Deliverable 3: Stock Market Predictor

Deepak Singh

1. Final Training Results

After the second deliverable, I was dedicated in finding a model with a testing accuracy of ~60-70%. My previous (deliverable 2) model's 50% testing accuracy basically shows randomness and no concrete predictions. The stock market is highly volatile and at times unpredictable due to insider trading, random noise or even crashes. Additionally, stocks encompass a multitude of different macroeconomic and behavioural factors such as GDP, disposable income, PMI index, etc. Using news headlines only encompass some of these factors which is why it is expected to have low testing accuracies.

After changing my news headline dataset and implementing the S&P500 as the predictor variable, my testing accuracies saw no improvements from my previous attempt. This could be explained by the insignificance of many of these news headlines. The dataset encompasses all types of news (financial and non-financial). However, the S&P500 is mainly affected by financial and macroeconomic factors which explains why my classification attempts (LSTM, deep learning, naïve Bayes, logistics regression and random forest) failed to accurately predict the index.

Thus, I decided to focus on company stocks such as Apple and Goldman Sachs while gathering financial news headlines concerning these companies. As per my model, I had to exclude deep learning because of my significantly smaller sample size of ~1,000 news headlines per company. Additionally, I was not able to gather more data due to the potential high costs and copyright issues. With this new data, I fitted a Naïve Bayes, Logistics Regression, Random Forest and Support Vector Classification model to find which model yields the highest testing accuracy. These are my results for Apple and Goldman Sachs:

Figure 1: Apple Results

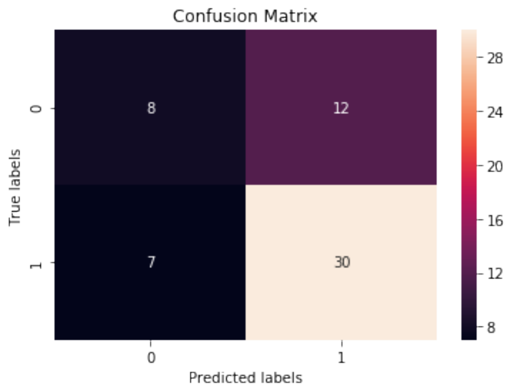
	Naïve Bayes	Logistics	Random Forest	SVC
Training Acc	0.906276870	0.937231298	0.977644024	0.95872742
Testing Acc	0.666666667	0.649122807	0.596491228	0.66666667

Figure 2: Goldman Sachs Results

	Naïve Bayes	Logistics	Random Forest	SVC
Training Acc	0.914728682	0.954457364	0.986434108	0.97868217
Testing Acc	0.527777777	0.444444444	0.555555555	0.58333333

Previously (deliverable 2), I chose Naïve Bayes due to its high testing accuracy of ~52% compared to other models but did not consider trying Support Vector Classification. With this new dataset and project approach, I decided to additionally implement SVC to see if it yields significant results. The results in Figure 1 and 2 show SVC having the highest training and testing accuracies for Apple and Goldman Sachs. While overfitting can still be seen in all four models due to their high training accuracies, the extent of overfitting significantly decreased compared to the results from deliverable 2 which had ~100% training accuracies and ~50% testing accuracies. Thus, with these results, SVC is the most promising model for this project.

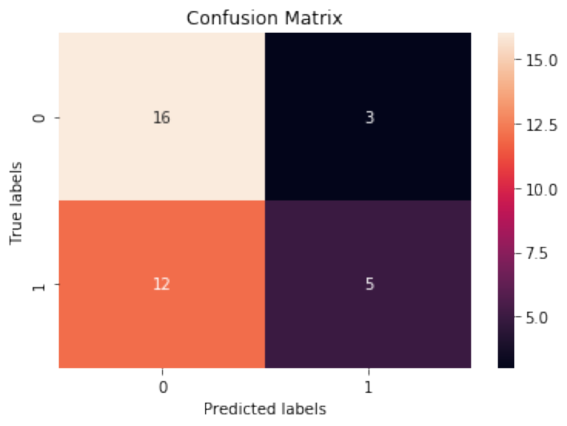
Figure 3: Apple Confusion Matrix



The chosen model predicts Apple stock price upswings at an accuracy of 81% while predicting the stock price decreases at an accuracy of 40% (recall). Out of all upswing predictions, the model is 71.4% precise and 53.3% precise for decrease predictions (Figure 3). For a predictive model, we would most likely **LONG** the stock when news seems favourable and hold no position when headlines indicate a price decrease due to the high recall and precision for price increases.

We can also see similar results with Goldman Sachs:

Figure 4: Goldman Sachs Confusion Matrix



In the case of Goldman Sachs, our model predicts its stock's downfall with an accuracy of 84%. For price increases, we see an accuracy of 29% (recall accuracy). Out of all upswing predictions, the model is 62.5% precise and 57% precise for downfall predictions (precision) (Figure 4). Due to the high relative count of false negatives and false positives, for a predictive model, we would go **SHORT** during price decreases and go **LONG** during price increases to compensate for this imbalance.

One downside of this project is its sample size. My dataset consists of 100,000 financial news headlines but only ~1000 concern Apple and Goldman Sachs (each) which is why our training and testing sizes are low. I excluded other companies like Google and Facebook because there was not sufficient data. Furthermore, one could question the reliability of these results due to the small sample size. For a future iteration of this project, gathering more data in the forms of web scraping would be preferable, however, as mentioned, this causes copyright issues. Additionally, I assume these news articles are published before being incorporated within stock prices because of the difficulty of gathering stock data in a per-minute, second or even millisecond basis. In reality, stock prices may fluctuate before news publishing because of insider trading.

Final Demonstration Proposal

For this project, I decided creating a web application showing users their profit opportunities and return rates depending on two investment strategies: Passive and Active. The passive strategy will consist of holding a position (LONG or SHORT) depending on past news article headlines. The active strategy will consist of continuously changing positions according to news article headlines in a per-day basis. This will last for the number of days in the testing set (~1.5 months). The web application will allow users to visualize their potential return depending on each strategy. I will also implement a section where users can add current Apple or Goldman Sachs news article headlines from the past few days to visualize their potential return depending on the strategy.

In terms of technologies, I will use Flask for the backend due to the small scale of this project. Additionally, since the project was coded in Python, it makes more sense to use a Python backend library for a better flow of data. For the front-end, I will use React (Javascript) because of my experience with the framework and because of its abundance of open-source libraries. I won't implement databases for this project but in fact keep all of my data and results in local storage (csv format) due to the small size of my data.