A drawing of a cartoon character

Description automatically generated

Jeff Levesque

https://github.com/jeff1evesque/ist-exit-portfolio | Final Project

Data Science: Portfolio

2023

Table of Contents

[Introduction 2](#_Toc127905004)

[Capstone Projects 2](#_Toc127905005)

[Portfolio Analysis (FIN-654) 3](#_Toc127905006)

[Chatbot (IST-664) 4](#_Toc127905007)

[Stock Market Sentiment Analysis (IST-736) 6](#_Toc127905008)

[Conclusion & Follow-up Interests 8](#_Toc127905009)

# Introduction

This paper provides a summary of requirements met for the MS in Applied Data Science. A collection of three projects was chosen to help portray the learning achievements made during the IST program at Syracuse University. Various techniques ranging from data mining, exploratory data analysis (EDA), natural language processing (NLP), as well as predictive and financial modeling were performed. This often entailed using languages such as python, R, hadoop, as well as basic systems and cloud engineering. Since each IST course was roughly 10 weeks long, performing a meaningful project was sometimes challenging. Nonetheless, each course and respective project provided valuable experience in making actionable insight from collected data.

# Capstone Projects

|  |  |  |
| --- | --- | --- |
| Course | Capstone Project | Skills |
| FIN-654  Financial Analytics | **Portfolio Analysis**  Chosen stock tickers were analyzed to determine optimal portfolio allocation, using financial and time series modeling | Python, R, time series analysis, financial analysis, Shiny Dashboard |
| IST-664  Natural Language Processing | **Chatbot**  EDA was initially performed to study the data distribution. A trained classifier was ensembled with an LSTM/NMT model to producer the overall chatbot experience | Python, MongoDB / Hadoop, Jupyter Notebook, Time series analysis, Classification analysis, Natural language processing |
| IST-736  Text Mining | **Stock Market Sentiment Analysis**  An attempt was made to determine whether sentiment from financial analysts can predict the stock market. Topic modeling was performed to determine most relevant stock tickers. Sentiment analysis was performed on the same financial analyst tweets and were coupled with corresponding stock ticker price using granger analysis to determine whether sentiment could predict stock price | Topic modeling, Sentiment analysis, Time series analysis, Classification analysis, Signal analysis, Data mining, AWS, Jupyter Notebook |

## Portfolio Analysis (FIN-654)

Assignments: <https://github.com/jeff1evesque/fin-654-hw>

Project: <https://github.com/jeff1evesque/fin-654>

The course for this project focused on teaching methods and tools for decision making in the financial industry. We had a total of four assignments (roughly biweekly), performed as a group and turned in for grading. Each assignment generally entailed a skeleton R code in a word document, sometimes requiring finesse, and often accompanied by questions requiring financial interpretation. What was interesting about this course, was that the final project was an encapsulation of these four group assignments. Individually, we were allowed to recycle any components obtained/learned from these course assignments to satisfy a final project topic approved by the professor.

Since the course was 10 weeks long, it’s almost over before getting the grasp of some of the course materials. Earlier assignments taught us risk management as a function of supply, volatility, as well as interpretation of statistical measures including data moments, heteroscedasticity, autocorrelation, standard deviation, and kurtosis. Concepts were visualized using ggplot in R. During the middle of the course (roughly week 5/6), I started looking for potential data sources related to publicly traded companies portraying risk. I found a dataset on the “World’s Biggest Data Breaches & Hacks”[[1]](#footnote-1), and considered it as a candidate data source. I wanted to study ways to minimize risk of breaches either before occurring, or minimizing the blast effect after as function of modern portfolio analysis. As weeks 7-9 unfolded, we learned more sophisticated R tools, and financial theories. Assignments[[2]](#footnote-2) were visualized using flexdashboard[[3]](#footnote-3) or shinydashboard[[4]](#footnote-4) (instead of snippets of R in a word document). New risk measures were introduced, not limited to:

* Value At Risk
* Expected Shortfall
* Efficient Frontier
* Markowitz Model

We were able to take financial scenarios for the last two assignments, and provide business remarks regarding asset allocation, as well as recommend the distribution of supply goods to purchase in order to reduce risk at a given price. Assignment 4[[5]](#footnote-5) provided a scenario of “A freight forwarder with a fleet of bulk carriers want to optimize their portfolio of metals markets with entry into the nickel business and use tramp trade. They have allocated $250 million to purchase metals”. Using supplied information, we were able to make recommendation regarding how the $250 million should be dispersed into purchasing nickel, copper, and aluminum. By roughly week 8, we had about 2 weeks to potentially integrate concepts learned from the assignment 4 into a final project. At this point I had to iron out the ambiguous data source into a viable final project topic. I decided to select all companies that was breached, if the company name was able to inner join against a list of ticker symbol dataset using the quandl API. I had originally wanted to construct a more meaningful project topic, due to time constraint reduced the problem statement “how can I reduce the risk of a portfolio consisting of companies recently breached?”.

Since my background was much stronger in Python instead of R, at roughly week 7 (before the topic solidified), I started coding the data mining/integration into the R shinydashboard using reticulate[[6]](#footnote-6). This allowed me to perform numpy/panda operations[[7]](#footnote-7) on imported data. Once the data was in the right format, I was able to ship it back to R for visualization using ggplot2 (or equivalent). Specifically, I saved the earlier “World’s Biggest Data Breaches & Hacks”, along with a similar dataset from “Privacy Rights Clearinghouse”[[8]](#footnote-8) locally. These two datasets were merged into a dataframe in python, then inner joined against a list of stock ticker symbols obtained from the quandl API[[9]](#footnote-9).

From the shinydashboard, I was able to visualize through a series of barcharts, which tickers were riskier individually, by reviewing the variance of ticker volume. Since I had massaged/formatted timeseries data in a dataframe format, I decided to recycle code from other courses, in order to perform time series analysis on the overall portfolio. Specifically, I performed a side-by-side comparison between LSTM vs ARIMA ability to predict the portfolio price. Similar Efficient Frontier and General Pareto Distribution code from previous assignment(s) was recycled and visualized within the same shinydashboard. Ultimately, I was able to produce a shinydashboard similar to assignment 4, showcasing some exploratory analysis, more complicated timeseries analysis (ARIMA and LSTM), as well as financial analysis concepts learned during the course.

## Chatbot (IST-664)

Assignments: <https://github.com/jeff1evesque/ist-664-hw>

Project: <https://github.com/jeff1evesque/ist-664>

The IST-664 course was focused on Natural Language Processing. While the initial goal for the final project was an attempt to produce a chatbot, simpler objectives were made to investigate the data distribution as well as perform classification modeling. The smaller objectives satisfied the minimum requirements for the project, while the chatbot was a larger endeavor for the final few weeks of the 10-week course. In general, three different datasets were used to generate respective models:

* QuestionAnswerCMU[[10]](#footnote-10): question-answer pair to train question classifier
* Reddit[[11]](#footnote-11): json data used to train RNN based NMT chat agent
* StackOverflow[[12]](#footnote-12): list of postings grouped by channels used to train classifier to label questions

The StackOverflow dataset was also chosen to satisfy the minimum requirement objective. It was noted in the associated report[[13]](#footnote-13), the StackOverflow dataset originally consisted of 75 GB of data, which was reduced to 6 GB of data after the data was filtered to only question and verified answer pairs. To further reduce the dataset, only the top 10 most popular channels from the total 173 were selected. These channels include: Math, Russian, LaTex, Super User, Server Fault, Code Review, Ask Ubuntu, Portuguese, Unix, and Physics. By using these channels, the dataset was again reduced to just over 3 GB.

Next, the data was loaded from json format into pandas. Surprisingly, only 9 of the 1.3 million records would not load due to special escape character conditions. However, the dataset still required roughly 102 GB of memory to store the listing of features. To reduce the scope of the data, a sample of 20% of the data was taken without replacement. This reduced dataset was used to train a Naïve Bayes (NB) classifier from the NLTK package using an 80/20% split. A five-fold cross validation was initially used. Initial results were acceptable, returning 74% precision, 76% recall, and 75% F1 score[[14]](#footnote-14). In general, the classifier was good at identifying StackOverflow questions with Math, Russian, Portugese, and LaTex channels, and suffered making distinction between Physics, Math, and other computer technology related channels. This is not surprising, since the Russian and Portuguese characters generally appeared significantly less in the other channels. While the NB classifier performed well, a Random Forrest (RF) was trained with the same data. The cross-validation result indicates 81.2% mean accuracy.

As a comparison, a unigram word frequency model was performed for each NB and RF using sklearn pipeline. However, both results returned a lower accuracy of roughly 0.01%. Since unigram word frequency appeared to be a nonfactor, a bigram variant was attempted. Results in this effort show a more significant drop of mean accuracy at 73.5%.

Since the base study was capable of training a model identifying a channel for a given question, we were able to proceed with the remaining two models:

* QuestionAnswerCMU: question-answer pair to train question classifier
* Reddit: json data used to train RNN based NMT chat agent

Similar approaches to the StackOverlow classifier were made on the QuestionAnswerCMU dataset. The train was performed on question-answer pairs[[15]](#footnote-15), and produced a RF model having 72.31% accuracy[[16]](#footnote-16).

Since both classifiers were fairly accurate, we proceeded with the last segment of training an RNN based NMT chat agent. Roughly 6 months of Reddit data was downloaded, containing various posts and responses. MongoDB was utilized for the map-reduce functionality, with regex to clean unnecessary character patterns. Aside from data cleansing, the map-reduce checks whether a post contains a parent\_id, then recursively attempts to match an id from another post. If a match is found, the body field is appended to the appropriate comments or posts array.

Finally, to train the chatbot neural network, the resulting map-reduce data structure needed to output the corresponding comments, and posts to be individual files defined by NMT[[17]](#footnote-17). It is important to note, while 6 months of Reddit data was collected, only a small portion of the first month was utilized. This was largely compute constraint (as a function of cost), as well as fast approaching course deadline. 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. Further, if a low-quality response is returned, then a default classifier is re-engaged. 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.

## Stock Market Sentiment Analysis (IST-736)

Assignments: <https://github.com/jeff1evesque/ist-736-hw>

Project: <https://github.com/jeff1evesque/ist-736>

This final project was my favorite of the three. In the first project (FIN-654), my interests were a bit vague initially, and focused more on data engineering by interpolating python in R before forming a conclusive topic. In IST-736, I had defined a more concrete topic earlier in the course, and had the benefit of being at the tail end of the overall Data Science program. This largely meant that I could reuse various python code I had developed from previous courses (i.e. FIN-654).

The project ultimately attempted to answer – Can Market Sentiment Predict the stock market? While the topic seems a bit predictable, my personal goal was to fuse previous course materials, and complete a project more that was more than just text mining or NLP. To address the overall topic, the following techniques were applied:

* Exploratory Data Analysis (EDA): twitter topic modeling to determine which stock ticker to study
* Sentiment Analysis: normalized sentiment scores for each financial analyst tweets
* Granger Analysis: find significant sentiment and ticker pair combinations
* Timeseries Analysis: train LSTM and ARIMA models on tickers filtered from Granger Analysis

While one main focus of the study was between timeseries models determined by LDA topic modeling and 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 volume/price momentum:

* Signal Analysis: determine whether ticker volume/price exceed standard deviation cutoff
* Classification Analysis: TF-IDF text corpus (X) trained against signal result (y)

More generally, stock prices exceeding the upper threshold was binned a value 1 if it also was greater than the previous time step. Conversely, points below the lower threshold and less than the previous time step was binned a value -1. When the first two conditions are not satisfied, a value of 0 is assigned. While this project was fairly large, and a good representation of the overall IST program, I was quite excited about this small signal analysis component. Specifically, I had created a simple system of equations, then interpreted it into python code. The icing on the cake for this section was that, it could attempt to address a corollary topic – Can Tweets from financial analyst predict stock price momentum? Further, the actual generated data for the signal analysis was predicated on earlier granger analysis, which addresses whether sentiment from financial analyst could be said to granger cause stock price. So, there was some (maybe tangential) relationship back to the primary topic question.

In general, numerous types of classifiers with different vectorizing techniques including parts of speech (POS) tagging were implemented. The k-fold cross validation was taken for each variation (i.e. classifiers) of financial analyst tweets against binned signals. Some of our better results at 80/20 split was found using the Bernoulli Naïve Bayes (BNB). Specifically, tweets from JimCramer could predict AMZN ticker volume with roughly 76% accuracy, while SJosphBurns and ReformedBroker were over 90% accurate. Corresponding precision, recall, and fscore were over 75% for JimBroker, and roughly 95% for both SJosphBurns and ReformedBroker.

Finally, timeseries analysis was performed (very similar to FIN-654) on both ticker price and financial analyst normalized sentiment scores. However, the original codebase received a major upgrade for ARIMA. Specifically, each p,q,d were allowed to vary within a specified range. This allowed a grid-search approach for the ARIMA models, where results were retained if scores were significant using the Dicker-Fuller test. This largely put the LSTM at a disadvantage, even more so since 20% of the data was set aside for validation. In the case of AMZN ticker price, while ARIMA had 20% more data points, it was interesting to see that the mean square error (MSE) outperformed LSTM by a factor of 10.

In general, a fair comparison could have been engineered between the two. However, the compute time as well as cost for LSTM would have greatly exceeded my compute budget. A small benchmark was constructed for the project. Again, since the course was 10 weeks long, each benchmark generally captured the total time for a particular segment to complete (i.e. overall ticker time series, overall sentiment time series, overall classification, etc), rather than individual models:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: Compute Performance (min) | | | | | | | |
| Resources | **TS Stock\*\*** | | **TS Sent\*\*** | | **Class Sent\*\*** | **Granger\*\*** | |
| 3.0GHz (2 core), 8GB RAM | ~13 | (p,q,d) = autoscale  epochs = 750  cells = 4  units = 50  dropout = 0 | 42 | (p,q,d) = range(0,4)  epochs=750  cells = 4  units = 50  dropout = 0 | ~4 | ~3 | range(0,4) |
| p2.xlarge: 1GPU, 4vCPU 61GB RAM | ~20 | (p,q,d) = autoscale  epochs = 3,000  cells = 4  units = 50  dropout = 0 | 43 | (p,q,d) = range(0,4)  epochs = 3,000  cells = 4  units = 50  dropout = 0 | ~2 | ~1 | range(0,4) |
| p2.xlarge: 1GPU, 4vCPU 61GB RAM | ~13 | (p,q,d) = autoscale  epochs = 1,500  cells = 4  units = 50  dropout = 0.2 | 29 | (p,q,d) = range(0,4)  epochs = 1,500  cells = 4  units = 50  dropout = 0.2 | N/A | N/A | range(0,4) |
| p2.8xlarge: 8GPU, 32 vCPU, 488 GB RAM | N/A | (p,q,d) = autoscale  epochs = 5,000  cells = 4  units = 512  dropout = 0.4 | N/A | (p,q,d) = range(0,4)  epochs = 5,000  cells = 4  units = 512  dropout = 0.4 | N/A | N/A | range(0,4) |

**Note:** p2 architecture was implemented based on 2019 resource attributes[[18]](#footnote-18).

# Conclusion & Follow-up Interests

As discussed above, IST-664 involved an attempt to create a chatbot through numerous ensembled machine learning (ML) modeling on a personal laptop. While some parts were functional, the overall endeavor was greatly restricted by compute resources. On comparison, recent release of ChatGPT reportedly costs $100,000 a day to run[[19]](#footnote-19). Desired outcomes in “Data Science” are often at mercy of available data and compute resources. What I have learned generalized from the IST-664 chatbot vs. ChatGPT, my interests towards Cloud Native technologies supporting Big Data and streaming analysis have grown immensely. As retrospect, it’s interesting to see the evolution of “Data Science”. While practitioners perform varying degree of EDA to help contextualize problems for humans, we’re in the age where AI/ML frameworks are just beginning to dynamically solve problems orders of magnitude more sophisticated.

For these reasons, I have taken lessons learned from this program, and have expanded into the development of a platform (jefflevesque.com) to aggregate streaming data to facilitate generating and sharing ML models. For example, some parts of IST-736 have become a small part of the overall effort – just one stream, and one datalake of many that people can access. The platform has been ingesting roughly 200 stock ticker price every minute during the business day, eventually consumed into a parquet partitioned datalake. To expand on FIN-654, candlestick analysis has been devised as an Apache Flink application on the same ingest stream. While the exact streaming codebase is private, an example demo codebase[[20]](#footnote-20) has been publicly released. Future plans may include integration of FIN-654 concepts including the Efficient Frontier as well as the Markowitz model. However, a greater desire of adding additional data streams or developing neural networks may take precedence. In IST-736, we had some experience using p2.8xlarge instances for timeseries analysis. Thus, these concepts may provide some initial insight into the transition into tools such as Amazon SageMaker.

Various fields within “Data Science” often try to visualize data to help contextualize a problem set. It will be interesting to see whether simple data science questions become less prevalent with time. In FIN-654, a staple component of the course was R with Shiny dashboard. However, it is not an unimaginable future, a ChatGPT equivalent can expose an API over the internet to directly answer the actual desired problem set. The IST program at Syracuse has afforded me foundational experience in applied Data Science, with a sharper sense of direction. T.S. Eliot once said ‘The journey not the arrival matters’. While numerous learning objectives have been met in this program, my journey as practitioner has only just begun.

1. https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/ [↑](#footnote-ref-1)
2. https://github.com/jeff1evesque/fin-654-hw/tree/master/hw3 [↑](#footnote-ref-2)
3. https://posit.co/blog/flexdashboard-easy-interactive-dashboards-for-r/rd [↑](#footnote-ref-3)
4. https://rstudio.github.io/shinydashboard/ [↑](#footnote-ref-4)
5. https://github.com/jeff1evesque/fin-654-hw/tree/master/hw4 [↑](#footnote-ref-5)
6. https://rstudio.github.io/reticulate/ [↑](#footnote-ref-6)
7. https://github.com/jeff1evesque/fin-654/blob/master/python/dataframe.py [↑](#footnote-ref-7)
8. https://privacyrights.org/data-breaches [↑](#footnote-ref-8)
9. https://www.quandl.com/ [↑](#footnote-ref-9)
10. https://github.com/jeff1evesque/ist-664/tree/master/QuestionAnswerCMU/data [↑](#footnote-ref-10)
11. https://github.com/jeff1evesque/ist-664/tree/master/Reddit/data [↑](#footnote-ref-11)
12. https://github.com/jeff1evesque/ist-664/tree/master/StackOverflow/data [↑](#footnote-ref-12)
13. https://github.com/jeff1evesque/ist-664/blob/master/Wilson\_Levesque\_Final\_Project.docx [↑](#footnote-ref-13)
14. https://github.com/jeff1evesque/ist-664/blob/master/StackOverflow/StackOverflow\_Classification.ipynb [↑](#footnote-ref-14)
15. https://github.com/jeff1evesque/ist-664/tree/master/QuestionAnswerCMU [↑](#footnote-ref-15)
16. https://github.com/jeff1evesque/ist-664/blob/master/QuestionAnswerCMU/QuestionAnswerCMU\_Classification.ipynb [↑](#footnote-ref-16)
17. https://github.com/daniel-kukiela/nmt-chatbot/blob/59551eed4f868ba2030f933f92842177720ca121/README.md#setup [↑](#footnote-ref-17)
18. <https://aws.amazon.com/ec2/instance-types/p2/> [↑](#footnote-ref-18)
19. https://www.ciocoverage.com/openais-chatgpt-reportedly-costs-100000-a-day-to-run/ [↑](#footnote-ref-19)
20. https://github.com/jeff1evesque/kinesis-analytics-demo [↑](#footnote-ref-20)