1. Project Title

Stock market analysis

2. Project Topic

Stock price movement prediction using an ensemble of different machine learning techniques and sentiment analysis

3. Problem Statement

Stock price prediction is a heated topic in prediction study of the financial area. The stock market is essentially a noisy, nonlinear, nonparametric system that is extremely hard to model with any reasonable accuracy. Investors have been trying to find a way to predict stock prices and to find the right stocks and right timing to buy or sell. Most of the techniques used in technical analysis are highly subjective in nature and have been shown not to be statistically valid.

4. Motivation

Predicting stock exchange data is an important financial subject that many financial analysts delve into. While numerous scientific attempts have been made, no method has been discovered to accurately predict stock price movement. According to a study made by International Research Journal of Engineering and Technology (2018)[1], putting up a system that can accurately predict the stock price movement may lead to attractive profits by making proper decisions.

5. Literature review

5.1. Incorporating sentiment analysis into stock price movement prediction

This report[2] gave us a preview into the incorporation of sentiment analysis in stock price prediction. To perform a sentiment analysis, there was a great deal of data preprocessing done. It is important to note that the data wouldn't be as simple as just getting daily stock prices. It has to contain a composite of multiple natures of data such as technical data macro data of the economy (GDP, CPI), daily stock prices(open price, closing price), and qualitative consisting of daily news content scraped from various sources. These features extracted were then used in conjunction with technical data to predict the direction of movement of stock prices using classification models such as logistic regression, random forest, SVM, and various neural networks (fully-connected, recurrent and convolutional).

Results from the second report[3] support the stance made and proved that sentiment analysis does improve the stock price movement prediction. Ultimately, news and social media

[1] (n.d.). stock market prediction using ann - irjet. Retrieved April 8, 2020, from https://www.irjet.net/archives/V5/i3/IRJET-V5I3634.pdf
[2] (n.d.). Novel Approaches to Sentiment Analysis for Stock ... - CS229. Retrieved April 8, 2020, from http://cs229.stanford.edu/proj2018/report/72.pdf
[3] (2019, August 10). (PDF) Stock Market Prediction Analysis by Incorporating .... Retrieved April 8, 2020, from https://www.researchgate.net/publication/331037730_Stock_Market_Prediction_Analysis_by_Incorporating_Social_and_News_Opinion_and_Sentiment

sentiments of the stocks are important features in analyzing the stocks and should be incorporated into stock price movement prediction.

5.2. Problem of machine learning approaches for our sentiment analysis

There are two approaches to sentiment analysis namely the lexicon-based approach and machine learning approach. This paper[4] discussed the limitations of the machine learning approach for sentiment analysis and suggested a lexicon-based approach as the best approach for sentiment analysis. The problem of the machine learning approach is that the sentiment classifier is trained on the labeled data from one domain and it often does not work with another domain. For example, if we train our classification model on technological news, the model will only be able to classify technological news into their respective sentiments but will perform badly on other domains such as financial news. The paper also discussed the challenges faced in finding a substantial amount of labeled data for training purposes. To overcome this problem, lexicon-based approaches such as Textblob and vadarSentiment are recommended.

5.3. Leveraging on learning-based algorithm for stock prediction

As machine learning algorithms have great learning capabilities, it is imperative for us to utilize them to predict stock price movement as such data is usually unstructured, noisy and non-linear. Since neural networks can have many layers (and thus parameters) with non-linearities, they are very effective at modelling highly complex non-linear relationships and are often used with enhancement methods to predict stock price movement. Additionally, this paper[5] research results demonstrated that the incorporation of sentiment analysis enhances the performance of the learning-based methods. As such, we will be researching learning-based algorithms such as LSTM and CNN to predict our stock price movement with and without sentiment analysis.

5.4. Long Short-Term Memory (LSTM)

It was noted that LSTM has been one of the most effective solutions for sequence prediction problems which include stock price prediction as they are able to maintain contextual information and temporal behaviors of events that allows them to learn from long-term dependence data. This is due to an important ability of RNN to map from the entire history of inputs to each output and retrive more contextual information than RNN. In this paper[6], they integrate data from network public opinions and actual behavior data as a high dimension input time series, and propose a new time series model based on LSTM. This new model trains the input data (stock prices and emotional data) to deduce the price of the stock. The

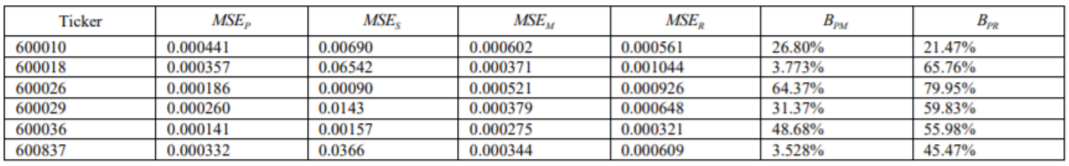
[4] (2019, June 26). (PDF) A Comprehensive Study on Lexicon Based Approaches .... Retrieved April 8, 2020, from https://www.researchgate.net/publication/333602124_A_Comprehensive_Study_on_Lexicon_Based_Approaches_for_Sentiment_Analysis

[5] (2019, August 10). (PDF) Stock Market Prediction Analysis by Incorporating .... Retrieved April 8, 2020, from https://www.researchgate.net/publication/331037730_Stock_Market_Prediction_Analysis_by_Incorporating_Social_and_News_Opinion_and_Sentiment

[6] (2017, May 24). LSTM Neural Network with Emotional Analysis for Prediction .... Retrieved April 8, 2020, from http://www.engineeringletters.com/issues_v25/issue_2/EL_25_2_09.pdf

paper had experimented with 4 models, LSTM, RNN, SVR and MLP, using the same input and used MSE as a benchmark to find the most suitable model in predicting stock prices. As shown from the figure below, MSE_p ,generated from their proposed LSTM model, has consistently shown better results as compared to other models, proving it to be the best suited for stock prediction.

The MSE of different models and B value



Ticker	MSE_P	MSE_S	MSE_M	MSE_R	B_{PM}	B_{PR}
600010	0.000441	0.00690	0.000602	0.000561	26.80%	21.47%
600018	0.000357	0.06542	0.000371	0.001044	3.773%	65.76%
600026	0.000186	0.00090	0.000521	0.000926	64.37%	79.95%
600029	0.000260	0.0143	0.000379	0.000648	31.37%	59.83%
600036	0.000141	0.00157	0.000275	0.000321	48.68%	55.98%
600837	0.000332	0.0366	0.000344	0.000609	3.528%	45.47%

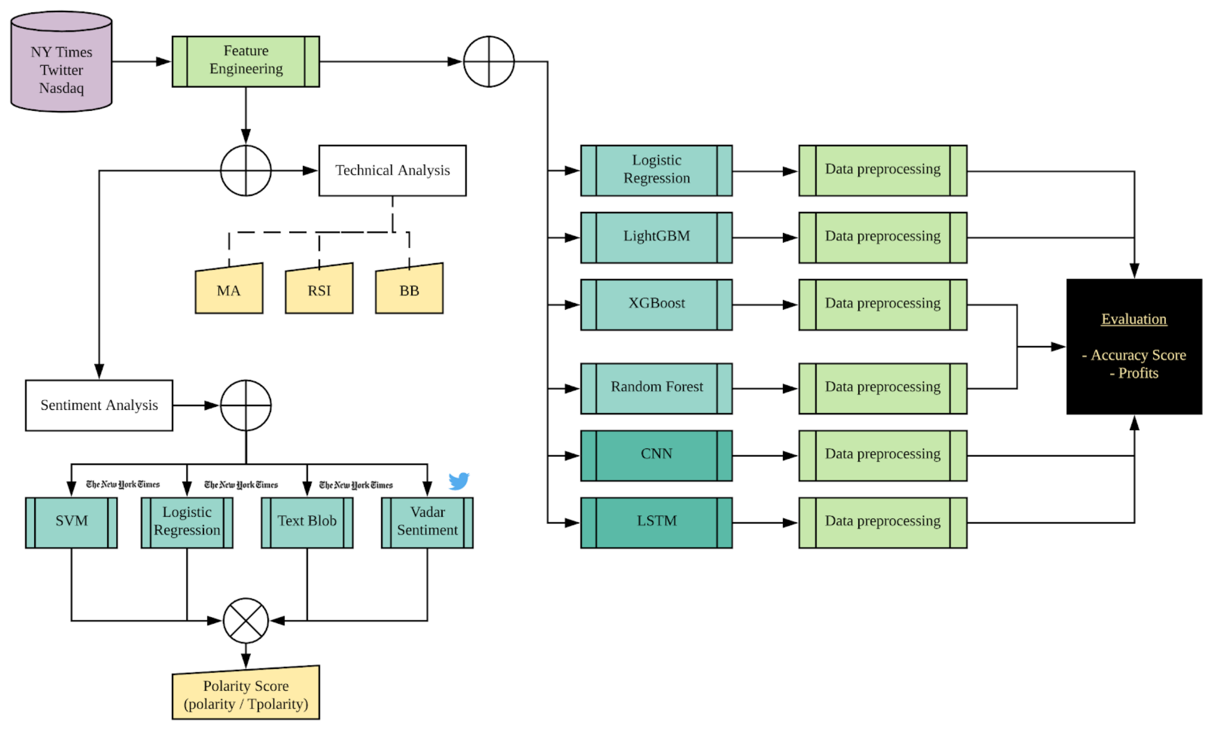
5.5. Deep Convolutional Neural Network (CNN)

CNN has been adopted a lot in recent years, primarily in the field of image recognition and that motivated many researchers to improve its implementation to minimise error rate of the model, achieving 75 - 80% success rate in terms of image recognition. This research paper[7] wanted to leverage on this improvement of CNN and use it on stocks market classification (Buy, sell, hold) through the conversion of 1 dimension time series financial data to 2 dimensional image like data. In the author's attempt, the author managed to attain a high 71.32% success rate. This is a novel idea that we would like to explore into for our project that will be beneficial for our learning.

6. Our Data Discovery Journey

After our literature review, we embarked on our data discovery journey. We started from data collection which consists of data from NY Times, Twitter and Nasdaq100. Subsequently we conducted data preprocessing and feature engineering for technical analysis and sentiment analysis. The newly engineered features together with the NASDAQ data are then fed into the various classification and learning models. Data processing model is then done on each model as the models are unique and need their own data preprocessing. The model is run twice with and without the inputs of our sentiment analysis so that we can obtain the best results. Each model is then evaluated based on their profits and their accuracy scores.

[7] (2018, April 28). (PDF) Algorithmic Financial Trading with Deep Convolutional .... Retrieved April 8, 2020, from https://www.researchgate.net/publication/324802031_Algorithmic_Financial_Trading_with_Deep_Convolutional_Neural_Networks_Time_Series_to_Image_Conversion_Approach



This flowchart shows a summary flow of what we did for our entire project

7. Tools and Resources Used

- **Techniques**: XGBoost, Random Forest, LightGBM, Logistic Regression, LSTM, CNN, SVM
- **Package Names:** Pandas, Numpy, Sklearn, Matplotlib, Seaborn, XGBoost, Selenium, Tensorflow, sklearn, TA, TextBlob, VadarSentiment, SVM
- **Programmes:** UiPath

8. Data Collection

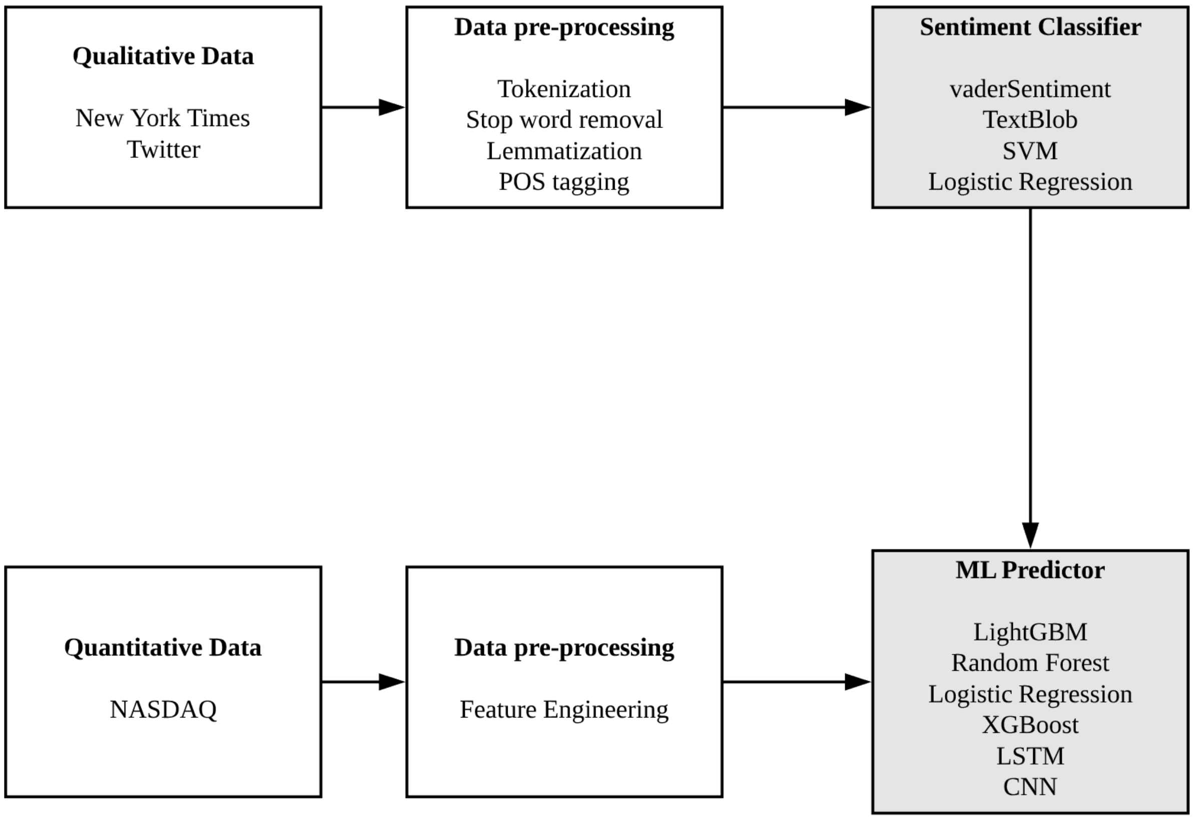
For this project, we performed stocks market prediction on NASDAQ. We obtained our qualitative dataset for sentiment analysis from Twitter and NY Times by extracting data using automated web scraping tools and by calling API endpoints of their services respectively.

For Twitter, we tried to make use of their API endpoints to extract the information we needed but due to the limit imposed by twitter, where their endpoints only return tweets from 7 days ago, we had to look for alternate ways to extract the data. Ultimately, we made use of an automation tool, UiPath, to automate our web scraping process to extract twitter data related to NASDAQ between the period of 2014 - 2020. Due to limited computational resources, it wouldn't be practical for us to return all tweets within these years because there will be millions of them. Therefore, our team did trial and error by testing the time taken for the tweets to be scraped and eventually concluded that we should scrape a random sample of 30-40 tweets in a given day between the above mentioned time frame. The scraped twitter data is stored in a csv file with two columns: (1) Date and (2) Text of the tweets.

For NY Times, as their API endpoints were free and were able to return history of the archives, we were able to get the necessary information by using NYTimes archive articles directly. Our script had to fit into the limit set (4000 requests per day and 10 requests per minute) by the NYTimes. The data extracted is stored in a csv file with four columns: (1) Published date, (2) Abstract, (3) Lead Paragraph and (4) Snippet of the article.

Next, we downloaded our quantitative data from Yahoo Finance on NASDAQ from the same time period. The downloaded file is a csv file with 7 columns: (1) Date (2) Open (3) High (4) Low (5) Close (6) Adjusted Close (7) Volume.

9. Data preprocessing



9.1. Feature Engineering - Sentiment Analysis

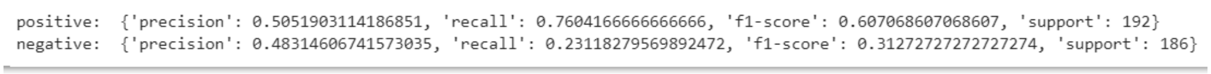
From our literature review, we note that it is imperative to clean our qualitative data before analyzing them as there could be noise in such data. Hence, before we classify our text into sentiments, we did a basic clean up of our data to ensure a more accurate sentiment classification using regular expression to create text patterns that we defined as noise such as Twitter handles and url links.

We had experimented sentiment analysis with 4 different models, Logistic regression, Support Vector Machine (SVM) and 2 unsupervised methods based on a lexicon database using TextBlob and VADER. We had settled on using Textblob and VADER for sentiment analysis for news and tweets respectively.

Textblob is an unsupervised lexicon-based sentiment analyzer for text that is built based on the NLTK and Pattern libraries. Both SVM and Logistic regression are supervised classification methods that require labeled data to train the model and the domains would be specific to the dataset we had trained.

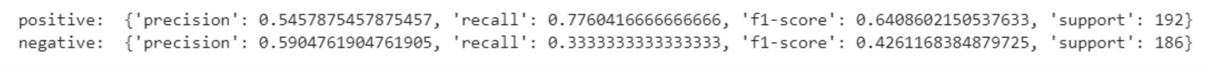
As the data we scraped from the internet do not have this feature, we had used an unrelated labeled financial news data set to train and test our model. We had done multiple testing to determine which are the best ngram parameters used for each supervised model, then implemented the best parameters for each model on the training data to find out which is the best for predicting polarity. To determine the best ngram, we experimented with different ngram permutations from(1,1) to (3,3) , a total of 12 models for both SVM and logistic regression. We then came to a conclusion that ngram(2,2) had the best results.

As shown from the figures below, Textblob is comparable to SVM and Logistic without training. In addition, if we were to use SVM and Logistic on the unlabeled news data, we are unable to guarantee this level of accuracy as the data we had scraped may not be specified towards financial news only. Furthermore, the labelled financial dataset we had used was insufficient to train our supervised models. Hence, the team has decided to utilise TextBlob for further sentiment analysis on the news.



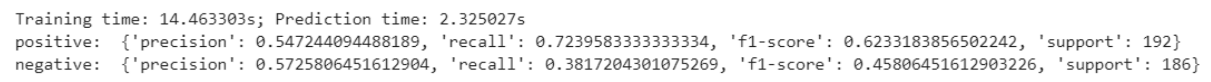
```
positive:  {'precision': 0.5051903114186851, 'recall': 0.7604166666666666, 'f1-score': 0.607068607068607, 'support': 192}
negative:  {'precision': 0.48314606741573035, 'recall': 0.23118279569892472, 'f1-score': 0.31272727272727274, 'support': 186}
```

Textblob



```
positive:  {'precision': 0.5457875457875457, 'recall': 0.7760416666666666, 'f1-score': 0.6408602150537633, 'support': 192}
negative:  {'precision': 0.5904761904761905, 'recall': 0.3333333333333333, 'f1-score': 0.4261168384879725, 'support': 186}
```

Logistic regression ngram(2, 2) for count vectorizer



```
Training time: 14.463303s; Prediction time: 2.325027s
positive:  {'precision': 0.547244094488189, 'recall': 0.7239583333333334, 'f1-score': 0.6233183856502242, 'support': 192}
negative:  {'precision': 0.5725806451612904, 'recall': 0.3817204301075269, 'f1-score': 0.45806451612903226, 'support': 186}
```

SVM ngram(2, 2) count vectorizer, kernel = "linear"

The team has deemed that VADER would be a good fit for analysis sentiments on tweets as it is a lexicon and a rule-based unsupervised sentiment analysis tool that is specifically tailored towards sentiment analysis in social media. For the tweets, we had left the text primarily untouched except for cleaning special characters that are not normally found in a sentence. This is because VADER utilises certain punctuation, capitalisation, conjunctions and degree modifiers to determine the magnitude of the polarity, and removing such context identifiers would have a significant impact on our polarity (Pandey, 2019). In addition, VADER also brings in real-word context with modern day slangs and emoticons commonly used in social media text so we did not have to take into account slangs that do not appear in a proper dictionary (e.g. laugh out loud as LOL).

After running Textblob and VaderSentiment, there were two new features : polarity and Tpolarity. polarity is the sentiment score achieved using Textblob on Newyork times dataset and Tpolarity is the sentiment score achieved using VaderSentiment on our Twitter dataset.

9.2. Feature Engineering - Technical Indicators

A lot of feature engineering will have to be done because the Open, High, Low, Adjusted Close, Volume(OHLCV) data only provides us with 2 useful features. This is because OHLC are very similar and highly correlated, so we would just take the ‘Close’ column, which is more indicative as it represents the price that the stock/index closed at. Another useful feature which we might be able to use will be the ‘Volume’ column.

Feature engineering will mostly involve usage of technical indicators such as Moving Average of 7 days, Relative Strength Index (RSI) and Bollinger Bands. We will also explore different features such as taking the past X day price, which can explain the stock movement in Y days with a high degree of certainty. More importantly, we will use the output (positive/negative news) of the sentiment analysis (polarity + Tpolarity), which is very useful for prediction of the prices.

Apart from that, we will also engineer other features such as the number of days the price has been increasing/decreasing. Also, no scaling and normalising has to be done as all features fed into the classification models are categorical.

10. Method

10.1. Classification Models

For all classification models, the features will be RSI, yesterday’s price, bbhi (Bollinger Bands High Indicator) and bbli (Bollinger Bands Low Indicator). Output will be either 1 (price goes up) or 0 (price goes down). Accuracy and profits are important, but we prioritise profit over accuracy in this case, as profit is eventually what we want to achieve. Therefore, profit will be used to compare the model’s performance. Note that we are still looking at accuracy even though profit is the main metric, because a model with higher accuracy tends to fare better, but the model that generates the best profit need not be the one with the highest accuracy.

We discovered that classification models tend to predict ‘Buy’ in most cases, and some even predict ‘Buy’ in all cases. This is due to the fact that NASDAQ has been on a bull market during this period, most of the data points would be positive (roughly ⅔ of the data). In view of that, we would want a model to predict negatives sometimes (to do swing trades), as predicting positives all the time is equivalent to holding the stock. As a result, we are only going to be looking at short profits for classification models, to see how good the model is recognising the swings in NASDAQ.

10.1.1. Light Gradient Boosting Method (LGBM)

Light GBM is a tree based learning technique that is applied onto a gradient boosting framework[8]. It is a relatively new technique. Unlike other tree based learning

[8] (2017, August 17). What is LightGBM, How to implement it? How to fine tune the .... Retrieved April 8, 2020, from

techniques where the trees grow vertically, LGBM grows trees horizontally (leaf wise). The reason why LGBM is growing in popularity is the ability for it to handle large sizes of data and the algorithm takes lower memory to run. Therefore, we thought it would be applicable to our project.

We achieved decent results with LGBM - it has higher than average profits compared to other models and its accuracy is high as well. The drawback to using this model is the higher variance of profits, running this model on different datasets might yield different results.

10.1.2. eXtreme Gradient Boosting (XGBoost)

Similar to LGBM, XGboost is a boosting framework. It is an ensemble technique that penalizes error by adding new models until no further improvement can be made. However, there are some structural differences in LGBM and XGBoost where the former makes use of Gradient-based One-Side Sampling (GOSS) to filter data for finding split values while the latter utilizes a pre-sorted algorithm and Histogram-based algorithm to find the best fit[9]. And unlike LGBM, XGBoost doesn't handle categorical values. However, given how XGBoost has been praised by many data scientists as the model with the better performance, we have decided to apply it in our dataset.

We achieved the highest accuracy with this model, but it is predicting positives almost 100% of the time. We inferred that this model prioritises accuracy a lot, which might not be ideal in this situation. As a result, it will not generate any short profits at all, and is generally less effective compared to other models.

10.1.3. Logistic Regression

Logistic regression is a regression analysis used to predict categorical variables[10]. To be specific, this technique predicts a probability value between [0,1] for a bivariate classification and it is done using a sigmoid function. Also, this technique uses a different loss function compared to regression. It uses a log loss function to calculate the error.

Logistic Regression gave us our best profits and it is the best model to use in our scenario. It predicts negatives very seldomly but it is right most of the time when doing so, resulting in higher than average profits compared to other models. The cons to this model is that the variance of the profits is the highest.

https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

[9] (n.d.). CatBoost vs. Light GBM vs. XGBoost - KDnuggets. Retrieved April 8, 2020, from https://www.kdnuggets.com/2018/03/catboost-vs-light-gbm-vs-xgboost.html

[10] (2018, May 28). Introduction to Machine Learning Algorithms: Logistic .... Retrieved April 8, 2020, from https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36

10.1.4. Random Forest

Random forest is built upon a decision tree model. A decision tree is a tree-like graph of decisions and their possible consequences. It is a flowchart-like structure whereby each node represents a feature, each link represents a decision and each leaf represents an outcome (classification/continuous). The nodes are split till a predetermined stopping condition is met or till no more nodes are remaining to be split. Random forest is a model that combines multiple different individual decision trees so that the predictions are more accurate[11]. It is random in nature because each decision tree in the whole “forest” considers a random subset of features when making predictions. With more experimentations and randomness, the overall predictions become more accurate.

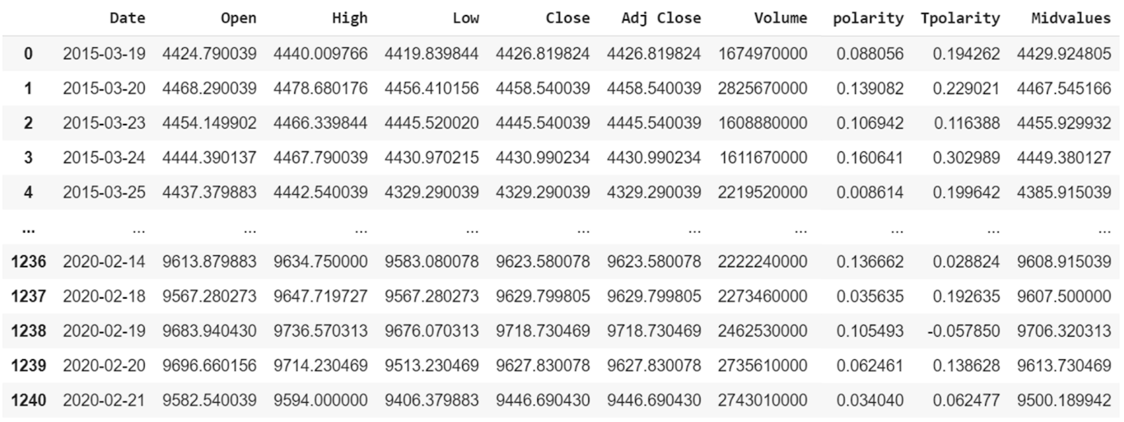
However, the performance of Random Forest in our case has been dismal with very low accuracy and profits, the lowest of all the models. By looking at the predictions, the model tries to predict more negative test cases, but by doing so has led to more inaccurate predictions. From this model, we have learnt that while we want our models to predict negatives, too much of negative predictions might actually affect profits negatively.

10.2. Long Short Term Memory (LSTM)

The specifics of LSTM has been explained in our literature review. For our LSTM model, we will be running with and without sentiment analysis in order to illustrate the significance of sentiment analysis on our stock price prediction. We will be using RMSE to compare the two model’s performance.

10.2.1. Feature Engineering

We create a list of technical indicators from open, high, low, close, adj close and volume data that we retrieved. On top of the technical indicators, we have included our sentiment analysis from NY Times and Twitter as shown as “polarity” and “Tpolarity” respectively in the table below.



	Date	Open	High	Low	Close	Adj Close	Volume	polarity	Tpolarity	Midvalues
0	2015-03-19	4424.790039	4440.009766	4419.839844	4426.819824	4426.819824	1674970000	0.088056	0.194262	4429.924805
1	2015-03-20	4468.290039	4478.680176	4456.410156	4458.540039	4458.540039	2825670000	0.139082	0.229021	4467.545166
2	2015-03-23	4454.149902	4466.339844	4445.520020	4445.540039	4445.540039	1608880000	0.106942	0.116388	4455.929932
3	2015-03-24	4444.390137	4467.790039	4430.970215	4430.990234	4430.990234	1611670000	0.160641	0.302989	4449.380127
4	2015-03-25	4437.379883	4442.540039	4329.290039	4329.290039	4329.290039	2219520000	0.008614	0.199642	4385.915039
...	...	...	...	...	...	...	...	...	...	...
1236	2020-02-14	9613.879883	9634.750000	9583.080078	9623.580078	9623.580078	2222240000	0.136662	0.028824	9608.915039
1237	2020-02-18	9567.280273	9647.719727	9567.280273	9629.799805	9629.799805	2273460000	0.035635	0.192635	9607.500000
1238	2020-02-19	9683.940430	9736.570313	9676.070313	9718.730469	9718.730469	2462530000	0.105493	-0.057850	9706.320313
1239	2020-02-20	9696.660156	9714.230469	9513.230469	9627.830078	9627.830078	2735610000	0.062461	0.138628	9613.730469
1240	2020-02-21	9582.540039	9594.000000	9406.379883	9446.690430	9446.690430	2743010000	0.034040	0.062477	9500.189942

[11] (2017, December 27). Random Forest Simple Explanation - Will Koehrsen - Medium. Retrieved April 8, 2020, from https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

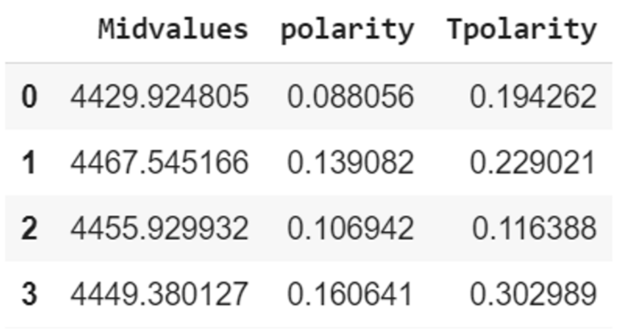
10.2.2. Feature Selection

When researching which features to use for our LSTM model, we discovered that most LSTM models only included one feature for sequence prediction which is the mid value (average price of high and low) as adding additional features such as open, close and volume will result in lower performance of the model. To validate this, we trained our model by adding various features such as Moving Average, RSI and BB to the model and have noted that the impact on the performance was insignificant.

Hence, we will be selecting only three features for the LSTM model with sentiment analysis which are polarity, tpolarity and midvalues. Whereas, we will only include one feature which is midvalues for the LSTM model without sentiment analysis.

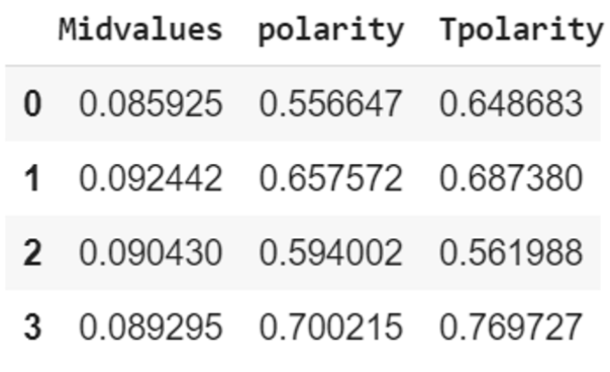
10.2.3. Normalising Data

Subsequently, we used MinMaxScaler from sklearn to normalise the data to a range of 0 to 1 since the scales of the data between midvalues and the other two features are very different.



	Midvalues	polarity	Tpolarity
0	4429.924805	0.088056	0.194262
1	4467.545166	0.139082	0.229021
2	4455.929932	0.106942	0.116388
3	4449.380127	0.160641	0.302989

Before Normalization



	Midvalues	polarity	Tpolarity
0	0.085925	0.556647	0.648683
1	0.092442	0.657572	0.687380
2	0.090430	0.594002	0.561988
3	0.089295	0.700215	0.769727

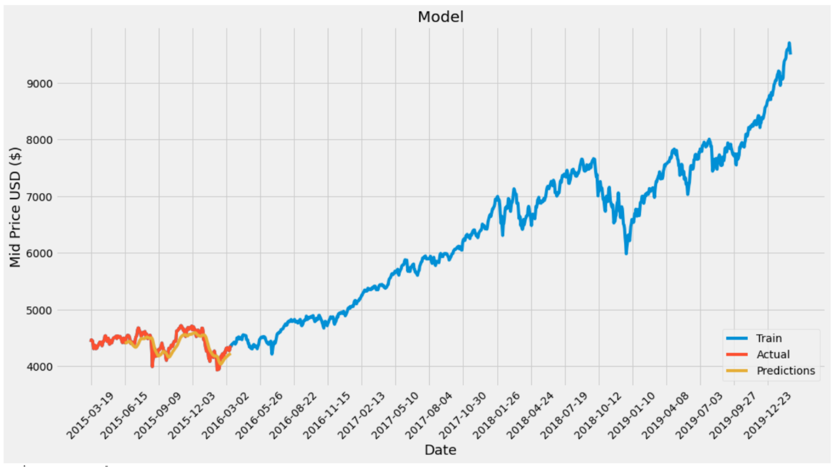
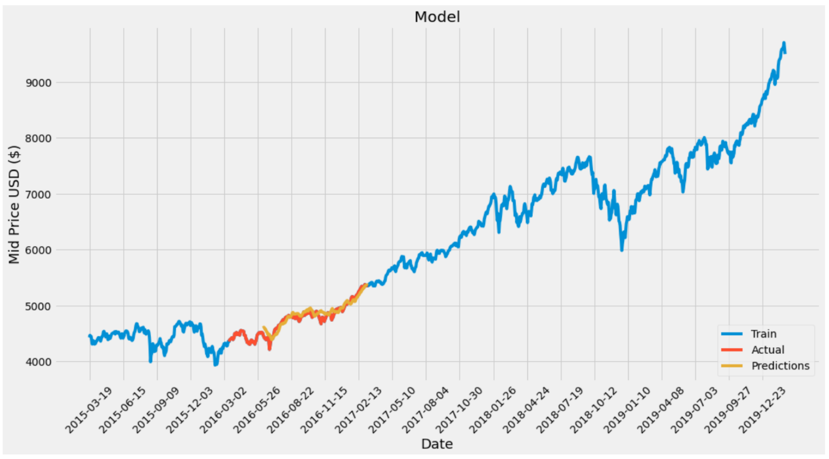
After Normalization

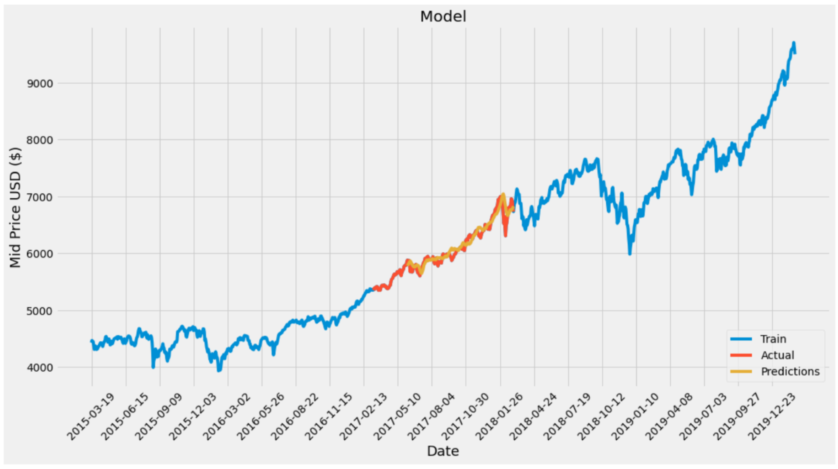
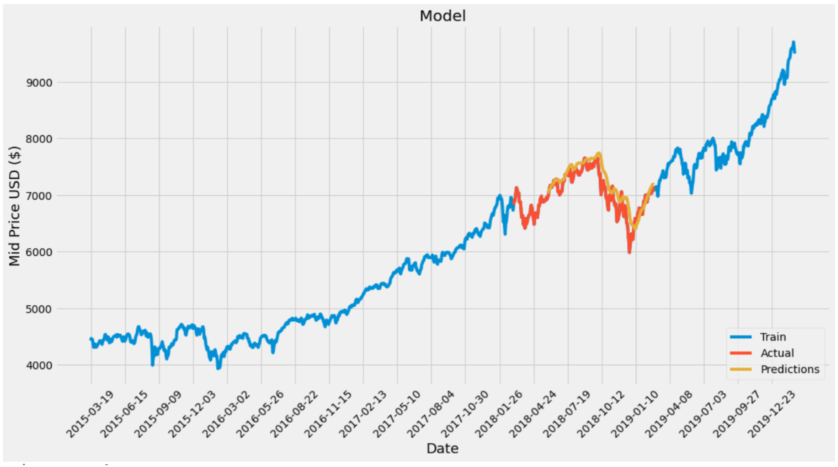
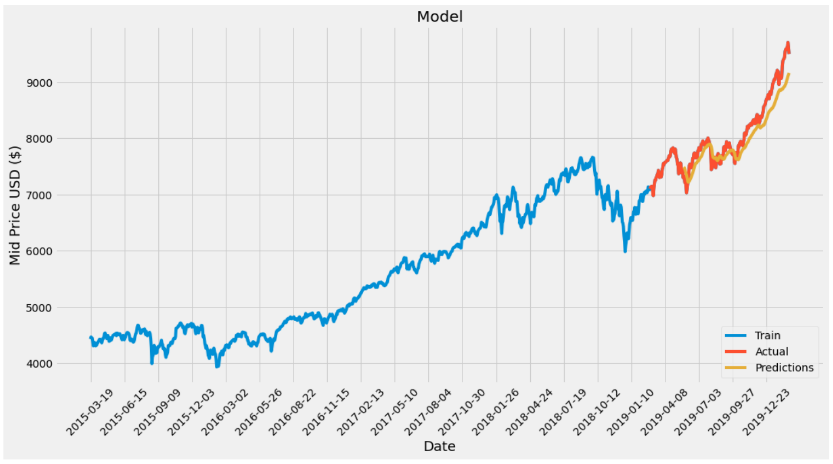
10.2.4. Splitting Data into Train and Test Data

After we have normalized the data, we have splitted the data into training data and test data by using kfold cross validation, setting the number of folds to 5. This means for every iteration, 80% of the data would be randomly chosen from the actual dataset for training and the remaining 20% for validation. As we are training the LSTM model to look back 60 days to predict the 61st day and so on, the first 60 days in the training data would always be excluded from the target data (y_train). Hence, the training data used for prediction will only contain "number of observations - 60" rows of data.

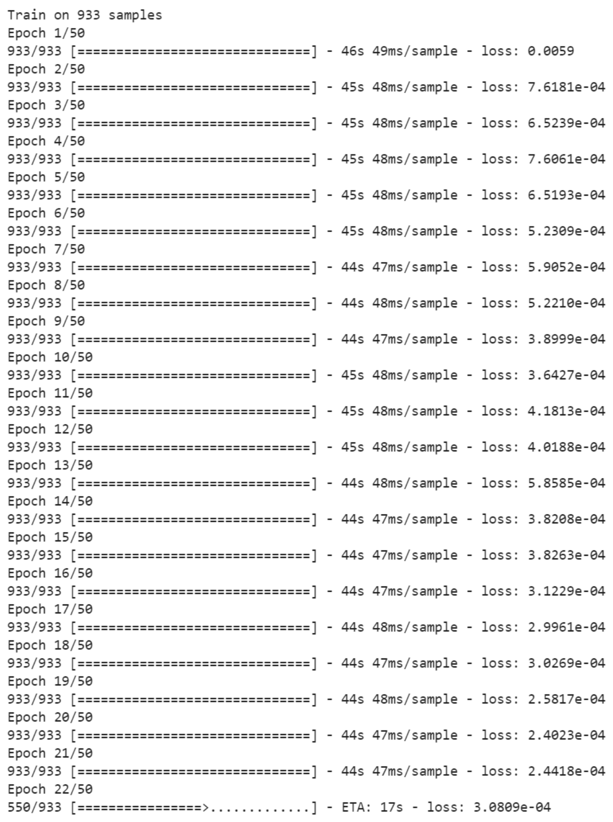
10.2.5. Training the Model

As mentioned, we have used kfold cross validation to run our LSTM models (shown in diagram below) as it ensures that every observation from the original dataset has the chance of appearing in training and test sets which could help to prevent the model from overfitting. We reshape our training and test data so that it could fit our LSTM model.

Iteration	Results (LSTM model with sentiment analysis)
1	
2	

3	
4	
5	

Next, we initialize our sequential model and add a LSTM layer with 200 memory cells and an input shape that takes in the number of observations in the training data as well as the number of features we are feeding the model. The return sequence for this layer will be set to True as we will want to return the full sequence in the output sequence. We then add another LSTM layer with 200 memory cells, setting the return sequence to false this time as we will only want the last output in the output sequence. We then add two Dense layers which are layers of neurons and finally, compile the model by using adam optimizer as our optimizer and MSE for our loss function. We train the model by setting batch size to 10 and epochs to 50. From the screenshot below, the MSE can be seen decreasing at every epoch. The loss did not change much after 50 epochs which is why we set our model to run only for 50 epoch.



```
Train on 933 samples
Epoch 1/50
933/933 [==============================] - 46s 49ms/sample - loss: 0.0059
Epoch 2/50
933/933 [==============================] - 45s 48ms/sample - loss: 7.6181e-04
Epoch 3/50
933/933 [==============================] - 45s 48ms/sample - loss: 6.5239e-04
Epoch 4/50
933/933 [==============================] - 45s 48ms/sample - loss: 7.6061e-04
Epoch 5/50
933/933 [==============================] - 45s 48ms/sample - loss: 6.5193e-04
Epoch 6/50
933/933 [==============================] - 45s 48ms/sample - loss: 5.2309e-04
Epoch 7/50
933/933 [==============================] - 44s 47ms/sample - loss: 5.9052e-04
Epoch 8/50
933/933 [==============================] - 44s 48ms/sample - loss: 5.2210e-04
Epoch 9/50
933/933 [==============================] - 44s 47ms/sample - loss: 3.8999e-04
Epoch 10/50
933/933 [==============================] - 45s 48ms/sample - loss: 3.6427e-04
Epoch 11/50
933/933 [==============================] - 45s 48ms/sample - loss: 4.1813e-04
Epoch 12/50
933/933 [==============================] - 45s 48ms/sample - loss: 4.0188e-04
Epoch 13/50
933/933 [==============================] - 44s 48ms/sample - loss: 5.8585e-04
Epoch 14/50
933/933 [==============================] - 44s 47ms/sample - loss: 3.8208e-04
Epoch 15/50
933/933 [==============================] - 44s 47ms/sample - loss: 3.8263e-04
Epoch 16/50
933/933 [==============================] - 44s 47ms/sample - loss: 3.1229e-04
Epoch 17/50
933/933 [==============================] - 44s 48ms/sample - loss: 2.9961e-04
Epoch 18/50
933/933 [==============================] - 44s 47ms/sample - loss: 3.0269e-04
Epoch 19/50
933/933 [==============================] - 44s 48ms/sample - loss: 2.5817e-04
Epoch 20/50
933/933 [==============================] - 44s 47ms/sample - loss: 2.4023e-04
Epoch 21/50
933/933 [==============================] - 44s 47ms/sample - loss: 2.4418e-04
Epoch 22/50
550/933 [================>.............] - ETA: 17s - loss: 3.0809e-04
```

10.3. Convolutional Neural Network (CNN)

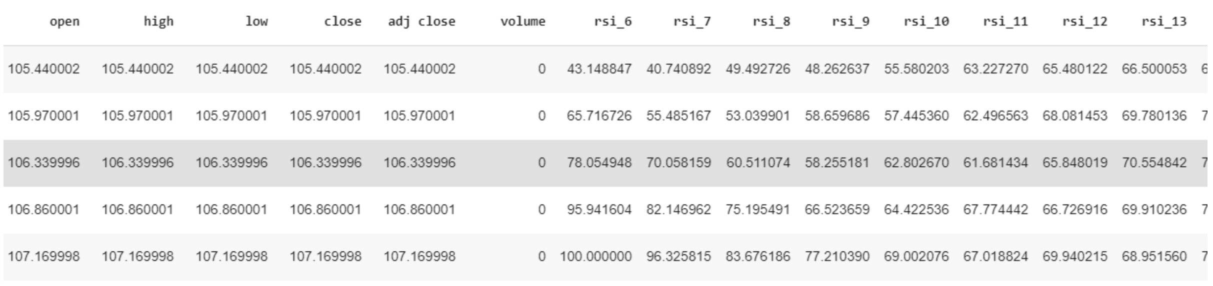
The specifics of CNN has been explained in our literature review. This is a technique that we found was very interesting upon reading the report. However, it is rather complex so we followed a tutorial[12] closely to achieve this.

Similar to LSTM, we performed CNN with and without sentiment analysis, using f1 score as the optimisation metric.

10.3.1. Feature Engineering

We create a list of technical indicators from open, high, low, close, adj close and volume data that we retrieved. The technical indicators formed is similar to those that we use in our classification models above, with Bollinger Bands, RSI etc. These technical indicators will form the list of features that we will be using to train the data.

[12] (n.d.). Stock Buy/Sell Prediction Using Convolutional Neural Network. Retrieved April 8, 2020, from https://towardsdatascience.com/stock-market-action-prediction-with-convnet-8689238feae3



open	high	low	close	adj close	volume	rsi_6	rsi_7	rsi_8	rsi_9	rsi_10	rsi_11	rsi_12	rsi_13	
105.440002	105.440002	105.440002	105.440002	105.440002	0	43.148847	40.740892	49.492726	48.262637	55.580203	63.227270	65.480122	66.500053	6
105.970001	105.970001	105.970001	105.970001	105.970001	0	65.716726	55.485167	53.039901	58.659686	57.445360	62.496563	68.081453	69.780136	7
106.339996	106.339996	106.339996	106.339996	106.339996	0	78.054948	70.058159	60.511074	58.255181	62.802670	61.681434	65.848019	70.554842	7
106.860001	106.860001	106.860001	106.860001	106.860001	0	95.941604	82.146962	75.195491	66.523659	64.422536	67.774442	66.726916	69.910236	7
107.169998	107.169998	107.169998	107.169998	107.169998	0	100.000000	96.325815	83.676186	77.210390	69.002076	67.018824	69.940215	68.951560	7

10.3.2. Normalising Data

Subsequently, we used MinMaxScaler from sklearn to normalise the data to a range of 0 to 1.

10.3.3. Feature Selection

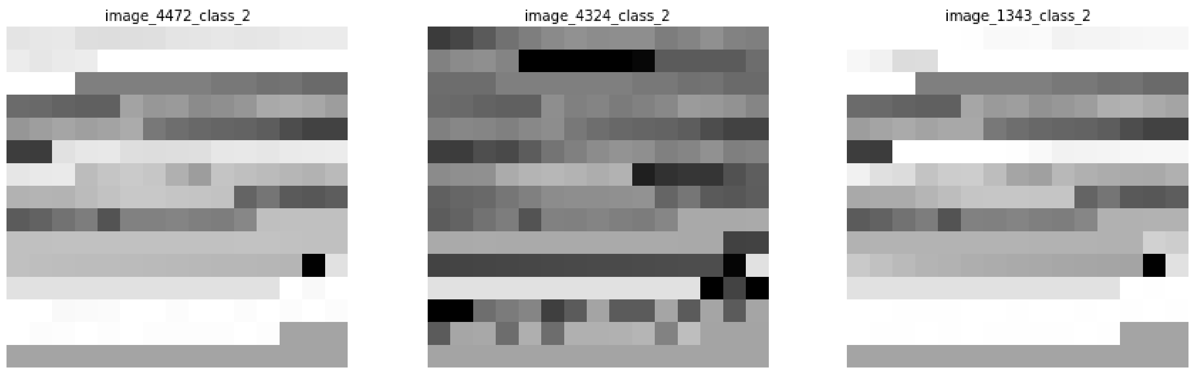
With feature selection techniques from sklearn, we selected 225 high quality features so that we can use the features to form a 15x15 image shown below. We used two feature selection techniques (1) methods f_classif and (2) mutual_info_classif and selected the common features chosen using the 2 methods.

10.3.4. Handling Imbalanced Class

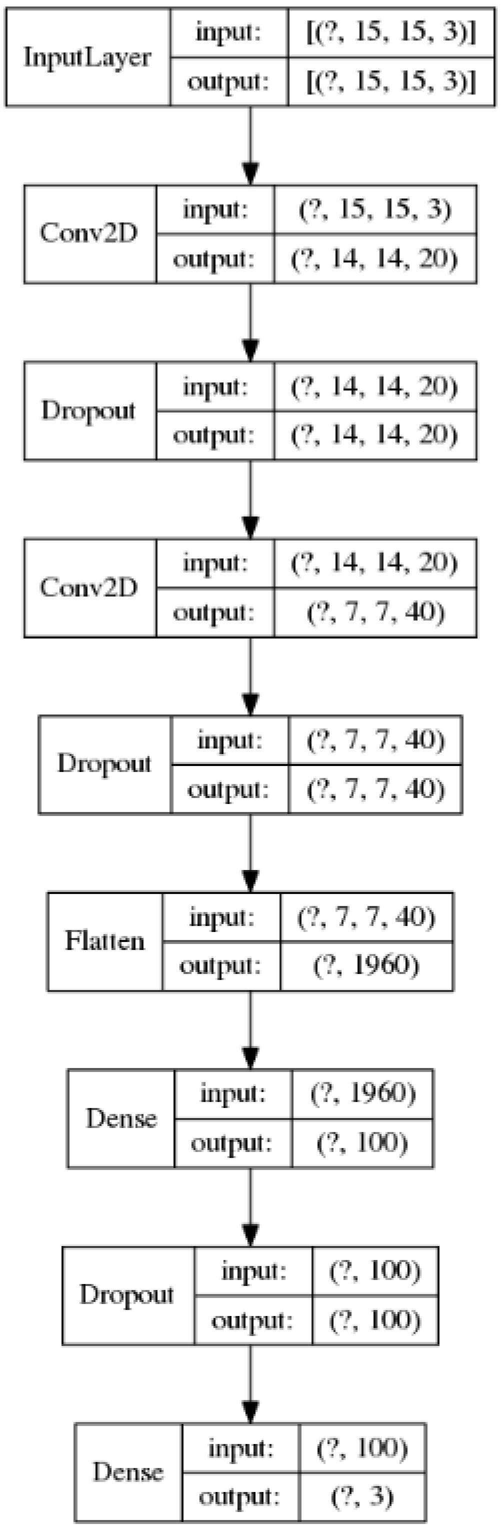
There is an imbalance in the number of instances between hold, buy and sell (3 classifications) therefore, we had to use sample weights, where the model will focus on some samples and give the sample more weight.

10.3.5. Training the Model

With the selected features, we converted the selected features into 2 dimension images split into the 3 classes.



15x15 "Image"



Structure of our model

Dropout layer - to prevent the model from overfitting
Flatten layer - Transform 2 dimensional matrix of features into vector
Dense layer - a layer of neurons
Conv2D - 2 dimensional convolution layer where the filter occurs

Relu activation function was used in our model and parameters were shown in the diagram above. EarlyStopping from tensor flow was used to prevent the overfitting of the model as well. For CNN, we did not tune the hyperparameters as we are unsure of the specifics of the convolution layer to do proper tuning and had a lack of time for the project to look further into that. Therefore, we used the hyperparameters that the author used and found that was the most optimised. This is one of the things we will explore in our future works.

11. Results and discussion

To evaluate across all our models, we use profit as the common evaluation metric. Profit is calculated using the 7 day price minus the current price for longs, and the opposite for shorts. In the case when our CNN and LSTM models predict a 'Hold', then nothing will be done. We will not use accuracy as our evaluation metric as an accurate model does not imply that it will generate high profits - XGBoost predicts all 'Buy', which is equivalent to holding the stock.

11.1. Classification Models Results

		Without sentiment analysis	**With sentiment analysis**
LGBM	Accuracy	61.466%	62.062%
	Average Profits	5.96	- 32.38
Logistic Regression	Accuracy	61.378%	63.151%
	Average Profits	40.63	- 11.77
Random Forest	Accuracy	59.706%	53.385%
	Average Profits	- 43.38	- 571.76
XGBoost	Accuracy	61.848%	59.650%
	Average Profits	0.00	- 161.36

From the above results, we can see that the classification models that ran with sentiment analysis have lower performance as compared to the classification models that ran without sentiment analysis. We concluded that one possible reason for this could be due to the market adage "buy the rumor, sell the news" where sometimes, the news have already been priced in and will not be useful in classification models in predicting the price. Another possible reason for this is that the classification models are not able to capture such complex and nonlinear relationships between the sentiment analysis and the stock price which resulted in lower performance.

From the above results, we also found that LGBM is one of our better models as it achieves our objective of high profits. It returned higher than average profits but variance is high.

LGBM	
Pros	**Cons**
Higher average profits	Generally takes longer to run
Does not predict all positives most of the time	High variance of profits
High accuracy	

Logistic Regression is good also with the average profits that it returned being the highest. However, we found LGBM being better because Logistic Regression tends to predict all positives that results in no trade being made.

Logistic Regression	
Pros	**Cons**
Able to make very high profit sometimes	Highest Variance of profits
Fast runtime	Predicts positives most of the time
	Lower accuracy

Random Forest performed badly and seldom returned a positive profit. Average profit derived from Random Forest

Random Forest	
Pros	**Cons**
Predicts the most negatives among the other models	Very low profits, negative most of the time.
	Low accuracy especially with sentiments

XGBoost performed badly as well. It predicts positive outcomes all the time, rendering the model useless as it is the same as a baseline model.

XGBoost	
Pros	**Cons**

Best accuracy	Very low profits
	Predicts positives most of the time

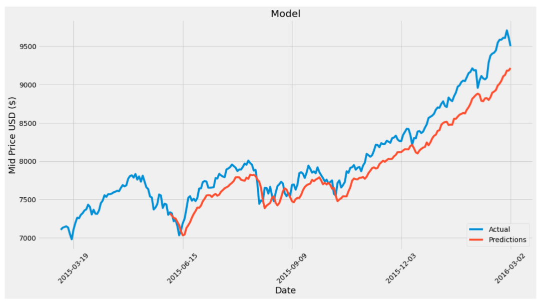
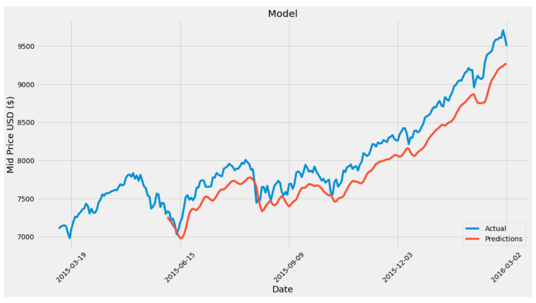
In conclusion, among the 4 classification models we tried, LGBM and Logistic Regression are the 2 models to choose from. Logistic regression generates the highest profits and also has the higher variance of profits compared to LGBM. Thus, logistic regression is recommended for traders who prefer to 'make it big' and have a high risk appetite. In contrast, LGBM though having a lower profit than logistic regression still yield high average profits. The variance of profit is not as large as mentioned, making it relatively safe. Thus, it is recommended for trades who have lower risk appetite.

11.2. Neural Network Models Results

		Without sentiment analysis	**With sentiment analysis**
LSTM	RMSE	8207.90	4750.00
	Accuracy	55.5%	57.3%
	Average Profits	12424.29	16179.18
CNN	F1 Score	83.8%	85.8%
	Accuracy	81.2%	84.7%
	Average Profits	- 2238.27	- 82.39

LSTM is the model that performed the best out of all the models that we attempted. Similar to what our research suggests, sentiment analysis did improve the performance of both neural networks. LSTM's unique architecture of memory cells played a significant role in achieving this as stock price movement prediction does require learning long term memory to make good decisions. As you may have noticed, the accuracy score is pretty low for the LSTM models. This is because when we train our LSTM model, we train our model to predict price and not movement whereas the accuracy here is to evaluate how well the model predicts movement. Hence the accuracy will be low. However, despite having low accuracy, our Profits are high as our LSTM model is able to capture the magnitude of the movement which you can see from the graph below. Also, from the visualization, we can also see that the LSTM model with sentiment analysis is able to capture the fluctuation in stock prices more accurately than the LSTM model with no sentiment analysis where the predicted curve for the LSTM model with no sentiment analysis is much smoother.

LSTM with sentiment analysis	**LSTM without sentiment analysis**

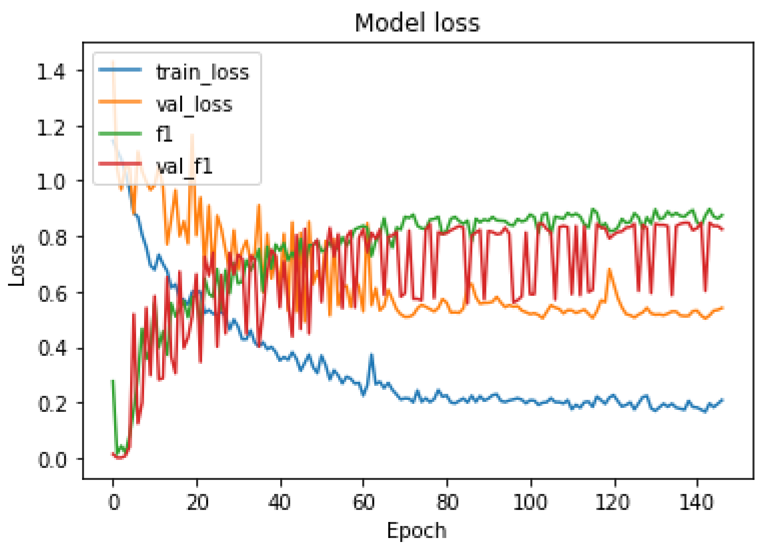
	
RMSE: 4750.00	RMSE: 8207.90

Model	**RMSE 1st iteration**	**RMSE 2nd iteration**	**RMSE 3rd iteration**	**RMSE 4th iteration**	**RMSE 5th iteration**
LSTM with sentiment analysis	***2511.45***	***2713.37***	***3555.59***	***4039.82***	***4750.00***
LSTM without sentiment analysis	***4488.33***	***4842.27***	***6193.06***	***7092.17***	***8207.90***

From the comparison above, we can see that the LSTM model with sentiment analysis has a lower root mean squared error compared to the LSTM model without sentiment analysis. We can infer that sentiments have a positive impact on stock price prediction.

LSTM	
Pros	**Cons**
Best profit result	High variance in profits
Able to learn long-term temporal dependencies	Poor accuracy results
Able to handle noise and nonlinear	Neural network tend to overfit

CNN lacklustre performance came as a surprise to us as research had shown promising results coming from CNN. Also, despite having a high accuracy score, CNN had a poor profit result. There are a few possible reasons for this poor performance. It might be due to our hyperparameters, network architecture or the imbalance data resulting in the model predicting a lot of buy and hold, forgoing sell. As mentioned, this is something that we will look into in our future works.



CNN	
Pros	**Cons**
Best accuracy	Poor profits results
	The output varies due to keras weight initialization (inconsistent)

In conclusion, among the 2 neural networks, LSTM is a better model. Despite having a higher variance of profits of the 2 models, LSTM generates significantly higher profits than CNN. The problem of neural networks having the tendency to overfit can be overcomed through various techniques like kfolds and dropout layers that we tried on LSTM and CNN respectively and with those, we were able to achieve a better result. Therefore, LSTM being an effective model for sequence prediction is a good model for stock price movement prediction.

12. Conclusion

In conclusion, we had decided to use accuracy and profits as our final metrics for evaluations of our model with and without sentiment analysis. As you can see from below, for accuracy, the majority, not all, of the models react positively to the addition of sentiment analysis. Random Forest and XGBoost reacted negatively to the addition of sentiment analysis in terms of accuracy. When looking solely at accuracy, we found that CNN was the best performer for both with and without sentiment analysis.

However, we felt that this was an unfair comparison as the models fundamentally differ from each other. Hence, we have decided to use profits as our **main** metric for evaluation as it would directly answer our objective. In terms of profits, the classification model has a negative reaction to the addition of sentiment into the model which is a different analysis from using accuracy. Both Neural Network models had a positive reaction to sentiment analysis. LSTM is the best model among all the models we attempted as it earned the highest profits.

	Classification Models				**Neural Networks**	
Models	LGBM	Random Forest	Logistic Regression	XGBoost	CNN	LSTM
Accuracy w/o Sentiment Analysis	61.466%	59.706%	61.378%	61.848%	81.2%	55.5%
Accuracy with Sentiment Analysis	62.062%	53.385%	63.151%	59.650%	84.7%	57.3%
Profit w/o Sentiment Analysis	1275 (5.96)	1022 (-43.38)	1358 (40.63)	1182 (0.00)	- 2238.27	12424.29
Profit with Sentiment Analysis	975 (-32.38)	225 (-571.76)	1043 (-11.77)	867 (-161.36)	- 82.39	16179.18

*Numbers in brackets indicate the profit made purely from shorting the stock. It is used to compare the effectiveness of classification models.

13. Limitations

For the project, we faced quite a few limitations that we were unable to resolve.

13.1. Limited Computational Power

Computational power limitation limited us primarily in our sentiment analysis. As mentioned above, we were unable to extract more data from twitter due to limited computation power. That resulted in us having to compromise data extraction to a random extraction of 30 to 40 tweets a day within the time period. Clearly, significantly more tweets will be made in a day. Hence, sentiment analysis we perform and use to train our model might not be fully representative of the public sentiments felt in the day.

13.2. Sub-optimal Trading Strategy

There are many trading strategies out there that dictate different conditions as to buy, sell or hold. Different trading strategies may work better with the different models based on the synergy between the trading strategy and the model. However, due to our lack of expertise, we are unable to distinguish which type of trading strategies would work best with which model and had only managed to utilise one type of trading strategy to measure the profits.

13.3. Bull Market Data

For classification models, what the model predicts is very dependent on what it is being trained on. In our case, the market data available is that of a bull market so the model tends to predict positives. But when the market turns into a bear market, accuracy of the models will fall.

13.4. Absence of Global Strategy

There is no one size fit all strategy that can work for all stocks. A model can have high accuracy and profit returns for a stock but perform badly for another stock.This is especially so in this project as we have included sentiment analysis as a feature in our models where the sentiment analysis are domain-specific (NASDAQ news and comments). The model will not perform as well on stocks such as S&P, DJI and FTSE.

13.5. Unaccountable for Black Swan Events

Models are unable to account for black swan events (e.g. 2007/08 Housing Bubble, COVID-19) as they occur rarely and yet have huge market disruption capabilities.

14. Future Work

In the future, we look to explore into other ways to optimise our models and expand our point of considerations:

(1) Can make use of genetic algorithms to optimize weights of neural networks as suggested in a research paper we found.
(2) Explore more effective trading strategies for buy, sell and hold in optimising profits.
(3) Explore fundamental analysis such as to include company's financial figures in financial reports in our analyzing process and long term value investment.
(4) There are a few possible reasons for the poor performance in CNN. It might be due to our hyperparameters, network architecture or the imbalance data resulting in the model predicting a lot of buy and hold, forgoing sell.We will explore the possible reasons to tune and improve CNN.

References

(n.d.). stock market prediction using ann - irjet. Retrieved April 8, 2020, from https://www.irjet.net/archives/V5/i3/IRJET-V5I3634.pdf

(n.d.). Novel Approaches to Sentiment Analysis for Stock Prediction - CS229. Retrieved April 8, 2020, from http://cs229.stanford.edu/proj2018/report/72.pdf

(2019, August 10). (PDF) Stock Market Prediction Analysis by Incorporating Social and News Opinion and Sentiment Retrieved April 8, 2020, from https://www.researchgate.net/publication/331037730_Stock_Market_Prediction_Analysis_by_Incorporating_Social_and_News_Opinion_and_Sentiment

(2019, June 26). (PDF) A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis Retrieved April 8, 2020, from https://www.researchgate.net/publication/333602124_A_Comprehensive_Study_on_Lexicon_Based_Approaches_for_Sentiment_Analysis

(2017, May 24). LSTM Neural Network with Emotional Analysis for Prediction of Stock Price Retrieved April 8, 2020, from http://www.engineeringletters.com/issues_v25/issue_2/EL_25_2_09.pdf

(2018, April 28). (PDF) Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach Retrieved April 8, 2020, from https://www.researchgate.net/publication/324802031_Algorithmic_Financial_Trading_with_Deep_Convolutional_Neural_Networks_Time_Series_to_Image_Conversion_Approach

(2017, August 17). What is LightGBM, How to implement it? How to fine tune the .... Retrieved April 8, 2020, from https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

(n.d.). CatBoost vs. Light GBM vs. XGBoost - KDnuggets. Retrieved April 8, 2020, from https://www.kdnuggets.com/2018/03/catboost-vs-light-gbm-vs-xgboost.html

(2018, May 28). Introduction to Machine Learning Algorithms: Logistic Regression Retrieved April 8, 2020, from https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36

(2017, December 27). Random Forest Simple Explanation - Will Koehrsen - Medium. Retrieved April 8, 2020, from https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

(n.d.). Stock Buy/Sell Prediction Using Convolutional Neural Network. Retrieved April 8, 2020, from https://towardsdatascience.com/stock-market-action-prediction-with-convnet-8689238feae3

Pandey, P. (2019, November 8). Simplifying Sentiment Analysis using VADER in Python (on Social Media Text). Retrieved from https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f