. The model includes 3 LSTM layers, then two densely connected layers. With a final dense output layer with sigmoid activation. Between every LSTM layer is a batch normalization layer. Other methods of regularization include L2, dropout, as well as early stopping in the actual training process. The goal of all this regularization is to prevent any overfitting as the stock market is usually volatile and our model needs to remain robust against these sudden changes in order to be viable. Unfortunately, this model would not be incredibly helpful in a trading environment as it being a bidirectional LSTM, it requires future data to be able to train successfully.



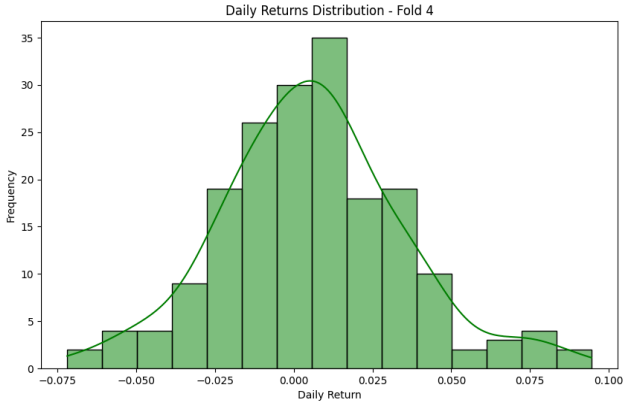
The output of our original bidirectional LSTM Model shows general understanding of the movement of stocks, however it does tend to overly adjust towards peaks specifically and valleys.

To create a more useful model that could accurately predict stocks, we moved in a different direction. We made a binary classification model using LSTMs (not bidirectional this time but similar architecture) as well as an additional transformer model to see if we could create a portfolio of stocks that grew faster than comparative benchmarks such as the S&P 500. The model predicts the Top_Performer label, which is a binary variable (1 for top performers and 0 for others) and outputs how likely a stock is going to be a “top performer” on a scale of 0-1. We used a 5-fold time-series split to make sure our model was correct while preserving temporal order. As roughly 20% of our stocks were considered “top performance”, we also used class weighting to assign higher weights to the minority class.

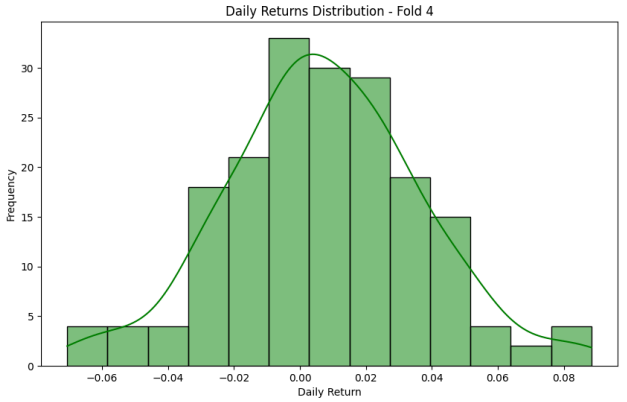
Results

Overall, our two models performed very well in predicting stocks to invest in. The following graphs show the model's performance and results.

LSTM:

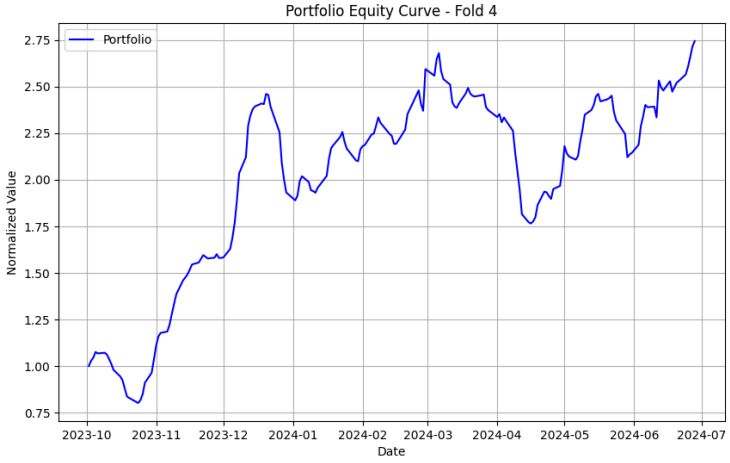


Transformer:

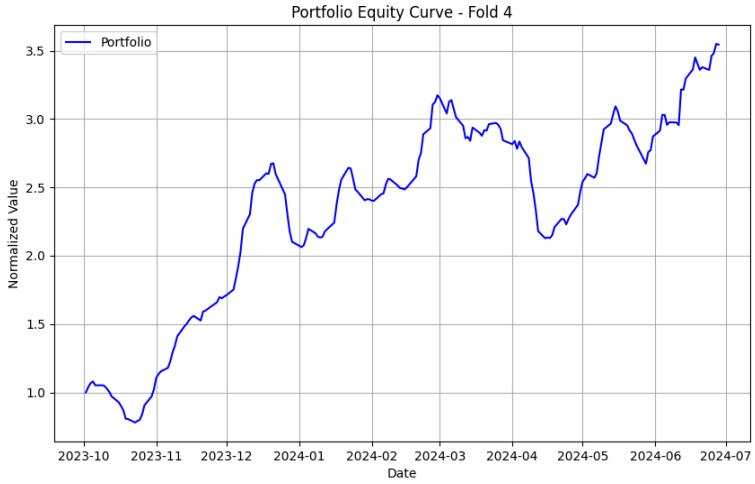


These graphs demonstrate the frequency of Daily Returns for the portfolio created by our models. The distribution follows a fairly normal distribution, however, we can see that the positive returns have more frequency than the negative returns. This indicates that on any given day, our model will provide a positive return more often than not. Over time, the small returns add up significantly, and the portfolio becomes exponentially larger. This can be seen in the next graph, which shows the portfolio's equity over time.

LSTM:



Transformer:



These graphs show that over the 9-month period the model was tested on (same stocks, but outside of the training period), the portfolio equity saw an extremely positive return.

LSTM:

Fold performance Metrics

Fold	Cumulative Return	Sharpe Ratio
1	148.60%	3.022723
2	210.56%	3.621823
3	125.80%	2.802602
4	174.48%	3.175965
Average	164.86%	3.16

Transformer:

Fold performance Metrics

Fold	Cumulative Return	Sharpe Ratio
1	276.41%	4.361688
2	277.84%	4.135574
3	262.02%	4.014116
4	254.42%	3.885298
Average	267.67%	4.10

While both models successfully had positive returns, the transformer model significantly outperformed the LSTM model. The transformer's ability to capture longer-range dependencies and more nuanced feature extraction in high-dimensional spaces caused it to yield better results.

Reflection

While we are all pleasantly surprised that we are seemingly rich, our models need significantly more backtesting to see if they can actually accurately predict high performing stocks. Our current analysis is also limited in the number of stocks we can reasonably train on. We needed to cut down our stocks as originally, 200 were attempted to run subsequently causing at least one of our computers to crash. While acquiring more stock data in this case is not particularly difficult, the more we attempt to train on, the computational resources it requires. Optimizing the model for more data would probably be the next large step as well as optimizing the actual output time of the model. Also introducing SMOTE to better represent the minority "top performers" could also improve our model.

Currently, the model predicts performance once per day and invests equally in the top 20 predicted. While this is very useful, moving to a model which predicts on a more granular level increases the useability in a real-world scenario. Being able to catch a stock as it drops mid-day due to an outside market factor isn't something our numbers only model could do so we would need to substitute for that behavior by simply checking more often.