Machine Learning on Stock Data to Inform Future Investment Choices

Rohan Gupta Rithik Pothuganti Alex Gao

Abstract

The task of this project is to use various machine learning methods to predict the probability distribution for a stock's future returns. Along the way, we explored other questions that were adjacent to the task, for which we used different machine learning methods to solve. We ended up using a variety of supervised learning methods as well as an unsupervised learning method on the past return data for multiple stocks, each of which yielded results that were not particularly robust. Our ultimate finding is that while we did try many different models and combinations of hyperparameters, the problem of finding a pattern that could predict future stock prices based on past return data is a complex one that we were not able to find a convincing solution for during this project based on our tests.

1. Introduction

In investment theory as well as betting, the fair price of an asset, such as a stock, is its expected value, or the sum of its expected values. Many in finance use the "Discounted Cash Flow" formula, which is essentially a sum of the future expected payoffs of an asset. However, many in the investment world simply use one probability and one payoff value and sum them up like so: $\sum_{t=1}^{n}\text{Pr}(\text{x}_t)\bullet\text{x}_t = \sum_{t=1}^{n}\text{EV}(\text{x}_t)$, where t represents how far in the future the payoff is expected and x represents the payoff in dollar or percentage values. The issue with this current approach is that it is discrete and oftentimes not very accurate as one probability and one payoff do not capture the full range of outcomes for a stock's payoff. The goal of this project is to use machine learning and the stock's past return and price data and other metrics to create an accurate and more continuous probability distribution that captures the full range of payoffs for a stock for however many years in the future you're hoping to predict. This helps derive a more data-driven and accurate prediction that we can use for the expected values that will determine the fair price of a financial asset, helping traders to make buy and sell decisions.

At the time of the final report, our goal has deviated slightly, as we chose related questions to answer based on what each machine learning method we used was best suited for. We chose to test models to answer four questions:

1. Can we accurately assign simple buy or sell labels?
2. Can we accurately assign one of the five labels from sell, underweight, neutral, overweight, and buy?
3. Can we predict the numerical return value?
4. Can we construct the probability distribution of future returns, our original question?

Since the Midterm Report, we have tested a combination of supervised and unsupervised learning models and have achieved varied results. Given that unsupervised learning finds a pattern in the data without labels, we will discuss the pattern our unsupervised learning method, Gaussian HMM, found and what we think it means rather than testing its accuracy as we did for the labeled supervised methods. For the Midterm Report, we tested linear and logistic regression. Through our study of existing related work, however, we found that deep learning models were more extensively used for stock price/return prediction. Thus, we incorporated deep learning models into our research, including Feed-forward Neural Networks, Recurrent Neural Network (RNN), Long Short-term Memory Networks (LSTM), and Gated Recurrent Unit Network (GRU). Furthermore, we explored other supervised learning methods including Random Forest and K-nearest Neighbors (KNN). The specific unsupervised method we used was an Hidden Markov Model with Gaussian emission probabilities. There is one section for the supervised methods and one section for the unsupervised methods that each contain results and a discussion of those results.

2. Related Work

There is existing work related to stock predictions based on analyzing a list of stock prices as time-series data. We've found five main papers that try to solve similar problems.

"Stock Price Prediction using Time Series, Econometrics, Machine Learning, and Deep Learning Models" predicts future stock prices using a combination of time series, econometric, and learning-based models (Chatterjee et al., 2021).

The study focuses on three important stocks that represent the IT, Banking, and Health sectors. The authors develop six different models and compare their performance to determine which works best in each sector. The models used include Holt-Winter Exponential Smoothing, ARIMA, Random Forest, MARS, RNN, and LSTM. The authors found that each method worked well for a specific sector — no method was clearly the best in all sectors, so it was hard for them to generalize a method that would work in other cases. Additionally, they still had relatively high RMSEs. Our approach differed in that we used only 10 tech stocks and only used the past 10 years of available data for each, while the authors of this paper used 3 stocks in 3 different sectors and tested each method on each of those stocks.

"Research on Stock Price Time Series Prediction Based on Deep Learning and Autoregressive Integrated Moving Average" uses machine learning models to predict the time-series data of stocks (Xiao & Su, 2022). The authors apply the ARIMA and LSTM models to train and predict stock prices and stock price subcorrelation. They evaluate the proposed model by several indicators and find that the ensemble model of ARIMA-LSTM outperforms other benchmark forecasting methods. The paper also discusses the challenges of stock market forecasting given the nonlinearity, volatility, and complexity of the market, and the use of machine learning and deep learning techniques to address this challenge. Our approach differed in that we didn't use the SP 500 index like they did and used other traditional financial forecasting models like the Full-Sequence Model and Single-Index Model to compare the ARIMA-LSTM ensemble to, which we did not do.

"Stock Price Prediction Using Machine Learning and Deep Learning Frameworks" uses daily stock price data at five minutes interval of time from the National Stock Exchange (NSE) of India (Sen, 2021). They then aggregate the granular data to build the forecasting framework for stock prices. The paper presents the performance of eight classification and eight regression models, including one deep learning-based approach, using data of two stocks listed in the NSE - Tata Steel and Hero Moto. The authors believe that their approach will provide several useful pieces of information to the investors in the stock market who are particularly interested in short-term investments for profit. Our approach differs in that we are aiming to find the return over a longer time period, specifically a year, while the authors of this paper are aiming to essentially do short-term day trading. They did end up using a few of the same models we did, such as KNN, Random Forests, but applied them to different goals. The challenge is that they still cannot generalize the data to more than 2 stocks since they only used 2 stocks as the data and that too in the Indian stock market.

3. Dataset and Evaluation

We used financial market databases like NASDAQ and CapitalIQ to get our daily stock price data. For the Midterm report, we only used Apple. For the Final report, we focused on more big tech stocks, specifically Apple, Amazon, AMD, Alphabet, Meta, Microsoft, Netflix, Nvidia, Qualcomm, and Tesla. We chose this based on their similarity with each other, which translates to similar stock market behavior.

The features are the past returns of the stock up until that point, and so far we've chosen the past 3-month, 6-month, 1-year, 2-year, 5-year, and 10-year returns. Because we would need at least 1 year from a specific date to have occurred already, we don't use data within the past year since it wouldn't have the 1 year forward return. Anything before a year ago would be valid data. We currently are using 10 stocks, of which each has 2,345 days worth of trading data from 2014-2023. The data can also be "featurized" to add more polynomial degrees. For instance, a degree of 2 would result in 12 features — 6 from the original features for the return windows and 6 from those original features squared. The degrees would be a hyperparameter we can test in the development set.

For the Final report, we modified our data from the Midterm to contain one more label which we call the "rating", which contains the 5 common financial industry ratings of stocks of "sell", "underweight", "neutral", "overweight", and "buy". We did this because we felt just "buy" and "sell" may lead to underfitting in classification tasks, like softmax or our neural network which are explained later. For each example, the label was a 3d tuple that contained the 1-year future return from the specific date to a date that already has occurred as the first element, a "buy" or "sell" label as the second element based on if the first element is positive ("buy") or negative ("sell"), and 1 of 5 ratings as the third element, denoted as 0 for "sell", 1 for "underweight", 2 for "neutral", 3 for "overweight", and 4 for "buy". We assigned that rating element based on the 1-year future return as follows:

- If the 1-year future return $<$ -10%, the rating is "sell"
- If the 1-year future return is between -10% and 0%, the rating is "underweight"
- If the 1-year future return is between 0% and +10%, the rating is "neutral"
- If the 1-year future return is between +10% and +20%, the rating is "overweight"
- If the 1-year future return $>$ +20%, the rating is "buy"

We chose these classes to obtain a better distribution of classes. Stocks have performed extremely well from 2013-

2022, so we used these classifications to attempt to create a more balanced dataset.

We chose to split the data into training and development sets by allocating the first 50% of examples, which are chronologically ordered, as the training set, the next 25% as the dev set, and the remaining 25% as the test set. We also decided not to shuffle the data to maintain the time-series nature of the data. Moreover, we found in our midterm report that shuffling the data causes overlap in data that is on neighboring days. E.g. if May 1st 2022 is in the train set and May 2nd 2022 is in the dev set (both have trailing return data that is almost identical).

The tables below shows the proportion of labels in each split.

Test set:

Rating	Proportion
0	0.3591
1	0.0671
2	0.0856
3	0.0752
4	0.4127

Dev set:

Rating	Proportion
0	0.0723
1	0.0745
2	0.0814
3	0.0725
4	0.6991

Train set:

Rating	Proportion
0	0.1346
1	0.0681
2	0.0835
3	0.1145
4	0.5990

We used each supervised method's natural evaluation metrics to test the accuracy of prediction. We used RMSE for Linear Regression, Recurrent Neural Networks, Random Forests, and K-Nearest Neghbors, and simple accuracy for Softmax Regression and Neural Network-based classification methods. However, for our Gaussian HMM, because it is an unsupervised learning method, it was difficult to find a way to test how accurate it would be in the real world. We include in the **Unsupervised Methods Experiments** section the top stocks we felt had the best fit with the Gaussian HMM based on how close one of the states' means was to representing that "neutral" rating, which we defined as between -5% and +5%, and how close they were in identifying the hype of tech stocks during COVID as a unique state. While this is more subjective and requires knowledge about specific financial events, it is still rooted in the idea of checking if the HMM was able to identify unique states for a stock. This was our way of evaluating how good the HMM was at finding states relevant to the use case for investors.

4. Methods

For the supervised learning models, we used Linear Regression, Softmax Regression, Simple Feed-forward Neural Network, RNN, LSTM, GRU, KNN, and Random Forests. We changed our Logistic Regression method from the Midterm report to instead use Softmax Regression since it allows multi-class classifications. For each example, we used the past 3 months, 1 year, 2 years, 5 years, and 10 years return as features. Each example has a 3d tuple: (1 year future return, buy[1]/sell[0], rating) and uses at least one of those elements as the label depending on the method. Below is a table showing the output of each method for each supervised method we used.

Method	Output	Hyperparameters
Linear Regression	Predicted Future Return 1 year return	degrees
Softmax Regression	Predicted Rating	degrees
Neural Network	Predicted Rating	degrees, dropout, activation function, learning rate
RNN	Predicted Future 1 year return	no. of layers, no. of hidden units, epochs, batch size
LSTM	Predicted Future 1 year return	no. of layers, no. of hidden units, epochs, batch size, learning rate, decay, momentum
GRU	Predicted Future 1 year return	no. of layers, no. of hidden units, epochs, batch size, learning rate, decay, momentum
KNN	Predicted Rating	neighbors
Random Forests	Predicted Rating	estimators

We tested our methods on many stocks. This allowed us to ask a new question about whether we could learn unique parameters for each stock rather than one set of parameters that would be applied to all stocks based on some aggregation of all the stocks' return data. All the supervised methods above used the aggregate dataset to learn one set of parameters to apply to all the similar tech stocks we chose to look at.

We only used one unsupervised method — Gaussian HMMs — which we used on each stock to find unique parameters for each state rather than one set of parameters for all the stocks. We only varied the number of states to fit to the dataset as our hyperparameter, and kept our number of iterations at 50 since changing it to a higher number did not change what values the HMM converged to. Our goal with the Gaussian HMMs was to find states to assign each example where the observation was the 1-year future return, which is the first element of the label in our dataset. This method yielded results that were probably the most inline with our original goal to learn a probability distribution — assuming it is Gaussian — for a stock's future returns. Given that we learn emission probabilities, where our emissions are 1 year future returns, as well as the other parameters, we can find different probability distributions for observing each 1 year future return for each state the HMM finds in our stock return data. As mentioned in the **Dataset and Evaluation** section, we had to evaluate the reasonableness ourselves of the outputs of the HMM for each stock relative to how states were assigned to each stock over the past 10 years.

5. Supervised Methods Experiments

We ran several experiments for Linear and Softmax Regressions, Simple Neural Networks, Random Forest, KNN, Simple RNN, LSTM, and GRU by varying the hyperparameters. We compiled return data for 10 popular tech stocks from 2013-2023 for our data. We used our models to predict the future 1-year return of the stocks and compared them with the actual return values, or the buy/sell classifications based on the future 1-year return.

Linear regression Results

Default values used for hyperparameters were LR=1e-2, Degree=1, iters=400. Below is the result for Apple stock. Best results came from Degree=3. Test RMSE=4.54.

Degree	Dev rmse
1	1.334
2	1.307
3	1.28
5	1.38

Softmax Regression Results

We found that accuracy for iters converged at approximately iters=200. The best results came at degree=4. The test accuracy for this configuration was 0.345.

Degree	Train Acc.	Dev Accuracy
1	0.602	0.493
2	0.610	0.521
3	0.610	0.504
4	0.606	0.524
5	0.611	0.508
6	0.605	0.310

Simple Neural Network Results

We got the best results with degree=1, hidden layer size=128, lr=1e-3, a tanh activation function, and a dropout rate of 0.5. We also experimented with different numbers of hidden layers and the connectivity of layers. We used a hidden dimension of 128 and a 5-layer neural network for our final configuration. The test accuracy was 0.525.

Learning Rate	Dev Accuracy
1e-1	0.699
1e-2	0.699
1e-3	0.472
1e-4	0.472

We found that learning rates $>$ 1e-3 caused the model to overfit, and predict "buy" every time (this happened even with a dropout rate of 0.9), so we decided to choose an lr = 1e-3.

Dropout Rate	Dev Accuracy
0.1	0.355
0.3	0.472
0.5	0.564

Random Forest Results

We found that degree did not significantly affect results here, so we used degree 1 for simplicity. The best results came with 10 estimators. The test accuracy for this configuration was 0.337.

Estimators	Dev Accuracy
10	0.598
20	0.548
50	0.569
100	0.588

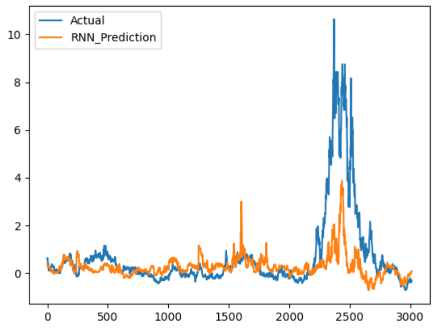
K-Nearest Neighbors Results

We limited the number of features by using degree=1. We obtained the best results by using Neighbors=500. The test accuracy for this was 0.4112.

Neighbors	Dev Accuracy
10	0.2875
20	0.2937
50	0.4826
100	0.6218
200	0.6705
300	0.6878
500	0.6974

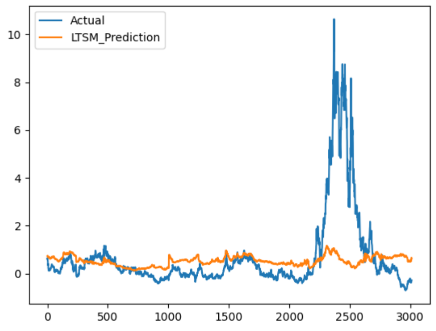
Simple Recurrent Neural Network Results

We used a Simple RNN that had 3 hidden layers with 32 hidden units each layer and tanh activation function. that minimizes the mean squared error. Using the Dev dataset, we found the optimal hyperparameters of 100 epochs and a batch size of 150. We normalized data using a Min-Max scaler. Below is our prediction of the future 1-year return on the Test dataset compared to the actual return value. The RMSE for the simple RNN model is 1.45.



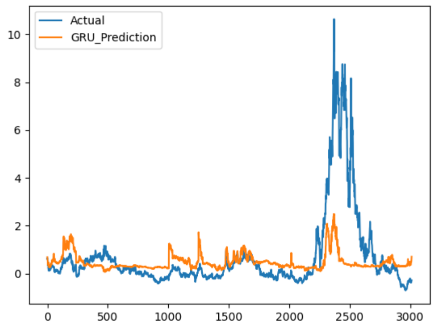
LSTM Results

We used an LSTM that had three hidden layers with 50 hidden units each layer, and a tanh activation function that minimizes the mean squared error. Using the Dev dataset, we found the optimal hyperparameters of 50 epochs, a batch size of 150, a learning rate of 0.01, a decay of 1e-7, and a momentum of 0.9. We also used dropout layers with a 0.2 dropout rate to prevent overfitting. Below is our prediction of the future 1-year return compared to the actual return value. The RMSE for LSTM is 1.7.



GRU Results

We used a GRU that had three hidden layers with 50 hidden units in each layer, and a tanh activation function that minimizes the mean squared error. Using the Dev dataset, we found the optimal hyperparameters of 50 epochs, a batch size of 150, a learning rate of 0.01, a decay of 1e-7, and a momentum of 0.9. We also used dropout layers with a 0.2 dropout rate to prevent overfitting. Below is our prediction of the future 1-year return compared to the actual return value. The RMSE for GRU is 1.58.



Best Model

The best result for the classification of the five ratings was achieved by the Simple Neural Network with an accuracy of 0.525. The Simple RNN achieved the best result for direct return predictions with a test RMSE of 1.45 and a graph more closely matching the actual return values, which was lower than the linear regression RMSE.

6. Supervised Methods Discussion

Despite having a large dataset, it was difficult to implement a model that performed well. We implemented Softmax and Linear regression as our baseline approaches. Linear Regression and the related work we looked into showed that it was extremely difficult to predict the future returns of stocks with desirable accuracy, so we focused the majority of our supervised approaches on predicting a buy rating.

In general, our supervised dataset drawn from 10 years of a bull cycle for each stock, meaning most of the data was classified as ”buy” (positive 1 year forward return) of which many were labeled ”strong buy” (10+% 1 year forward return). This means that a model could just predict ”strong buy” and achieve a decent accuracy, at least on the classification models. We faced this issue when using a high learning rate on the simple neural network, and with our KNN implementation. The KNN ended up predicting ”strong buy” for 99.8% of the test set. This likely occurred as when n-nearest neighbors was set to a high number, it just predicted the majority label of the test set, which was "strong buy."

With the classification models, we found that simpler models were ineffective. Excluding the simple neural network and KNN, the other models failed to do better than our baseline softmax regression. KNN only outperformed softmax when it converged to predict "strong buy" every time. This issue likely stems from the distribution of data in our dataset. To alleviate this we would need to include older stock data that doesn't just consist of this past 10-year bull run. Another option would be to pick stocks that gave us our desired distribution. But every industry is extremely different, and it's hard to decide how to group stocks together. Designing such a dataset is another research problem altogether.

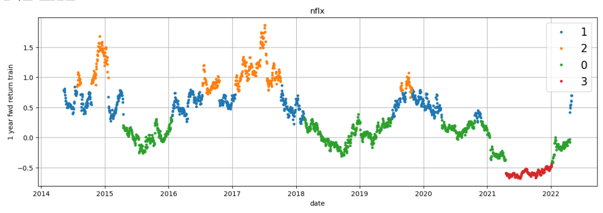
For directly predicting the 1-year future return, we found that RNN performed the best, since stock return data is sequential in nature with serial connectivity. When training the RNN models, we also realized that it was necessary to normalize the dataset to achieve better accuracy, thus we used the MinMax scaler. However, testing the different degrees of data proved to be infeasible as it would require exponentially more computing power, which our computing resources could not achieve. Thus, we used the original dataset with a degree of 1. We were surprised to find that the simple RNN model performed better than the LSTM and GRU models, with an RMSE of 1.45, which is also much better than Linear Regression which had an RMSE of 4.54. We struggled with finding an explanation of why LSTM and GRU performed worse, as we had limited knowledge of the two models. The models in general also did a much job of predicting the return of the stock in the sideways market compared to abnormal market conditions, as we expected. For example, the models did a much worse job when predicting the stock returns during COVID, when the market experienced abnormal overall returns.

Despite the drawbacks of our dataset, the simple neural network performed substantially better than our softmax regression, and the simple RNN performed better than linear regression. This is promising and leads us to believe that there is some sort of underlying pattern in past returns and the 1-year future return.

7. Unsupervised Methods Experiments

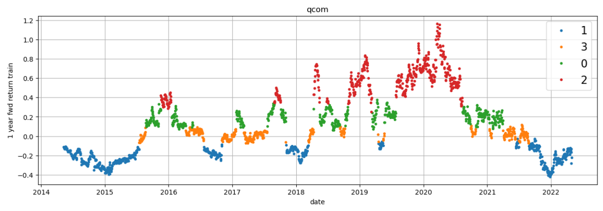
As mentioned earlier, we only used one supervised method — Gaussian HMMs — given its relevance to time-series data and due to one of its learned parameters — emission probabilities — being very close to our initial question regarding finding the probability distributions of future returns for each stock. The hyperparameter we varied was the number of states as we kept the number of iterations the same at 50 since that was sufficient enough to lead to convergence. For each stock, we treated its 1 year forward returns as the observations of the HMM. We tried to fit 5 states to the data, one for each of the 5 ratings "sell", "underweight", "neutral", "overweight", and "buy". However, this provided to be overfitting, as 2 of the states ended up having nearly identical means and covariances, which meant certain examples were essentially randomly being assigned to one of those 2 states. We reduced the number of states to fit the data to 4 and the results were much more reasonable as it segmented each example's 1-year forward return into 4 clearly different states. The best 3 stocks that fit the criteria mentioned for the HMM in **Dataset and Evaluation** are NFLX, META, and QCOM. Below are their learned means, covariances, and charts of each 1 year forward return plotted against its date and an assignment of each learned state to its date.

NFLX



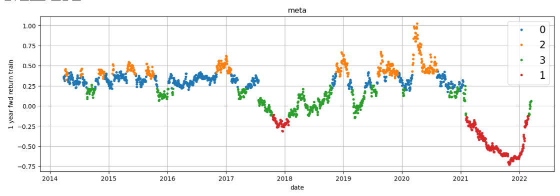
State	Mean	Covariance
0	0.01561443	0.02955345
1	0.545472	0.01699349
2	1.07640732	0.05996554
3	-0.5866326	0.00255361

QCOM



State	Mean	Covariance
0	0.19178012	0.00436335
1	-0.21206793	0.00651778
2	0.59063778	0.02886395
3	0.01418251	0.0017031

META



State	Mean	Covariance
0	0.31146542	0.00276778
1	-0.41539433	0.03290102
2	0.4919898	0.013592
3	0.06768417	0.01061731

Given that we only tried this one unsupervised learning method, we don't have other unsupervised learning methods to compare it to, however the next section discusses the results in regards to our more customized evaluation criteria.

8. Unsupervised Methods Discussion

We found these 3 stocks to have the most interesting results because it shows that the HMM, assuming Gaussian emission distributions, was able to learn a state with the mean close to 0%, or neutral without that mean being the lowest mean across all the states. In the 3 stocks above, NFLX's state 0 and QCOM's state 3 means very close to 0 while META's state 3 mean was just 1.7% greater than our chosen neutral state upper bound of +5%. State 1 for NFLX, state 2 for QCOM, and state 2 for META each also successfully were able to identify "hype", i.e. extremely large values, during the time of COVID. While this is promising, the parameters are highly dependent on the 10 year data we collected for each stock.

We chose 4 states based on their use for the investment industry. We imagine that it would be useful information to know historically how a specific stock's 1 year future returns have looked based on what state it's in. For instance, if we look at META's stock above, we see that towards the end, it entered the green "neutral" state (3) from the red "sell" state (1). An investor or trader could say that in a year, the stock is likely to fall between the mean for the green state and some range based on that states standard deviation. This is assuming the future returns for the stock follow a Gaussian.

In summary, the unsupervised nature of HMMs makes it hard to test the accuracy of the states. However, using common knowledge and some contextual financial knowledge, we can see that for some of the stocks, it can consistently assign states to values that could make sense, which could help investors have an idea of the potential returns a stock could get if they believe it's in a certain HMM state based on a 10 year window of historical data.

We had expected it to model some stocks well and some not so well. It only really found a state that was truly neutral for only 3 stocks assuming a Gaussian distribution, so out of the 10 stocks we tested, we could say it was 30% accurate in identifying relevant states to our goal and to a financial context.

Another limitation of this method is naturally that it assumes Gaussian emissions. We only tried this method since we learned GMMs in class and felt like we understood that probability distribution the best, and in 3 of the stocks, it seems to do a decent job in finding certain states. It is quite possible that returns do not follow a Gaussian distribution.

9. Conclusion

In this project, we tried to answer complex questions about financial data. We wanted to find out if there was a way to model the probability distribution of a stock's future returns simply based off the past returns for the stocks, but we were not able to successfully do so. We ended up learning how the machine learning methods could help us answer different but related questions about our problem. We learned that in order to achieve better results, we'd likely have to refine our data and include more features than returns like revenues or trading volumes or news headlines. All that would have been quite a complex and noisy set of data to find and refine and draw conclusions from, as the entire financial industry exists to do this and still is not able to do it with consistency. Not to mention, the dataset we used was from the last 10 years, which have been, on the whole, quite good for tech stocks, which means that there is an inherent trend within our data that could have biased our results. We also had not tried more unsupervised methods since we thought HMMs were the best for time-series data, and we also did not try reinforcement learning because we were taught that much later on. We did find that some of our methods were able to find some interesting details about individual stocks, which would prompt further investigation. That being said, our best classification model ended up being **a simple neural network**, with an accuracy **52.5%**, which is better than expected for the classification of stocks. We also found more success with the simple RNN for predicting the exact target return compared to the Midterm's linear regression.

Additionally, in real-world quantitative trading, people only strive to be correct a little bit over 50 percent of the time with their models, which can already lead to significant gains. In summary, we learned that this task is quite difficult but in trying to solve it, we became more familiar with various machine learning methods and learned how to experiment with them.

10. Codebase

Here's the github link to our codebase:

https://github.com/rithikp06/csci467-project

Instructions to run our models are in the README, and the stock data we used can be found in the data folder

References

Chatterjee, A., Bhowmick, H., and Sen, J. *Stock Price Prediction Using Time Series, Econometric, Machine Learning, and Deep Learning Models*. PhD thesis, Department of Data Science Praxis Business School, 2021.

Sen, J. *Stock Price Prediction Using Machine Learning and Deep Learning Frameworks*. PhD thesis, Department of Analytics and Information Technology Praxis Business School, 2021.

Xiao, D. and Su, J. *Research on Stock Price Time Series Prediction Based on Deep Learning and Autoregressive Integrated Moving Average*. PhD thesis, School of Finance, Central University of Finance and Economics, Beijing, China, 2022.