Predicting the S&P 500 with LSTMs and GloVe

Simon Fraser University
Department of Computing Science
Burnaby, BC V5A 1S6

Winfield Chen
Computing Science
wca88@sfu.ca

Khizr Ali Pardhan
Cognitive Science
kpardhan@sfu.ca

David Pham
Computing Science
dpa35@sfu.ca

Abstract

In this paper, we describe how we created a feature rich dataset composed of GloVe word vectors and stock prices to feed as input into different Long Short Term Memory (LSTM) networks to predict the stock prices of 5 stock tickers with a focus on S & P. Our preprocessed primitive features consist of textual reddit data from r/wallstreetbets via Pushshift and hourly stock prices from Wharton Data Research Services. We outline the data extraction, transformation, and aggregation process of our datasets, which is then used as input into our LSTMs. Experimental results explore the effects of aggregation choices on our prediction and demonstrate the importance of data granularity and LSTM architecture. Our analysis suggests that additional natural language processing information from r/wallstreetbets permits our LSTM(s) to identify more meaningful patterns in our dataset to predict S&P with greater accuracy than standard models.

1 Introduction

Early stock market prediction research was based on two prominent theoretical frameworks, random walk hypothesis and the Efficient Market Hypothesis (EMH)[1], [2], [3]. The former states that the market evolves according to a random walk that does not move in any discernible pattern rendering past performance a poor indicator for future performance. It even goes as far as to boldly proclaim that both fundamental analysis and technical analysis are fruitless attempts to impose order on chaos. EMH, on the other hand, dictates that stock market prices are largely driven by new information rather than past prices. Together they constitute that since stock prices are driven by unpredictable news, which follows a random walk pattern and is therefore difficult to predict consistency with more than 50% accuracy [4]. A landmark paper by Bollen et. al. [5] explores this further by demonstrating that it is not news that drives the stock market, but rather the public sentiment around the news that shapes their collective decision making. Like most behavioral economics theories, this is rather intuitive and digestible. Our research extends these principles in the pursuit of finding some combination of public sentiment data and regression models with the ability to predict a set of economic indicators.

The simultaneous rise in social media and discount brokerages has spurred the proliferation of online communities dedicated to investing and trading on the stock market. The most popular of these communities is the subreddit r/wallstreetbets (WSB). At over 1 million active subscribers, it is the fourth most popular subreddit at this time. Bloomberg Businessweek recently featured the subreddit in an article describing the message board's ability to reshape the options market and uncanny ability to push prices, at least for the short term. While initially incredulous, veteran traders and derivatives strategists no longer deny the effects of small-investor enthusiasm. With such a large following and arguable influence, the question arises as to whether the community's sentiment can be successfully used as a predictor of stock performance.

To develop our hypothesis, we underwent three distinct phases: Creating a feature rich dataset, selecting an underlying model, and iterative experimental testing and analysis.

Dataset Creation. The composition of our dataset has two main arteries with one extending from reddit (sentiment data) and the other extending from Wharton Research Data Services (WRDS) for stock prices. To pull WSB from reddit, we used Pushshift and Google's BigQuery Platform. Once we had our raw reddit data, we used GloVe to translate them into document word vectors, which we then concatenated stock prices from Apple, Amazon, Boeing, SPY (S & P), and Tesla. This process generated two final datasets, with 25 and 300 dimensional word vectors respectively. We used the 25 dimensional dataset due to computational resource limits and leave the 300 dimensional dataset for future work.

Model Selection. In parallel with creating our feature rich dataset, we ran comparative tests on a variety of popular ML algorithms with a bare stock price dataset. Ultimately, we chose to use a Long Short Term Model as our underlying model. We choose LSTM layers for two main reasons; the sequential nature of our dataset and the hyper configurability of its architecture and parameters.

Iterative Experimental Testing and Analysis. Once we had our dataset and selected our model, we began to train our models to determine whether sentiment data improved prediction accuracy. This iterative experimental phase tested a range of LSTM architectures, hyper parameter tuning, and varying degrees of data granularity to assess our models ability to predict changes in S&P over time. Our results indicate that the prediction accuracy of our model is significantly improved when a stacked LSTM is paired with a dataset with finer granularity and lower information loss.

2 Approach - Methods employed and design choices

2. 1 Dataset Creation.

Stock Price Data from Wharton Research Data Services (WRDS)

WRDS, a division of the Wharton School of Business at the University of Pennsylvania, is a highly-regarded source in academic research and has a dedicated quality control analyst at the NYSE. Once granted access to their database, we created an hourly stock price dataset from their consolidated Millisecond Trade and Quote (TAQ) database. Our dataset consists of hourly stock prices for Apple, Amazon, Boeing, SPY (S&P), and Tesla from Jan-01-2017 to Nov-31-2019. The selection criteria was for stocks with the most mentions on WSB for 2019 with Amazon and SPY far surpassing the other 3.

Pushshift / Google BigQuery

Pushshift is a computational tool for social media data collection, analysis, and archiving platform that collects Reddit data and makes it available to researchers. Google BigQuery Platform is an integrated web service that permits interactive analysis of massive datasets. Its free tier allows 1 TB of free data-processing per month setting a size constraint on our dataset. Using BigQuery, we downloaded a textual dataset for r/wallstreetbets from Jan-01-2017 to Nov-31-2019. This time frame was selected to mitigate the stock volatility effects of the unforeseen coronavirus pandemic.

GloVe

Global Vectors for Word Representation (GloVe) is a project by the Stanford Natural Language Processing Group. It is an unsupervised learning algorithm for obtaining word vector representations from a word corpus. The GloVe project hosts several pre-trained word vector datasets from different corpora which are the largest publicly-available word vector datasets in existence. Comments on WSB often use niche terminology common to other social media platforms. If a word is not in the dataset's vocabulary, the vector to which it is assigned may not reflect its meaning. Therefore, it was important to choose large datasets gathered from online discourse such as those provided by GloVe.

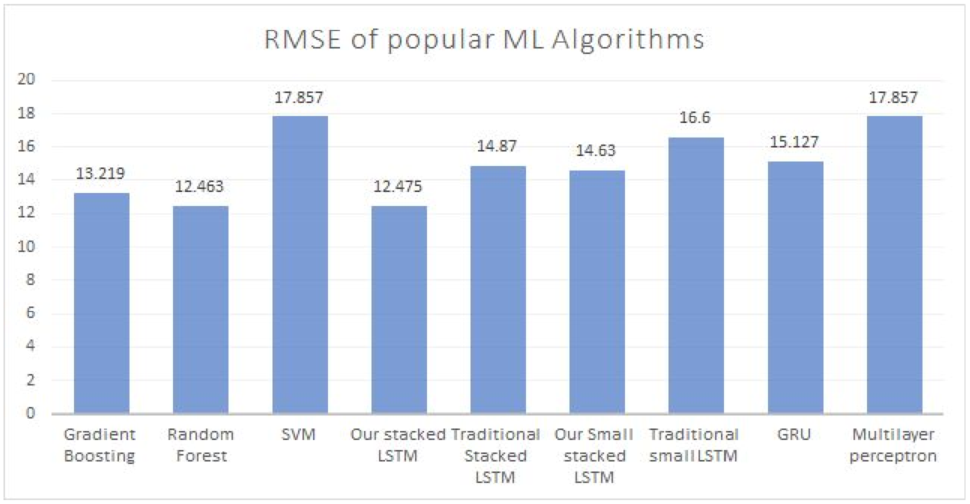
GloVe word vectors also possess several beneficial properties which enable the use of meaningful vector arithmetic. The Euclidean distance or cosine similarity between two vectors encodes similarity in meaning between the corresponding words. The vector differences in GloVe represent relationships between the meaning of words. For example, *king* - *man* + *woman* = *queen*, encoding analogies, and *cat* is closer to *dog* than *rabbit*, encoding similarity. These rich algebraic properties made GloVe the best choice for our analysis; comments with similar meanings would have similar vectors, and there would be an axis in the vector space of greatest correlation with each stock price (and a hyperplane of greatest separation orthogonal to that axis) which our models could find.

We used a Python library known as Magnitude to query two of these datasets, GloVe Twitter (2 billion tweets containing 27 billion tokens, with a vocabulary size of 1.2 million) for 25-dimensional word embeddings and GloVe Common Crawl (web archive containing 840 billion tokens, with a vocabulary size of 2.2 million) for 300-dimensional word embeddings. We embedded each comment by averaging the word vectors of each word in the comment to form a document vector. This generated two embeddings, a 25-dimensional embedding and a 300-dimensional embedding using the respective GloVe datasets.

Generating the embeddings is a CPU-bound task which takes several days on a single machine yet is trivially parallelizable in both cases. To speed up this computation, we parallelized the task with ~60 "idle" Computing Science Instructional Laboratory (CSIL) machines each with a 12 core CPU. The posts were equally partitioned by their index in the WSB dataset. Each core was assigned a partition of posts to embed. Both the WSB and GloVe datasets were distributed to each machine via the CSIL network file system. In addition to the standard document vectors, we appended score, gildings (awards for the author purchased by readers of a post), and word count of each post as additional features.

The outputs were gathered from each machine and concatenated to form a featurized dataset for both dimensional cases. The 25-dimensional featurized dataset was then averaged over hours and days to create two aggregated datasets of smaller size which were then joined with hourly and daily stock ticker prices respectively for our selected stocks from WRDS to form two final 25-dimensional input datasets for our models, one by hour and one by day. The 300-dimensional featurized dataset was joined with the most recent trade price of SPY as of each post. Since the size of this high-dimensional and unaggregated dataset is large, we leave it for future work and proceed only with the 25-dimensional hourly and daily datasets as inputs.

2.2 Model Selection



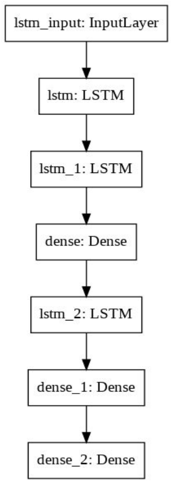


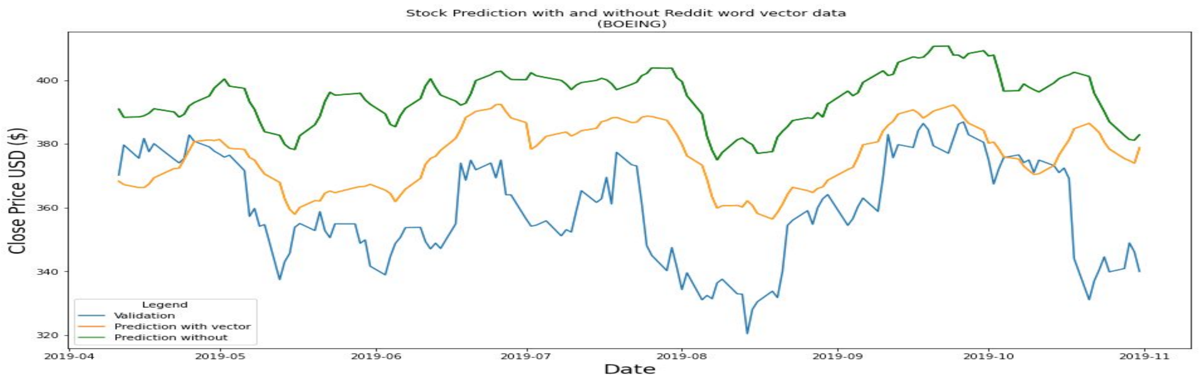
Figure 1: The left box displays a comparison between several machine learning algorithms in a comparative prediction accuracy run. We used our hourly aggregate data set, without GloVe vectors, as we assumed relative performance would remain consistent. Our final architecture is displayed on the right with a final output layer with 1 dimension. Each node represents a stock price prediction for SPY, using AAPL, AMZN, BA, and TSLA.

Except for the random forest regressor from scikit-learn, LSTM outperformed all other ML algorithms tested above due to the sequential nature of our dataset. We ultimately chose to use LSTMs over random forest for the project because we believed that their slight disadvantages in our initial test runs would be offset with the configurability of their architecture and hyper parameters. Hyper parameter tuning revealed several insights. With our two criteria, loss and training time, we compared activation functions ReLu and tanh. There was a trade-off between computational time and loss between the two. ReLu sped up training time at the cost of slightly less accurate predictions and the opposite was true for tanh. Additionally, we found that adding a dense layer between a larger LSTM stack and a small LSTM stack significantly increased prediction accuracy. This suggests that the additional stacks gave the overall model more weight adjustment freedom to subscribe to different features in the dataset.

2.3 Iterative experimental testing and analysis

To improve the portability of our code for collaboration, we chose to use Google Colab as our development environment coupled with TensorFlow libraries. This increased our productivity by reducing unnecessary library compilation, minimizing computation time, and alleviating environment compatibility issues. We ran 3 general experiments iteratively, tuning hyper parameters with each iteration, for 5 different stocks to assess our model's ability to use stock price and NLP data as indicators for economic stock value. The three experiments were formatted as follows: next day's opening price prediction using a timestep of 60 days with each time step containing aggregated daily information, next hour's opening price prediction using a timestep of 20 with each time step containing aggregated hourly information, and next day's opening price using a timestep of 24 with each timestep containing aggregated hourly information. The feature space had 33 dimensions and the number of samples ranged from 1,000 to 20,000 depending on the aggregation step size. Sequence padding with zero vectors was added when necessary and our loss function was mean squared error. We also iteratively increased the granularity of our NLP data as we had more successful runs.

3 Experimental results



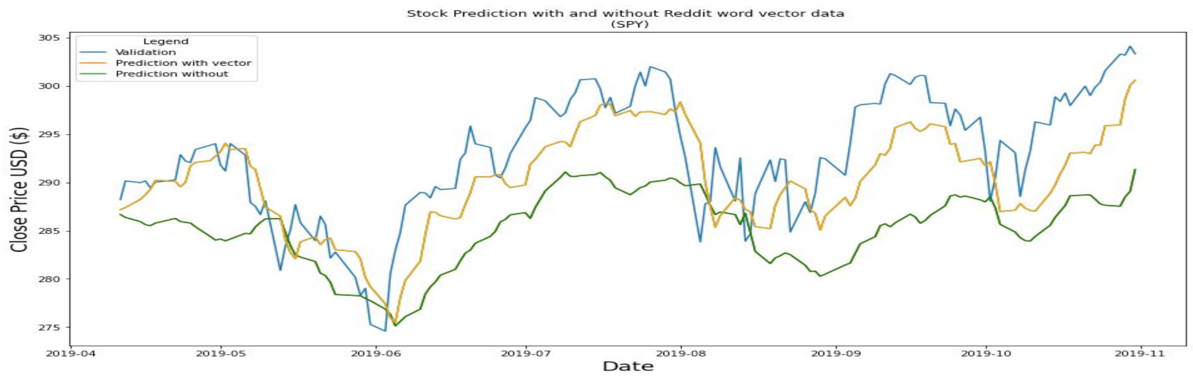
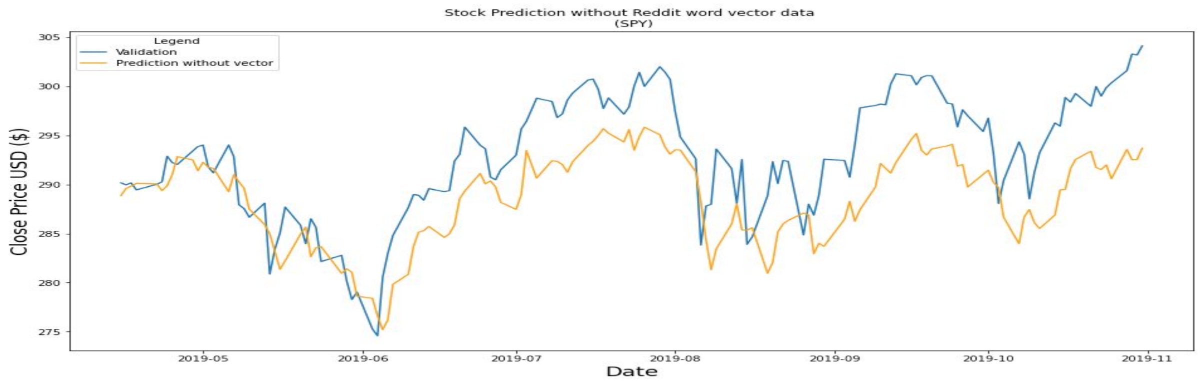


Figure 2. Next day stock prediction accuracy for SPY (bottom picture - most mentioned stock in WSB dataset) and

Boeing (top picture - least mentioned stock in WSB). Models were trained with daily aggregated WSB NLP data and opening price data. Parameter configuration: timestep size of 60, a batch size of 16, and an adam optimizer with default learning rate of 0.001.

Overall, we found that our models were able to better predict stocks that had a higher reference count in WSB reddit. Our models predicted SPY and AMZN price evaluation with greater accuracy compared to AAPL, BA, TSLA. Although LSTMs without NLP data were able to capture run patterns, they weren't nearly as sensitive to dips and spikes compared to their NLP counterparts. They displayed a conservative behaviour falling short on dips and climbing low on spikes. To pull more mileage out of our data, we ran the same experiments with increased granularity over our WSB data set. Instead of aggregating WSB NLP over a day, we chose to aggregate over the hour to minimize loss of information and variance. We suspect the loss of the latter hindered our models from responding appropriately to spikes and dips.



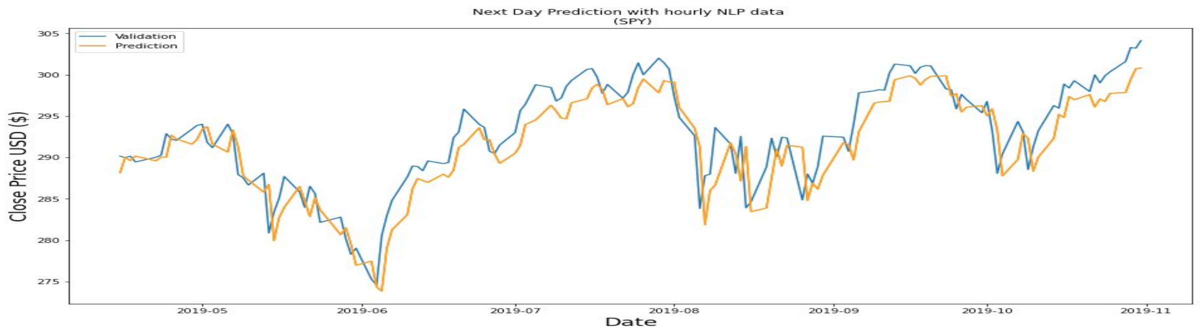


Figure 3. Next day's opening price prediction for SPY using hourly data. One LSTM model was trained with hourly aggregated WSB NLP data + WDRS hourly stock prices (bottom picture), while the other was trained with only hourly stock price data (top picture). For hours outside of market hours, we appended either the market opening price or the market closing price. Both models were trained with parameters: timestep of 24 for each hour in a day, batch size of 16, and an adam optimizer with a learning rate of 0.001.

With a higher degree of granularity, our prediction accuracy went up quite dramatically. We see that our model's ability to follow spikes and dips increases significantly. Because the architecture of the LSTM remains largely identical, we attribute this increase in accuracy to the shape and pattern of data being fed into our LSTMs. Instead of using the previous 60 days as a series to train our model to identify patterns in stock movement, we instead use the market movement in the previous day to predict the next day. We believe that this is an easier pattern for our LSTMs to identify correctly due to the reduced variation in stock prices hourly compared to daily.

3 Conclusion

Our results did not refute the hypothesis that WSB is a predictor of stock market performance. The addition of NLP data in the form of averaged word vector representations of WSB comments increased the predictive accuracy of our LSTMs. This suggests that our LSTMs were able to learn axes in the vector space of WSB comments which correlated with movements in each stock price (or dually, hyperplanes of separation). The changes in these vectors with respect to time may also have been a source of information for our LSTMs. Our experiments show that the addition of WSB comments has a consistent net positive impact for the accuracy of our LSTMs, and is beneficial from our empirical evidence. In fact, our experiments show that the finer the granularity of WSB data and therefore the larger the WSB data, the better our LSTMs perform suggesting that data aggregation results in the loss of crucial information. For example, averaging posts over a day results in a reduction of information and variance and it may be vectors from this variance that are needed to train our models to accurately predict spikes and dips. Volatile posts may be more telling of stock performance compared to their lukewarm counterparts. The ability of our NLP trained models to predict spikes and dips is even more apparent when compared to models that were trained with only stock data. Models trained with only stock data were hindered to confidently follow sudden spikes or dips displaying a conservative lag behaviour.

However, this does not establish a causal relationship between WSB sentiment and stock market performance; WSB may very well be an influence in the market, or it may just be an extremely efficient news and sentiment aggregator. Stocks which were frequently mentioned in WSB were more accurately predicted than stocks which were less frequently mentioned, meaning the information from WSB used by our model in this time period is company-specific (as opposed to market-wide) and indeed exists and is correlated with stock market performance, at least for these companies.

Our project demonstrates that NLP-augmented approaches are feasible in offline modelling. From our experience during this project, we believe that online-learning is possible for practical applications of this method in securities trading; the computational resources used for computing post vectors in our work was large only due to the sheer number of posts incurring a large up front computational cost. Independently processing a single post is trivial, so subsequent on the fly online processing of posts as they are created is not an issue. Traders, analysts, and investors already use technical indicators in an advisory role, and we believe our method, among other neural network NLP approaches in general, are feasible and useful additions to the field of finance.

4 Future work.

Improvements to model architecture. There are several Interesting LSTM architectures such as Convolutional LSTMs [7], LSTMs with attention [8], and Bayesian LSTMs [9] that may perform equally well or even better. We did not seem to have issues with over fitting, so we chose not to use kernel regularizers, dropout, or recurrent dropout layers. We were interested in recurrent dropout to mitigate overfitting, however it would prevent us from using CUDA library extensions. Using CUDA extensions drastically reduces training time providing the freedom to explore a larger set of hyper parameters and LSTM configurations. Keras uses Glorot uniform, *Xavier initialization*, as the kernel initializer by default, which was satisfactory. Future iterations should explore additional initializations [10]. Due to computational constraints and the size of the dataset, the permutations of LSTM architecture and hyperparameter setting explored were relatively shallow.

Improvements to dataset. Additional features such as dividends, volume, business sector, and technical indicators, namely the Relative Strength Index, could all be potentially beneficial since they are all indicators that our model could pick up patterns from. Averaging the constituent word vectors of a document to form a document vector is a primitive approach which loses information about word order and importance. Future work should explore the use of a trainable neural network document vector model such as Doc2Vec which extends word vectors to documents in a more sophisticated fashion. In addition, clever effective dimensionality reduction [6] techniques for word embeddings may be a reasonable way to decrease computational cost. A future document model could be trained specifically on finance-related news and social media and could incorporate special handling of stock tickers such as "TSLA". Such a finance-specialized model could provide better results. Finally, using the 300-dimensional unaggregated large dataset we set aside during our current work is an exciting proposition.

5 References

[1] Fama, E. F. (1991) Journal of Finance 46, 1575–617.

[2] H.Cootner, P. (1964) The random character of stock market prices.

[3] Fama, E. F. (1965) The Journal of Business 38, 34–105.

[4] Qian, Bo, Rasheed, & Khaled. (2007) Applied Intelligence 26, 25–33.

[5] Bollen, Johan, Mao, Huna, Zeng, Xiao-Jun (2010) Twitter mood predicts the stock market.

[6] Rauna (2017) Simple and Effective Dimensionality Reduction for Word Embeddings.

[7] Shi (2015) Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.

[8] Kim, Yang (2019) Financial series prediction using Attention LSTM.

[9] Fortunato, Blundell, Vinyals (2017) Bayesian Recurrent Neural Networks.

[10] Talathi, Vartak (2016) Improving performance of recurrent neural network with ReLu nonlinearity.

6 Contributions

David Pham	**Winfield Chen**	**Khzir Ali Pardhan**
Project work: - Background research - Created hourly stock price dataset from WRDS - LSTM architecture research - Implemented opening price prediction from previous days LSTM - Implemented hourly prediction using aggregated hour LSTM - Implemented opening day prediction with varied hour time step LSTM **Report:** - Abstract - Introduction - Methodology - Results - Conclusion - Edits	**Project work:** - Group name - Background, stock, & NLP research - Queried the Pushshift database for WSB posts using Google Cloud Console's BigQuery - Created GloVe Vectors using distributed computing - Hourly and daily aggregation of 25-d glove vectors and joining with resampled stock prices - Direct join of 300-d glove vectors with tick prices **Report:** - Methodology - Conclusion - Edits	**Project work:** - Created and Resampled hourly stock price dataset from WRDS - Conducted regression with several ML algorithms, and provided comparative analysis for model selection - Provided past keras code for daily stock prediction with LSTM & GRU - LSTM research & hyperparameter tuning - Attempted variable length LSTM without removal of any data (no resampling) **Report :** - Methodology - Conclusion - Future Work - Edits - Formatting