A drawing of a cartoon character

Description automatically generated

Jeff Levesque

https://github.com/jeff1evesque | Final Project

Data Science: Portfolio

2023

Table of Contents

[Introduction 2](#_Toc123369292)

[FIN 654 2](#_Toc123369293)

[Overview 2](#_Toc123369294)

[Learning Goals 2](#_Toc123369295)

[Data Collection 3](#_Toc123369296)

[Discovered Patterns 3](#_Toc123369297)

[Alternative Strategies based on Data 7](#_Toc123369298)

[Plan of Action 8](#_Toc123369299)

[IST 664 8](#_Toc123369300)

[Overview 8](#_Toc123369301)

[Learning Goals 8](#_Toc123369302)

[Data Collection 9](#_Toc123369303)

[Discovered Patterns 9](#_Toc123369304)

[Alternative Strategies based on Data 12](#_Toc123369305)

[Plan of Action 13](#_Toc123369306)

[IST 736 13](#_Toc123369307)

[Overview 13](#_Toc123369308)

[Learning Goals 14](#_Toc123369309)

[Data Collection 14](#_Toc123369310)

[Discovered Patterns 15](#_Toc123369311)

[Alternative Strategies based on Data 21](#_Toc123369312)

[Plan of Action 21](#_Toc123369313)

[Follow-up Portfolio 22](#_Toc123369314)

# Introduction

Do the opinions of Financial Analysts on Twitter impact market sentiment and volatility? There are many financial analysts on twitter offering investment advice to millions of people daily. These analysts also employ media outside of Twitter, including TV shows, blogs and newspaper columns. However, Twitter provides closer to real-time sentiment distribution than other media. For example, Jim Cramer is arguably the most famous financial analyst today, having over 1.1 million followers on Twitter, and host of the popular Mad Money television series. Because audience members often look to financial analysts such as Jim Cramer for market advice, the natural question arises of whether their advice impacts the investment decisions of followers, and in turn, market volatility?

# FIN 654

## Overview

This project focused on a fixed portfolio consisting of companies that have been recently hacked or breached. In general, efforts in this project attempts to put into place a frame of analysis to determine the impact to stock values. However, follow-up studies could determine whether companies had or lacked business acumen to mitigate or minimize the effect of such occurrences. Further analysis could also indicate whether to avoid volatile stocks that have been hacked or breached, and whether different sectors are more impacted.

## Learning Goals

Learning goals for this course entail introduction to methods and tools useful in decision-making in the financial industry including trading analytics and execution algorithms[[1]](#footnote-1). In this final project, more emphasis was placed on identifying patterns via statistical analysis using R programming within the Shiny visualization dashboard[[2]](#footnote-2). Concepts such as variance, (partial) autocorrelation, seasonality, trends, and general volatility were measured, and could be used as decision factors for downstream statistical modeling[[3]](#footnote-3). Additionally, a General Pareto Distribution[[4]](#footnote-4)[[5]](#footnote-5), and efficient frontier[[6]](#footnote-6) was constructed for a given portfolio to measure risk for the chosen portfolio.

Various concepts learned from other IST courses helped expand traditional market analysis by using ARIMA and RNN (LSTM variant) modeling. An R interface to python via the reticulate package[[7]](#footnote-7) was utilized. This ultimately allowed python code to be invoked from R, providing a comparison between ARIMA and LSTM through the same overall R Shiny dashboard.

### Data Collection

Three different datasets were ultimately used for the overall project:

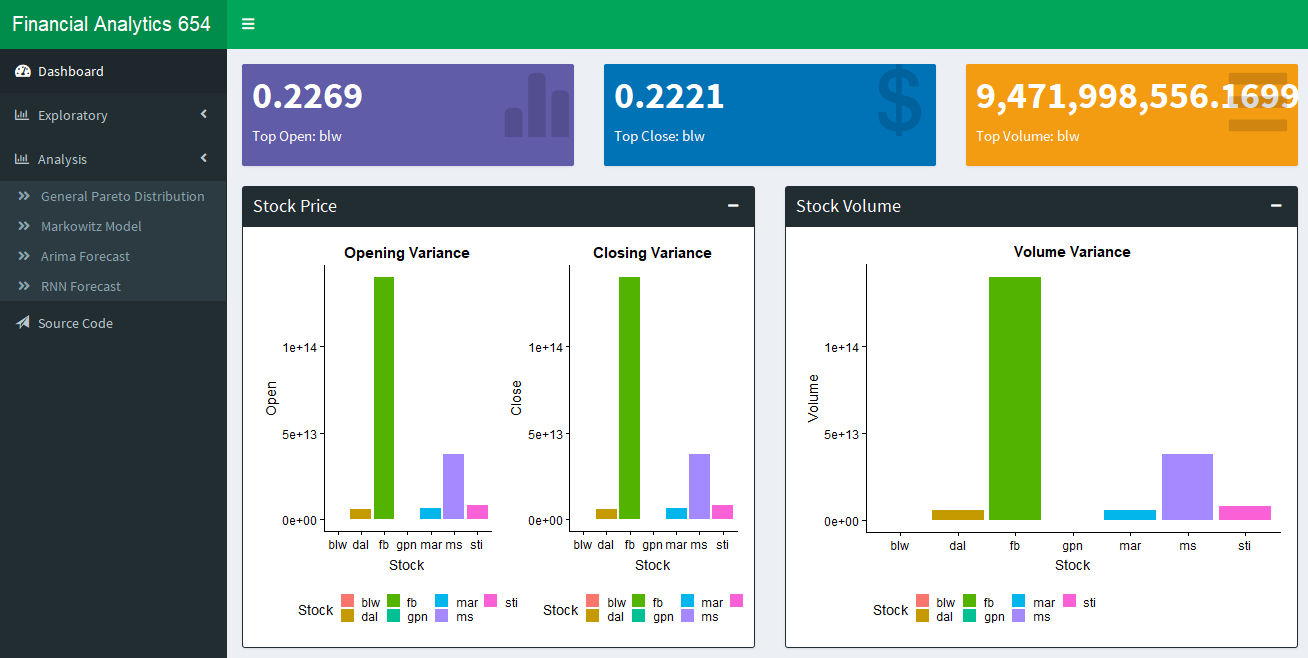
* security[[8]](#footnote-8): csv files listing company names that have been breached or hacked
* stock-exchange[[9]](#footnote-9): csv files listing ticker symbols and associated company names

The security and stock-exchange dataset were joined to produce a list of stock tickers that have been breached or hacked. This list of tickers was passed into the quandl api[[10]](#footnote-10) to produce the symbol dataset:

* symbol[[11]](#footnote-11): csv files of each ticker, containing daily open, close, low, high and overall volume

### Discovered Patterns

A plot was created showing variance for each ticker price and volume within the portfolio:



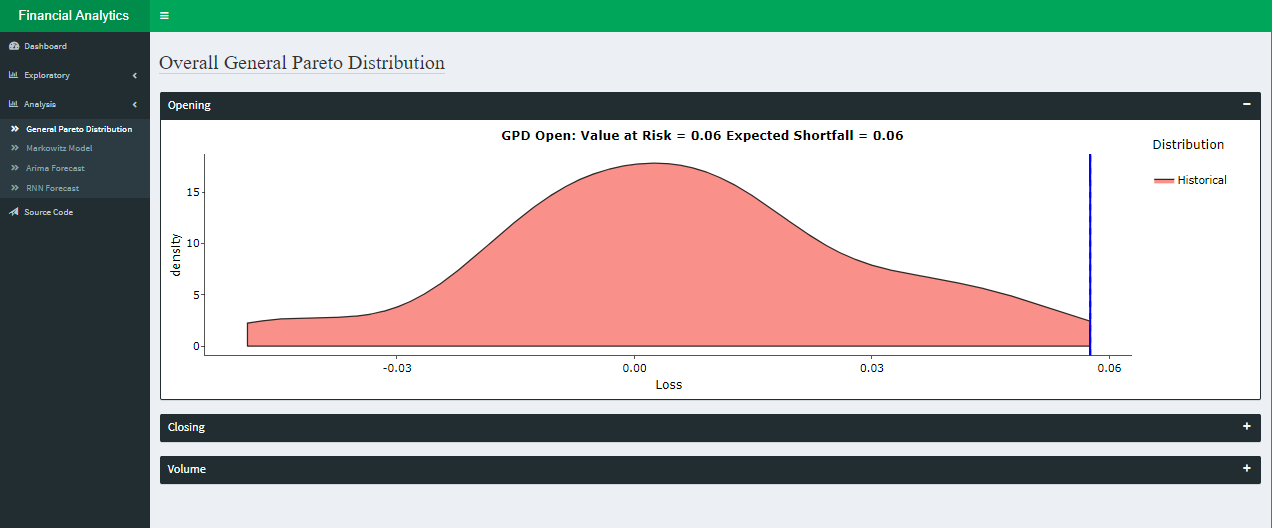
Since the above is an aggregated variance, it’s difficult to draw any conclusions. A complementary variance over time for a given ticker may contribute more meaningful insight. One could speculate that BLW is the least risk-averse overall ticker. Meanwhile, FB has greater volatility, which may provide more opportunity for gains (or loss at an associated risk).

Next, a decomposed timeseries[[12]](#footnote-12) was constructed for the overall portfolio:



The decomposed time series did not exhibit any seasonality, nor residual/noise. Rather, the trend seems to mirror the overall portfolio value. Despite successive modeling (i.e. ARIMA) being conducted for the overall portfolio, (partial) autocorrelation (P)ACF was not performed to help identify most optimal AR and MA arguments reducing seasonality. This was largely because the earlier decomposed timeseries did not indicate seasonality, and limited time within project deadline. However, (P)ACF was performed[[13]](#footnote-13) on individual tickers within the portfolio (showcased on Shiny dashboard), though no timeseries model were trained at the ticker level.

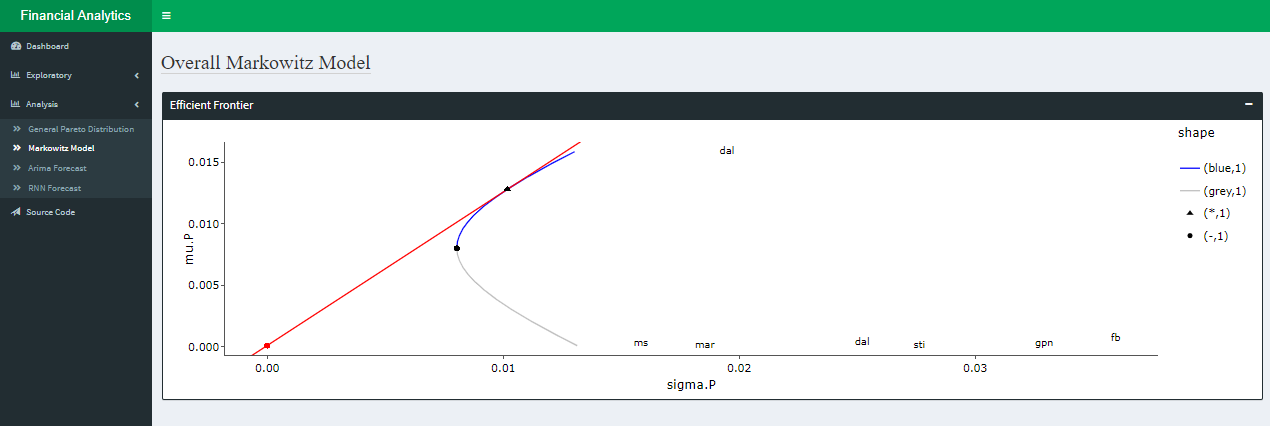
Next, a general pareto distribution (GPD) was computed[[14]](#footnote-14) with a 95% threshold for the overall portfolio. Both the value at risk (VAR) and expected shortfall (ES) exceed this threshold:



Generally, 95% of the time, loss will not exceed 0.06. Conversely, less than 5% of the time, the minimum loss of the portfolio will be 0.06. Since this portfolio was simplified with one share per ticker/stock[[15]](#footnote-15), the overall risk was much smaller than an ordinary diversified portfolio.

**Note:** GPD, VAR and ES code was supplied as course material[[16]](#footnote-16) from the professor, and recycled for purposes of this Shiny dashboard.

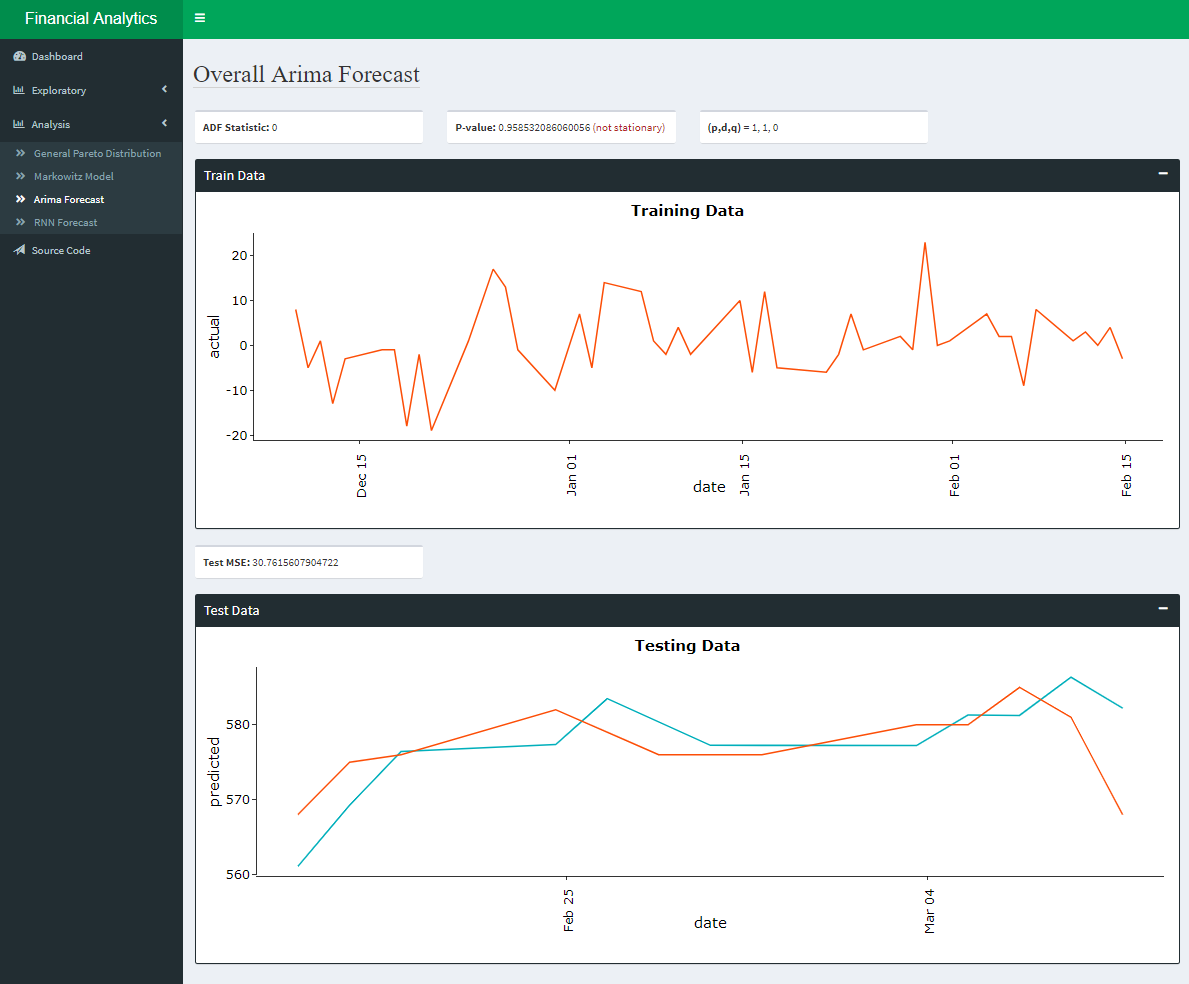
Next, an efficient frontier (EF) was created, along with the tangent Markowitz model to signify the most efficient frontier[[17]](#footnote-17):

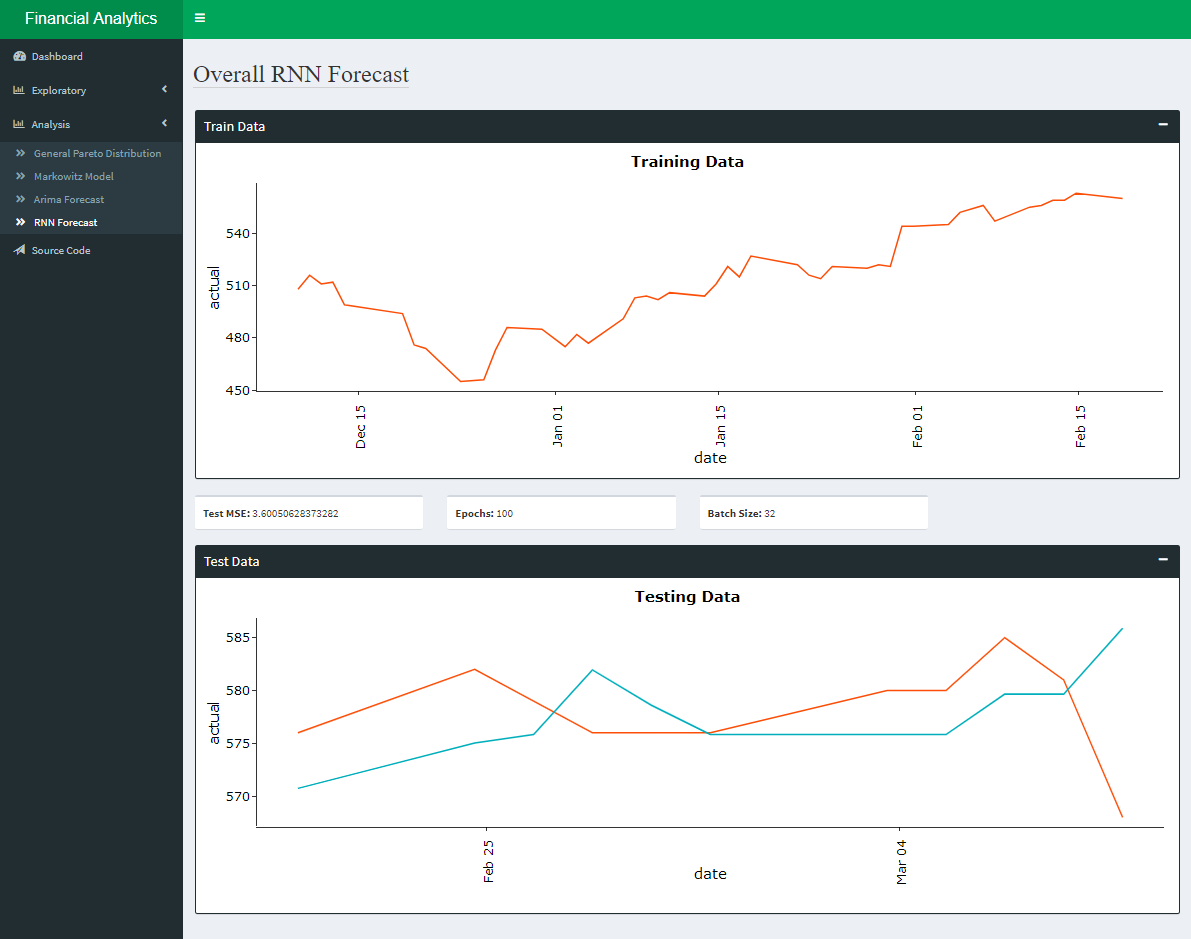


The EF is a curve for a given portfolio suggesting the most optimal rate of return for a given risk tolerance. Generally, the y-axis represents the rate of return, while the x-axis represents risk tolerance (i.e. volatility). The Markowitz model is a tangential line that represents the most optimal point within the efficient frontier with the least amount of risk.

**Note:** efficient frontier[[18]](#footnote-18) and Markowitz model[[19]](#footnote-19) code was supplied as course material from the professor, and recycled for purposes of this Shiny dashboard.

Next, an ARIMA and LSTM model were trained on the overall portfolio:





### Alternative Strategies based on Data

One alternative strategy would be to replace all R code with Python. This standardization would allow greater possibilities than a laptop digesting a few adhoc CSV files. Generally, python could be integrated into a stream analysis application[[20]](#footnote-20). For example, a spark-based application could check whether streaming ticker data follows a normal distribution, which is one of the assumptions[[21]](#footnote-21) of the efficient frontier. Ticker symbols that pass a set of criteria through the streaming application, would become available to be assigned to various portfolios. These portfolios could undergo EF and Markowitz modeling. There are many great sources[[22]](#footnote-22) showcasing a python variant[[23]](#footnote-23), which could be implemented as a sliding window[[24]](#footnote-24) operation. While a streaming application could be executing on an interval as low as seconds, it is possible to focus on a more delayed portfolio. Specifically, a daily EF could be computed, which would define preferred weights of individual tickers in a portfolio. This overall weighted portfolio could later define an overall timeseries model to help project anticipated returns of a given portfolio. Again, (P)ACF could be computed via a stream application, to help train an ideal ARIMA model.

To sum, the niceties of R and dashboarding are great tools for exploratory analysis. However, more robust modeling often entails larger volume and velocity of data, as well as associated triggers to help guide actions when conditions are satisfied.

## Plan of Action

The desire of this project was to assess whether stock prices for companies recently hacked or breached were significantly impacted, and whether to avoid them in a portfolio. Additionally, one could perform a case study, identifying corresponding companies had or lacked business acumen to mitigate or minimize the effect of such occurrences. However, this project lacked rigor on this topic. Instead, focused largely on engineering the overall portfolio analysis for a collection of companies that were hacked or breached through the shiny dashboard.

In general, modern portfolio theory (MPT) could be implemented as a series of case studies leading up to an overall portfolio analysis. For example, individual tickers could be assessed whether the stock price declined during period of hack/breach. If a significant decline is found, then MPT could be implemented by means of EF/Markowitz to determine an adjusted portfolio comparison. For the first case, a portfolio could consist of mostly “safe” stocks including a single (or few) ticker associated with a hack or breach. On a second case, the portfolio would consist of just the “safe” stocks. A comparison could be made between the two using the EF and Markowitz models. Specifically, a comparison could be made on the associated risk value of the two EF models. Lastly, the same ARIMA or LSTM timeseries models could be created, based on the overall weighted portfolio. These models could be inferenced for an arbitrary future prediction to assess which model is more profitable. Finally, the profitability could be weighed against the risk from the EF/Markowitz, as well as earlier VAR approach.

# IST 664

## Overview

In this project two rules-based classifiers supplemented a conversational agent. When a user provides input text, a classifier predicts whether the string is a question. If a question is detected, then the trained conversational agent is engaged. However, if the agent response score is below an accepted threshold, then the response omitted from the user. Instead, a secondary classifier is initiated. This classifier returns the best StackOverflow channel for the original provided question.

## Learning Goals

Learning goals for this course entail linguistic and computational aspect of natural language processing technologies[[25]](#footnote-25). This includes the use of Natural Language Toolkit (NLTK), tokenization and parts of speech (POS), as well as rudimentary regular expressions (REGEX) to clean training data. All these concepts were implemented in this project including additional concepts not covered in the course. For example, Hadoop via MongoDB (mapreduce training data), as well as deploying recurrent neural networks (RNN). Further, virtualization by means of vagrant, docker, as well as AWS Cloud were all implemented to gain a better understanding of how to deploy a trained minimal viable product.

### Data Collection

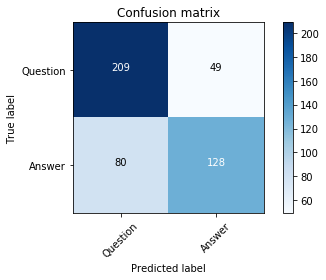
Three different datasets were used to generate respective models:

* QuestionAnswerCMU[[26]](#footnote-26): question-answer pair to train question classifier
* Reddit[[27]](#footnote-27): json data used to train RNN based NMT chat agent
* StackOverflow[[28]](#footnote-28): list of postings grouped by channels used to train classifier to label questions

Each of the above datasets were download and later referenced via CLI or Jupyter notebooks, while the Reddit data was also loaded into MongoDB[[29]](#footnote-29). Using MongoDB, allowed the map-reduce[[30]](#footnote-30) to be leveraged prior to RNN/NMT train[[31]](#footnote-31). Specifically, Reddit json data was filtered on relevant keys with a mapper[[32]](#footnote-32), then grouped into a post/comment association[[33]](#footnote-33) through a common parent\_id and id key[[34]](#footnote-34). A reducer removed irrelevant records with regular expression. The returned map-reduce query would ultimately be tokenized into numerous post/comment pairs used by the RNN/NMT train[[35]](#footnote-35).

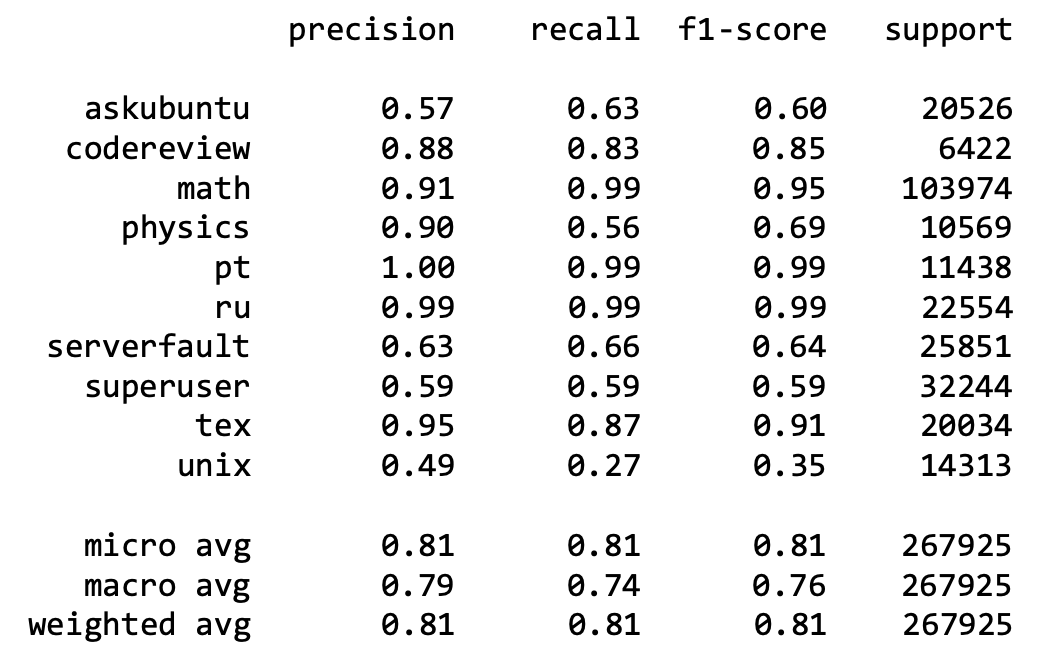
### Discovered Patterns

A random forest classifier was created for the QuestionAnswerCMU dataset, and was able to perform roughly 72% accuracy:



Next, multiple classifiers were trained to determine what StackOverflow channel a sentence is most related to. Different Naïve Bayes techniques were implemented including, unigram Boolean, and POS tagging[[36]](#footnote-36) into bag of words list. Further, SkLearn with NLTK tokenizer[[37]](#footnote-37) were incorporated into random forest classifiers, again using unigram and bigram Boolean.

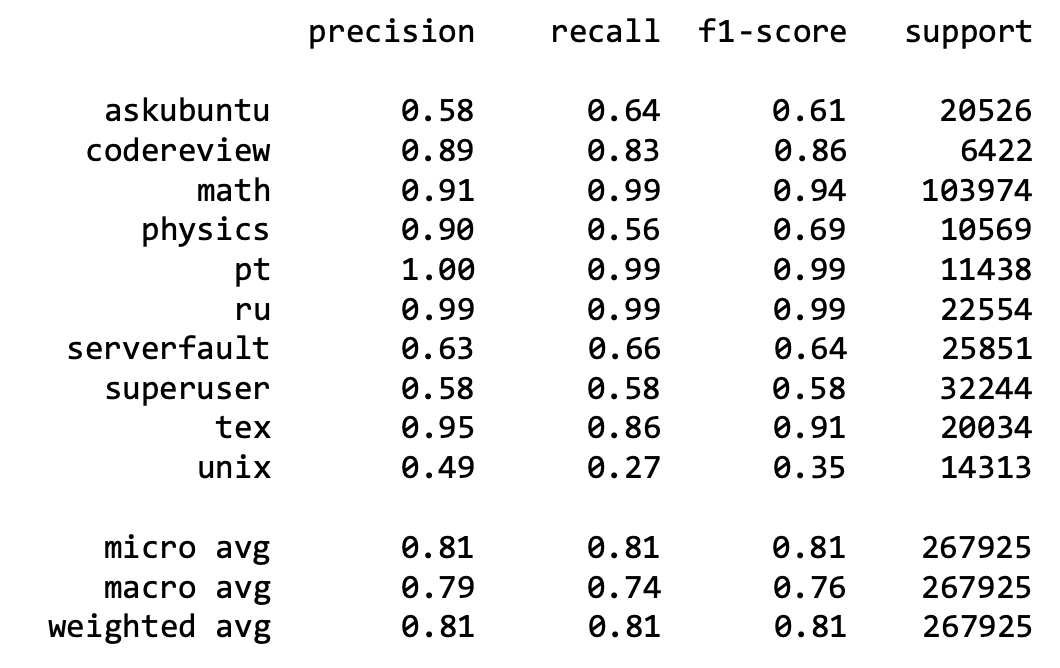
In general, the Unigram Boolean generated reasonable Naïve Bayes model with a cross validation mean accuracy of 81.2%:



A screenshot of a social media post

Description automatically generated

Likewise, the unigram Boolean produced reasonable random forest model, with just a reduction in accuracy of 0.01% compared to earlier Naïve Bayes approach:



A screenshot of a social media post

Description automatically generated

### Alternative Strategies based on Data

Creating a NMT/RNN based chat agent required substantial amount of train data. Due to limitations of compute, and cost, this segment did not produce a useful model. In general, attempting to deploy a large-scale data cluster on MongoDB was difficult. Firstly, none of the courses offered at SU provided background knowledge into AWS VPC, which contributed to cloud confusion. The following write-up segment[[38]](#footnote-38) was incorrect:

However, since five virtual machines did not have elastic ip addresses, their public ip addresses constantly changed upon machine startup. This posed challenges, and manual intervention were required to ensure network security groups were properly defined. Specifically, each mongodb blocked all incoming ports except other mongodb instances within it’s sharded system. Due to the complexity, this infrastructure was abandoned, but corresponding scripts to deploy this system is still available and version controlled.

In truth, public/elastic IP addresses were not needed to connect each mongodb nodes with one another. Rather, fixed private IP addresses would have sufficed using the default local route[[39]](#footnote-39), referenced within corresponding security groups. However, it was valid statement regarding prohibitive cost of running 8 virtual machines8 for a school project. Thus, the overall RNN/NMT training dataset was scaled back. Rather than deploying the entire 6 months of Reddit data onto a cluster of MongoDB nodes spread across AWS EC2 instances[[40]](#footnote-40), a single node was deployed on a local laptop. This same laptop was also used to train RNN/NMT models, as well as perform adhoc/on-demand inferences.

## Plan of Action

This project produced exploration into the idea of conversational agents with a supplement classifier to initiate the agent, as well as another classifier acting as a fail-safe response. Due to compute (8GB RAM, 0GB GPU), and storage (less than 200GB) limitations, training a meaningful RNN/NMT model would take too long, and/or stall on the local machine. However, numerous resources[[41]](#footnote-41) online have succeeded on the same RNN/NMT approach. Following a similar path to success, would likely involve integrating cloud resources. As mentioned above, to handle large scale Hadoop operations, more MongoDB nodes would be required. However, this would be two-folds, since the additional nodes would also store the additional data to be map-reduced into RNN/NMT train.

Despite the conversation agent shortcomings, the associated classifiers were largely successful. Perhaps a better approach is to deploy numerous specialized/supplement classifiers, with a final target of a specialized RNN/NMT model, with a default classifier if a low-quality response is returned. A workflow could be as follows:

Question Classifier 🡪 Topic Classifier 🡪 Sub Topic Classifier 🡪 Specialized RNN/NMT

When a step fails in the above workflow, the user could be re-engaged to a previous step, or given the option to end the overall workflow session. However, such an approach requires numerous models to be trained, which exponentiates the number of specialized RNN/NMT models. This would indeed require heavy cloud computing.

# IST 736

## Overview

In this project, an overall question was asked – Can Market Sentiment Predict the stock market? To address this overall question, different techniques were applied:

* Exploratory Data Analysis (EDA): topic modeling determines which stock ticker to study
* Sentiment Analysis: text corpus normalized into sentiment scores
* Granger Analysis: find significant sentiment and ticker pairs combinations
* Timeseries Analysis: train LSTM and ARIMA models on tickers filtered from Granger Analysis

While the main focus of the study was between timeseries models coupled with granger analysis, classification analysis was also performed. Specifically, signal analysis was used as a basis to predict whether TF-IDF corpus could predict positive or negative price momentum:

* Signal Analysis: determine exceeding timeseries points
* Classification Analysis: TF-IDF text corpus (X) trained against signal results (y)

Stock prices for a given ticker exceeding the upper limit threshold was binned a value 1, while points below the lower threshold was binned a value -1. In general, signal analysis define a target vector (y) for an associated TF-IDF corpus (X) during classification.

## Learning Goals

Learning goals for this course entail the following four objectives[[42]](#footnote-42):

1. Describe basic concepts and methods in text mining, for example document representation, information extraction, text classification and clustering, and topic modeling;
2. Use benchmark corpora, commercial and open-source text analysis and visualization tools to explore interesting patterns;
3. Understand conceptually the mechanism of advanced text mining algorithms for information extraction, text classification and clustering, opinion mining, and their applications in real-world problems; and
4. Choose appropriate technologies for specific text analysis tasks and evaluate the benefit and challenges of the chosen technical solution.

This final project utilizes basic concepts[[43]](#footnote-43) and methods in text mining[[44]](#footnote-44) in order to perform EDA[[45]](#footnote-45), sentiment analysis using open-source libraries, as well as reshaping data in order to perform text classification[[46]](#footnote-46). Standardized sentiment analysis[[47]](#footnote-47) is later paired with stock index/volume data if a significance is found between two time series through the granger causality test[[48]](#footnote-48). Next, compute benchmarks and conclusions are made, each helping to address what improvements can be made. Finally, a general recommendation is made whether the overall project topic should be reframed.

### Data Collection

Two different datasets were obtained using Twython[[49]](#footnote-49) and Quandl[[50]](#footnote-50) API:

* Financial analyst tweets[[51]](#footnote-51)
* Stock market index/volume measures[[52]](#footnote-52)

The index/volume dataset was determined via EDA through Latent Dirichlet Analysis (LDA) on each financial analyst set of tweets. Specifically, tickers were manually identified from LDA topic modeling were used as index/volume measures.

On a continuation of this project (see below Follow-up Portfolio), live streaming[[53]](#footnote-53) ticker data is ingested into an s3 datalake[[54]](#footnote-54) every minute of a given business day for nearly 200 ticker symbols.

### Discovered Patterns

Numerous reports[[55]](#footnote-55) and visualizations[[56]](#footnote-56) were generated. However, baseline analysis was performed to set the stage for successive measurements/analysis. Further, classification analysis was performed as an extended EDA for each financial analyst TF-IDF text corpus against standardized ticker price signals. While results were computed for numerous analysts, only a LizAnnSonders[[57]](#footnote-57) was considered. This determination was largely made because corresponding confusion matrices were only slightly less biased, after dataframe row filtering was applied.

|  |  |  |
| --- | --- | --- |
| Classification: VIX Total Volume for LizAnnSonders | | |
|  |  |  |
| Figure 24: BNB | **Figure 25: B**NB (POS) | **Figure 26:** MNB |

Note that the above classifiers only train if there are enough rows in the constructed dataframe, consisting of tweets in one column with associated sentiment scores in separate columns.

|  |  |  |
| --- | --- | --- |
| Check: VIX Total Volume for LizAnnSonders | | |
|  |  |  |
| Figure 27: BNB | **Figure 28:** BNB (POS) | **Figure 29:** MNB |

As shown in Figure 27 – Figure 29, BNB and BNB (POS) have the best combination of precision, recall and f-score. However, depending on a single execution may rely too much on chance. Thus, a k-fold[[58]](#footnote-58) implementation was utilized having n\_splits=5, repeated 750 times.

|  |  |  |
| --- | --- | --- |
| K-fold: VIX Total Volume for LizAnnSonders | | |
|  |  |  |
| Figure 30: BNB | **Figure 31: B**NB (POS) | **Figure 32:** MNB |

While the remaining k-fold model iterations did not outperform the above combinations to a significant degree (Figure 27 – Figure 29), performance of the remaining models can be reviewed separately[[59]](#footnote-59). In general, it can be said that TF-IDF tweets can moderately predict stock price momentum. Further, the chi-square was applied after the TF-IDF vectorizer, to reduce dimensionality to the top 1000 words:

|  |  |  |
| --- | --- | --- |
| Top 25 chi2: VIX Total Volume for LizAnnSonders | | |
|  |  |  |
| Figure 40: BNB | **Figure 41: B**NB (POS) | **Figure 42:** MNB |

Due to a limitation of the port stemmer, the generalized words are difficult to discern. Some commonalities between models indicate the following top stemmed words:

* jan
* twitterback
* 267B
* Uneasi
* wow
* 8662
* ack
* short

Overall, the top words for VIX are related to President Trump’s proposed $267B tariffs on all goods from China early September 2018[[60]](#footnote-60). While market drop follow, general threats on American Chinese economic tension caused overall uneasiness in the global market. Counter tariffs largely impacted U.S. farm products the following year:

|  |
| --- |
| Farm products impacted by U.S. China trade war |
| Figure 43: Effects of Chinese Tariffs on U.S. Agriculture. |

Next, a series granger causality test was implemented as a hypothesis testing construct to help identify whether one time series can forecast another. While the premise involves stationarity, future studies should take better care ensuring stationary of the corresponding stock index/volume prior to the granger test. Since earlier EDA identified LizAnnSonders as an ideal financial analyst to observe, corresponding granger p-values were obtained for just this analyst:

|  |  |  |  |
| --- | --- | --- | --- |
| P-Value: Negative Sentiment “Granger Causes” VIX Volume | | | |
|  |  |  |  |
| Figure 69: chi-test | **Figure 70:** f-test | **Figure 71:** params ftest | **Figure 72:** ratio test |

|  |  |  |  |
| --- | --- | --- | --- |
| P-Value: Neutral Sentiment “Granger Causes” VIX Volume | | | |
|  |  |  |  |
| Figure 73: chi-test | **Figure 74:** f-test | **Figure 75:** params ftest | **Figure 76:** ratio test |

|  |  |  |  |
| --- | --- | --- | --- |
| P-Value: Positive Sentiment “Granger Causes” VIX Volume | | | |
|  |  |  |  |
| Figure 77: chi-test | **Figure 78:** f-test | **Figure 79:** params ftest | **Figure 80:** ratio test |

At 95% confidence, the negative LizAnnSonders sentiment scores can be said to “granger cause” the associated VIX index. Likewise, p-values were collected between sentiment and ticker volume. However, only neutral and positive sentiment was found to “granger cause” AMZN volume:

|  |  |  |  |
| --- | --- | --- | --- |
| P-Value: Neutral LizAnnSonders “Granger Causes” AMZN Volume | | | |
|  |  |  |  |
| Figure 85: chi-test | **Figure 86:** f-test | **Figure 87:** params ftest | **Figure 88:** ratio test |

|  |  |  |  |
| --- | --- | --- | --- |
| P-Value: Positive LizAnnSonders “Granger Causes” AMZN Volume | | | |
|  |  |  |  |
| Figure 89: chi-test | **Figure 90:** f-test | **Figure 91:** params ftest | **Figure 92:** ratio test |

Now, ARIMA and LSTM models were trained to predict VIX volume:

|  |  |  |
| --- | --- | --- |
| ARIMA Distribution: VIX Total Volume | | |
|  |  |  |
| Figure 44: train (3,2,0) | **Figure 45:** test (3,2,0) | **Figure 46:** overall MSE |

|  |  |  |
| --- | --- | --- |
| LSTM Distribution: VIX Total Volume | | |
|  |  |  |
| Figure 59: train (750 epochs) | **Figure 60:** test (750 epochs) | **Figure 61:** overall MSE |

The trained LSTM VIX volume model show significant performance, yet underperforming against the ARIMA variant, as shown by corresponding MSE scores. Again, ARIMA and LSTM models were trained to predict vader sentiment scores:

|  |  |  |
| --- | --- | --- |
| ARIMA Test Distribution: LizAnnSonders Sentiment | | |
|  |  |  |
| Figure 50: negative (3,1,0) | **Figure 51:** neutral (3,2,0) | **Figure 52:** positive (0,2,0) |

|  |  |  |
| --- | --- | --- |
| LSTM Test Distribution: LizAnnSonders Sentiment | | |
|  |  |  |
| Figure 60: negative (750 epochs) | **Figure 61:** neutral (750 epochs) | **Figure 62:** positive (750 epochs) |

The associated MSE comparison indicates that ARIMA outperforms LSTM at 750 epochs:

|  |  |  |
| --- | --- | --- |
| ARIMA Sentiment MSE | | |
|  |  |  |
| Figure 63: negative | **Figure 64:** neutral | **Figure 65:** positive (750 epochs) |

|  |  |  |
| --- | --- | --- |
| LSTM Sentiment MSE | | |
|  |  |  |
| Figure 66: negative | **Figure 67:** neutral | **Figure 68:** positive (750 epochs) |

While ARIMA outperformed LSTM at low epochs, the models implemented a grid-search technique[[61]](#footnote-61) to find optimal p,q,d parameters as input to the Augmented Dicker-Fuller test[[62]](#footnote-62). Only significant models were returned. However, LSTM models were trained single-shot with fixed attributes[[63]](#footnote-63). Finally, the LSTM variants utilized a validation\_split=0.2[[64]](#footnote-64) on the same ARIMA train data. Thus, the ARIMA models generally benefited with 20% more train data.

### Alternative Strategies based on Data

Based on results and findings from above, 2-3 additional techniques could be explored in order to better gauge ARIMA versus LSTM comparison:

1. standardize test data: ensure test dataset for both models are the same size, regardless of LSTM validation split
2. apply (partial) autocorrelation function (PACF) and grid search technique to optimize best resulting model for both ARIMA and LSTM
3. obtain larger overall dataset for both train

If the (P)ACF is implemented instead of grid search for ARIMA (2), it would drastically improve performance, since only desired/optimal configurations would be trained. Likewise, implementing a grid search for LSTM would provide an opportunity for a better MSE scores. Lastly, both (1) and (3) could provide marginal performance opportunity for LSTM. However, since LSTM is compute intensive, (2) may allow faster ARIMA model deployments, which may outweigh benefits gained for LSTM.

Overall, results indicate that one timeseries (i.e. analyst sentiment) could predict ticker volume or index price. However, sentiment analysis adds operational burden of maintaining a system of applications to perform natural language processing (NLP). It may be likely the same or greater level of benefit can be achieved by simply performing granger analysis directly on ticker, index, or volume measures. Perhaps various staggered ticker, index, or volume combinations, could reflect sentiment with substantial less noise and confounding variables. According to decay theory[[65]](#footnote-65), information becomes less valuable with time. Thus, applying macro-level optimization may be ideal to achieve fastest actionable findings.

## Plan of Action

Next steps could involve collecting a streaming dataset containing both ticker, and financial analyst tweet sentiment. A windowing operation (maybe a month or longer) could be applied, using granger analysis on ticker/analyst pairs. Three criteria could be used to gauge whether analyst sentiment is a relevant indicator for change in ticker price or volume:

1. Significant Granger Score
2. MSE sentiment timeseries score
3. MSE ticker/index timeseries score

Measuring a significance score for Granger analysis is straightforward. However, calculating MSE as a benchmark is less obvious. One idea is to dynamically define an acceptable MSE that is within a number of standard deviation(s) from a previous windowing time frame.

While the overall project was based on the question “Can Market Sentiment Predict the stock market”, perhaps sentiment measure can be (partially) reframed. As mentioned above, sentiment analysis adds operational burden, producing inherent complexity. However, staggered ticker price or index volume could be used as a measure of sentiment. Thus, volume optionally paired with ticker/index values, could attempt to predict ticker or index prices. This more direct approach eliminates NLP as an intermediate step, which would optimize findings and results.

# Follow-up Portfolio

Taking iterative lessons learned from previous project(s), a dynamic datalake has been constructed. Streaming ticker prices are analyzed using spark streaming, while simultaneously written to an intermediate dirty s3 bucket. To reduce cost, rather than streaming data directly to a target datalake, a nightly batch job compacts all daily json files into a partitioned target datalake. Simultaneously, the same nightly job performs a hive metatable[[66]](#footnote-66) update, to map new records within specified partitions. Since this follow-up project is self-funded, only a single datalake is configured consisting of roughly 200 stock ticker prices being ingested on a minute interval during the course of an open business day.

While current emphasis is largely on streaming analysis through Apache Flink[[67]](#footnote-67) on AWS Kinesis Data Analytics (KDA)[[68]](#footnote-68), data consistency is largely important for two reasons:

* Long term analysis
* Isolated historical analysis

For this reason, another nightly spark job has been created on the target datalake. When a stock split is detected for the following day, the spark job will refactor and repartition[[69]](#footnote-69) corresponding parquet files. This is particularly important, since streaming analysis uses smaller window as potential triggers, which inferences/updates time series models, each having been trained from the datalake.

As mentioned in the above FIN-664 project, general understanding of Cloud Architecture was not offered in the SU curriculum. Thus, extra time was needed to devise the follow-up datalake. Various tools such as AWS CloudFormation[[70]](#footnote-70) and AWS CodePipeline[[71]](#footnote-71) have automated the entire infrastructure and deployment of analysis applications. However, the most interesting components thus far, is likely the Apache Flink application dedicated to perform candlestick analysis[[72]](#footnote-72). While the exact streaming codebase is private, an example demo codebase[[73]](#footnote-73) has been publicly released. Lessons learned here, may be a precursor for other streaming applications yet to be developed. However, streaming analysis will be used as a feedback loop and inference mechanism for downstream models.

1. 1 http://coursecatalog.syr.edu/content.php?catoid=14&navoid=1866 [↑](#footnote-ref-1)
2. https://rstudio.github.io/shinydashboard/ [↑](#footnote-ref-2)
3. https://github.com/jeff1evesque/fin-654#exploratory [↑](#footnote-ref-3)
4. https://github.com/jeff1evesque/fin-654#general-pareto-distribution [↑](#footnote-ref-4)
5. https://github.com/jeff1evesque/fin-654/blob/master/app.R#L472-L518 [↑](#footnote-ref-5)
6. https://github.com/jeff1evesque/fin-654#efficient-frontier [↑](#footnote-ref-6)
7. [↑](#footnote-ref-7)
8. https://github.com/jeff1evesque/fin-654/tree/2a0c2ed070db4f3bcd17efa6219658388519187f/data/security [↑](#footnote-ref-8)
9. https://github.com/jeff1evesque/fin-654/tree/2a0c2ed070db4f3bcd17efa6219658388519187f/data/stock-exchange [↑](#footnote-ref-9)
10. https://github.com/jeff1evesque/fin-654/blob/master/packages/fin654/R/load\_symbol.R#L19-L43 [↑](#footnote-ref-10)
11. https://github.com/jeff1evesque/fin-654/tree/2a0c2ed070db4f3bcd17efa6219658388519187f/data/symbol [↑](#footnote-ref-11)
12. https://github.com/jeff1evesque/fin-654/blob/master/app.R#L691-L724 [↑](#footnote-ref-12)
13. https://github.com/jeff1evesque/fin-654/blob/master/app.R#L661-L689 [↑](#footnote-ref-13)
14. https://github.com/jeff1evesque/fin-654/blob/master/packages/fin654/R/gpd.R [↑](#footnote-ref-14)
15. https://github.com/jeff1evesque/fin-654/blob/master/app.R#L472-L486 [↑](#footnote-ref-15)
16. https://github.com/jeff1evesque/fin-654-hw/blob/master/hw4/template.Rmd#L230 [↑](#footnote-ref-16)
17. https://github.com/jeff1evesque/fin-654/blob/master/packages/fin654/R/markowitz.R [↑](#footnote-ref-17)
18. https://github.com/jeff1evesque/fin-654-hw/blob/master/hw4/template.Rmd#L951 [↑](#footnote-ref-18)
19. https://github.com/jeff1evesque/fin-654-hw/blob/master/hw4/template.Rmd#L230-L232 [↑](#footnote-ref-19)
20. https://spark.apache.org/docs/latest/streaming-programming-guide.html [↑](#footnote-ref-20)
21. https://www.princeton.edu/~markus/teaching/Fin501/07Lecture.pdf [↑](#footnote-ref-21)
22. https://towardsdatascience.com/efficient-frontier-in-python-detailed-tutorial-84a304f03e79 [↑](#footnote-ref-22)
23. https://nbviewer.org/github/rian-dolphin/Efficient-Frontier-Python/blob/main/Markowitz.ipynb [↑](#footnote-ref-23)
24. https://towardsdatascience.com/spark-3-2-session-windowing-feature-for-streaming-data-e404d92e267 [↑](#footnote-ref-24)
25. https://answers.syr.edu/display/ischool/IST+664+-+Natural+Language+Processing [↑](#footnote-ref-25)
26. https://github.com/jeff1evesque/ist-664/tree/master/QuestionAnswerCMU/data [↑](#footnote-ref-26)
27. https://github.com/jeff1evesque/ist-664/tree/master/Reddit/data [↑](#footnote-ref-27)
28. https://github.com/jeff1evesque/ist-664/tree/master/StackOverflow/data [↑](#footnote-ref-28)
29. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/insert.py [↑](#footnote-ref-29)
30. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/select.py [↑](#footnote-ref-30)
31. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/train.py#L23-L157 [↑](#footnote-ref-31)
32. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/select.py#L42-L50 [↑](#footnote-ref-32)
33. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/select.py#L97-L102 [↑](#footnote-ref-33)
34. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/select.py#L90 [↑](#footnote-ref-34)
35. https://github.com/jeff1evesque/ist-664/blob/master/Reddit/app/train.py [↑](#footnote-ref-35)
36. https://github.com/jeff1evesque/ist-664/blob/master/StackOverflow/StackOverflow\_Classification.ipynb#POS-tagging [↑](#footnote-ref-36)
37. https://github.com/jeff1evesque/ist-664/blob/master/StackOverflow/StackOverflow\_SKLearn.ipynb#Define-the-function-for-tokenization [↑](#footnote-ref-37)
38. https://github.com/jeff1evesque/ist-664/blob/master/Wilson\_Levesque\_Final\_Project2.docx [↑](#footnote-ref-38)
39. https://docs.aws.amazon.com/vpc/latest/userguide/VPC\_Route\_Tables.html#subnet-route-tables [↑](#footnote-ref-39)
40. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Instances.html [↑](#footnote-ref-40)
41. https://github.com/daniel-kukiela/nmt-chatbot [↑](#footnote-ref-41)
42. https://answers.syr.edu/display/ischool/IST+736+-+Text+Mining [↑](#footnote-ref-42)
43. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/topic\_model.py [↑](#footnote-ref-43)
44. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/penn\_treebank.py [↑](#footnote-ref-44)
45. https://github.com/jeff1evesque/ist-736/blob/master/brain/exploratory/word\_cloud.py [↑](#footnote-ref-45)
46. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/text\_classifier.py [↑](#footnote-ref-46)
47. https://github.com/jeff1evesque/ist-736/blob/master/brain/exploratory/sentiment.py [↑](#footnote-ref-47)
48. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/granger.py [↑](#footnote-ref-48)
49. https://github.com/jeff1evesque/ist-736/tree/master/consumer/twitter\_api [↑](#footnote-ref-49)
50. https://github.com/jeff1evesque/ist-736/blob/master/consumer/quandl\_api/quandl\_query.py [↑](#footnote-ref-50)
51. https://github.com/jeff1evesque/ist-736/tree/master/data/twitter [↑](#footnote-ref-51)
52. https://github.com/jeff1evesque/ist-736/tree/master/data/quandl [↑](#footnote-ref-52)
53. https://docs.aws.amazon.com/streams/latest/dev/introduction.html [↑](#footnote-ref-53)
54. https://aws.amazon.com/big-data/datalakes-and-analytics/datalakes/ [↑](#footnote-ref-54)
55. https://github.com/jeff1evesque/ist-736/tree/master/reports [↑](#footnote-ref-55)
56. https://github.com/jeff1evesque/ist-736/tree/master/viz [↑](#footnote-ref-56)
57. https://github.com/jeff1evesque/ist-736/tree/master/viz/analysis/chris--cboe\_vx1/LizAnnSonders [↑](#footnote-ref-57)
58. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html> [↑](#footnote-ref-58)
59. <https://github.com/jeff1evesque/ist-736/tree/master/viz/classification/chris--cboe_vx1/LizAnnSonders> [↑](#footnote-ref-59)
60. <https://www.cnbc.com/2018/09/07/trump-says-tariffs-on-another-267-billion-in-china-goods-ready-to-go.html> [↑](#footnote-ref-60)
61. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/arima.py#L236-L297 [↑](#footnote-ref-61)
62. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/arima.py#L374 [↑](#footnote-ref-62)
63. https://github.com/jeff1evesque/ist-736/blob/master/brain/model/timeseries.py#L114-L123 [↑](#footnote-ref-63)
64. https://github.com/jeff1evesque/ist-736/blob/master/brain/algorithm/lstm.py#L355 [↑](#footnote-ref-64)
65. https://en.wikipedia.org/wiki/Decay\_theory [↑](#footnote-ref-65)
66. https://docs.aws.amazon.com/glue/latest/dg/tables-described.html [↑](#footnote-ref-66)
67. https://docs.aws.amazon.com/kinesisanalytics/latest/java/gs-python.html [↑](#footnote-ref-67)
68. https://aws.amazon.com/kinesis/data-analytics/ [↑](#footnote-ref-68)
69. https://stackoverflow.com/a/56301268/2063478 [↑](#footnote-ref-69)
70. https://aws.amazon.com/cloudformation/ [↑](#footnote-ref-70)
71. https://aws.amazon.com/getting-started/hands-on/set-up-ci-cd-pipeline/ [↑](#footnote-ref-71)
72. https://www.investopedia.com/trading/candlestick-charting-what-is-it/ [↑](#footnote-ref-72)
73. https://github.com/jeff1evesque/kinesis-analytics-demo [↑](#footnote-ref-73)