Tristan McCarty
CSIS638

A project I've routinely considered and would be interested in trying to do would be creating a Machine Learning model that attempts to predict stock prices. The project offers a lot of interesting methods to train the algorithm, including learning from previous trends within a stock's history or taking in news articles about finance and other topics. This allowed the project to scale depending as I encountered issues with a minimum model based on historical data, and if there are few issues the dataset can grow to include news data and or to use Apache Kafka to stream additional market trends to be used for additional predictions. There are also many papers on this topic, some produced using a toolkit I'm familiar with, SciKit Learn, which I was able to combine with technologies related to what we covered in class, in this case Apache Spark and Pandas.

From an investing standpoint, the project can be considered important as a way to see how the feature set of inputs play into the ability of increased success in predictions, possibly comparing the incorporation of certain news outlets and how that might relate to investors who subscribe to those outlets. How well will the model perform with just historic data, vice the addition of more inputs, such as news or other financial data? There are also suggestions of the limitation of Machine Learning models, in that if these models were accurate enough to produce consistent profit off of the stock market, there would be a variety of funds advertising their models and how they can outpace catch-all index funds. The papers summarized in the latter half of this report support this as well, as none had greater than 60% accuracy in their models and mine struggled to supply interesting output.

Looking at the project from a learning and database standpoint, It was also an important project for me as a way to become more familiar with tools involved with data analytics, and the involved processes to set up the tools and use them. I currently work in a job that is more data-focused, and would like to catch up to my peers in their knowledge about the field. Due to this project, I was able to approach Scikit Learn again, make use of Pandas and Apache Kafka, with more of a foundation than I had prior to the class.

As stated above, the topic garners a lot of interest, and as such it is easy to find a variety of different research papers describing their methods, the data incorporated, or the systems and models they attempted to use. "Empirical analysis: stock market prediction via extreme learning Machine" uses, as suggested by the title, "extreme" learning practices to try and predict intramarket trends, or trades during the day of trading. The authors also considered both financial news and trends as the same feature set, rather than inputting the data into two separate models. The paper tested different algorithms on resource usage, time taken, and prediction accuracy, and the highest accuracy that was back-tested on historical data seemed to be about 60% accurate.

"Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFinText System" caught my eye due to the title suggesting the increased usage of financial news as part of the model's feature set. Interestingly enough, they used a "simulated trading engine" as well to test the performance of the model, and found their model to be 60% accurate and the trading engine had a 2% return. They describe more thoroughly their processing of the textual data found in their news sources, which could be useful for considerations when if I were to incorporate additional features into the model.

"Stock Price Prediction using Reinforcement Learning" proposes an alternate approach, suggesting the use of Reinforcement Learning, considering that the stock market provides a unique set of problems for Machine Learning in that the rewards are not always instantaneous.

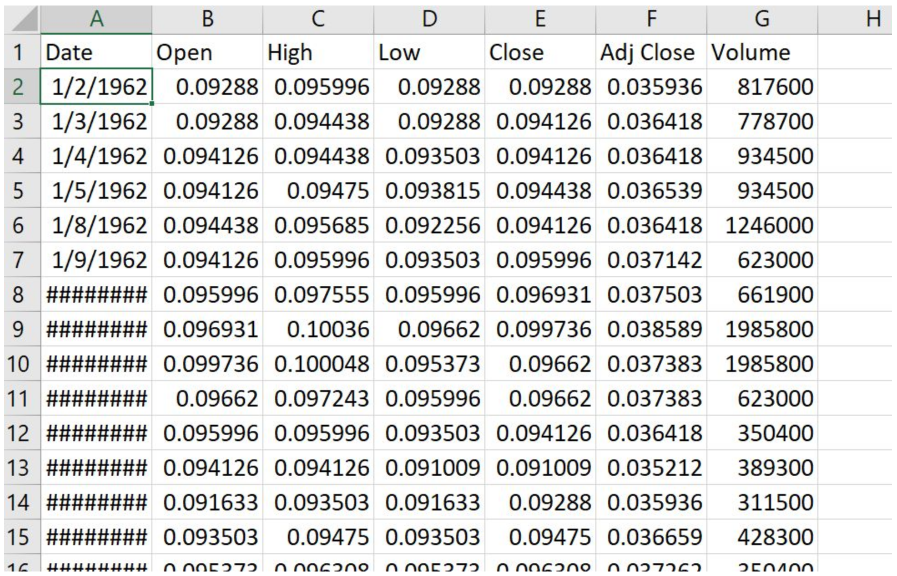
In this paper, the author Jae Won Lee uses a Neural Network with varied days of returns.The model takes in data from each stock daily, including data on each stock two years prior, and was then tested on data on stocks for the following year. After finalizing the project, this approach is one I would have liked to try.

The first week of June was projected to be acquiring a dataset, by collecting data and setting up the infrastructure in which the data would be stored. A popular recommendation for doing so was extracting data from Yahoo! Finance's API, but according to users on the linked Stack Overflow question that Finance service was discontinued in 2017. There are many paid services, some asking for several thousand dollars for terabytes of data that cover only a single year, implying that this will likely be the most challenging piece. At worst, I could manually collect the data for one or more symbols from Yahoo! Finance charts.

The second week of June was estimated to be developing the model based off of the data, including preliminary data analytics to select possible features for the feature set and organizing the data hopefully without bias. I near instantly discovered Yahoo! Finance's historical data option, and so I narrowed the scope to a single symbol to reduce the amount of data collected which allowed me to focus more on Apache Kafka.

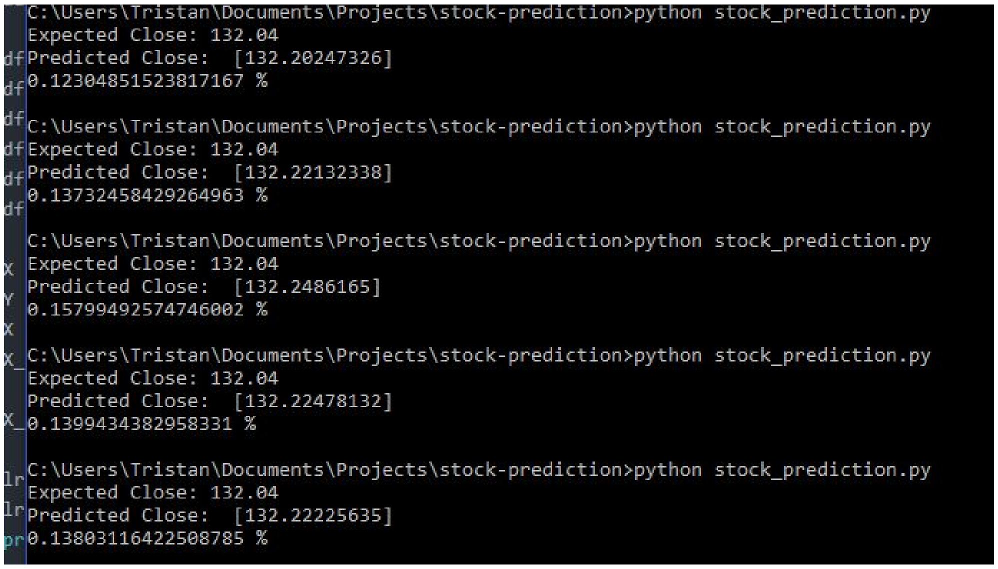
The final weeks of June were meant to be overflow from the previous week, to finish the basic model that takes data from the collected historical data. If time allowed, I would have liked to find additional features to add to the feature set, primarily financial and geopolitical news and pull from those pages to attempt to enhance predictions. This flexible third and fourth week allowed some room to include integration with Apache Kafka for streaming data and predictions instead and the issues that arose with webscraping.

The first week of June was the kickstart of the project, starting as anticipated. Initial research suggested finding stock data would be difficult, however Yahoo! Finance still has options for downloading historical data for individual symbols. This data contained opening and closing, and then adjusted values based on the amount of stocks available in present day. This data was excellent, was easily obtained, and helped frame the future aspects of the project. With at most an hour spent on finding the data for a symbol and looking over the data obtained, this left plenty of time to work on building a basic model.



	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Adj Close	Volume	
2	1/2/1962	0.09288	0.095996	0.09288	0.09288	0.035936	817600	
3	1/3/1962	0.09288	0.094438	0.09288	0.094126	0.036418	778700	
4	1/4/1962	0.094126	0.094438	0.093503	0.094126	0.036418	934500	
5	1/5/1962	0.094126	0.09475	0.093815	0.094438	0.036539	934500	
6	1/8/1962	0.094438	0.095685	0.092256	0.094126	0.036418	1246000	
7	1/9/1962	0.094126	0.095996	0.093503	0.095996	0.037142	623000	
8	########	0.095996	0.097555	0.095996	0.096931	0.037503	661900	
9	########	0.096931	0.10036	0.09662	0.099736	0.038589	1985800	
10	########	0.099736	0.100048	0.095373	0.09662	0.037383	1985800	
11	########	0.09662	0.097243	0.095996	0.09662	0.037383	623000	
12	########	0.095996	0.095996	0.093503	0.094126	0.036418	350400	
13	########	0.094126	0.094126	0.091009	0.091009	0.035212	389300	
14	########	0.091633	0.093503	0.091633	0.09288	0.035936	311500	
15	########	0.093503	0.09475	0.093503	0.09475	0.036659	428300	
16	########	0.095373	0.096308	0.095373	0.096308	0.037262	350400	

The initial data is of Disney's stock, and I used Yahoo!'s CSV file as input into the Python library Panda as a dataframe. Then I formatted the dataframe into the feature set and the predicted outcome based on Towards Data Science's articles on using SciKit Learn for Stock Prediction. As mentioned in the initial research, predicting finances with Machine Learning and data is a popular topic, such that there were a wide array of tutorials available for jump starting the project. Lucas Kohort's "Predicting Stock Prices with Python" and Vincent Tatan's "In 12 minutes: Stocks Analysis with Pandas and Scikit-Learn" gave me an excellent skeleton. This put me ahead of schedule, as I had the data formatted and a basic model which could now be integrated with Apache Kafka for an initial minimum viable product. I had a planned vacation from June 4th to June 9th that I expected would be intertwined with work on this project, but with the early progress made, I was able to focus on the vacation and studying for the midterm and push the finalizing of this minimum project to the following weekend.



```
C:\Users\Tristan\Documents\Projects\stock-prediction>python stock_prediction.py
Expected Close: 132.04
Predicted Close:  [132.20247326]
0.12304851523817167 %

C:\Users\Tristan\Documents\Projects\stock-prediction>python stock_prediction.py
Expected Close: 132.04
Predicted Close:  [132.22132338]
0.13732458429264963 %

C:\Users\Tristan\Documents\Projects\stock-prediction>python stock_prediction.py
Expected Close: 132.04
Predicted Close:  [132.2486165]
0.15799492574746002 %

C:\Users\Tristan\Documents\Projects\stock-prediction>python stock_prediction.py
Expected Close: 132.04
Predicted Close:  [132.22478132]
0.1399434382958331 %

C:\Users\Tristan\Documents\Projects\stock-prediction>python stock_prediction.py
Expected Close: 132.04
Predicted Close:  [132.22225635]
0.13803116422508785 %
```

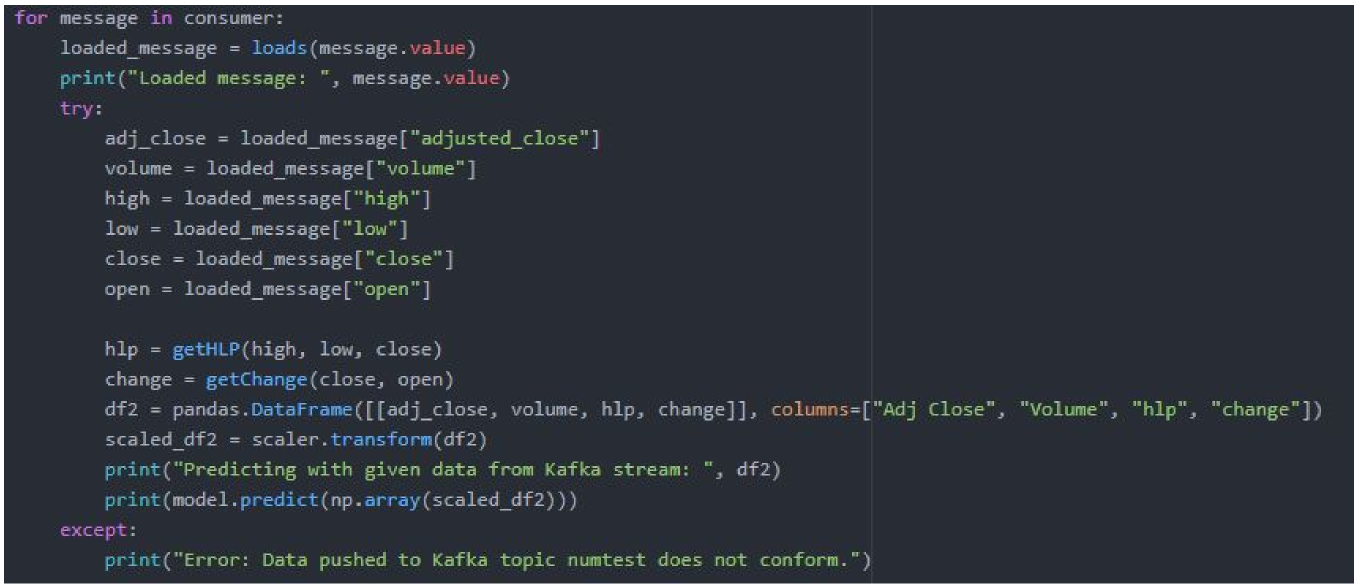
That is where the project currently stood as of the intermediate report, with the above image shows it executing with a single prediction of the most recent closing. I anticipated integration with Apache Kafka and more data processing the weekend before the last class, and then tweaking the model to output a prediction of investment favorability rather than the likely more difficult prediction of the future stock price. As I began work on the integration with Kafka, I met some of the first major roadblocks for the project.

Apache Kafka is not built towards Windows, so it required specialized installation on my machine, and used up multiple days of development. Fortunately the Kafka API for Python was much simpler, and I stood up a topic with a producer and consumer easily after both Zookeeper and Kafka were set up. The next major roadblock was webscraping for finance values for the producer to provide, as there was a lack of free finance APIs and both Google and Yahoo! Finance have shut theirs down. After significant trial and error, the producer now can scrape using some hardcoded values.

The end project is two Python scripts, stock_prediction.py and kafka_host.py. The former is where most of the initial work was done, and there was plenty of documentation on taking financial CSV files and using them as a model. The latter required significantly more effort, as Kafka was difficult to install and then to provide an effective producer, I needed to develop a script that could scrape a website for the correct financial data.

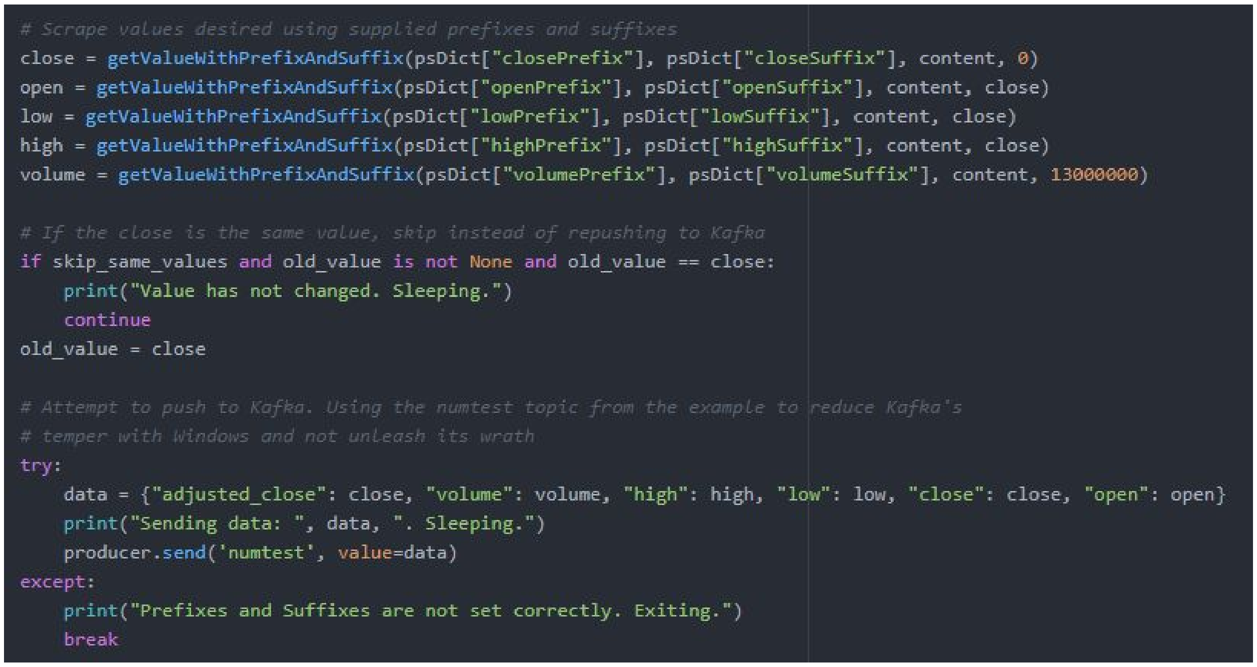
There are two portions of the Stock Prediction file. The first portion makes the model using a combination of Pandas dataframes and CSV ingestion and Scikit Learn's machine learning-oriented methods. After the dataframe is loaded, some basic modifications are made to the dataframe to create a feature set. For this portion I followed the combined instructions cited in the "Towards Data Science" articles, with a slight change to suit the second portion of the program.

After the model is made, the Stock Predictor enters a program loop as it waits to consume content from Kafka. Content pushed to the Kafka topic should be in a JSON format with values for the low, high, close, volume, and adjusted close for the day. That is then loaded into a new dataframe that leverages the model to predict based on that new input. The prediction is outputted, and it continues to wait for new messages at the Kafka topic.



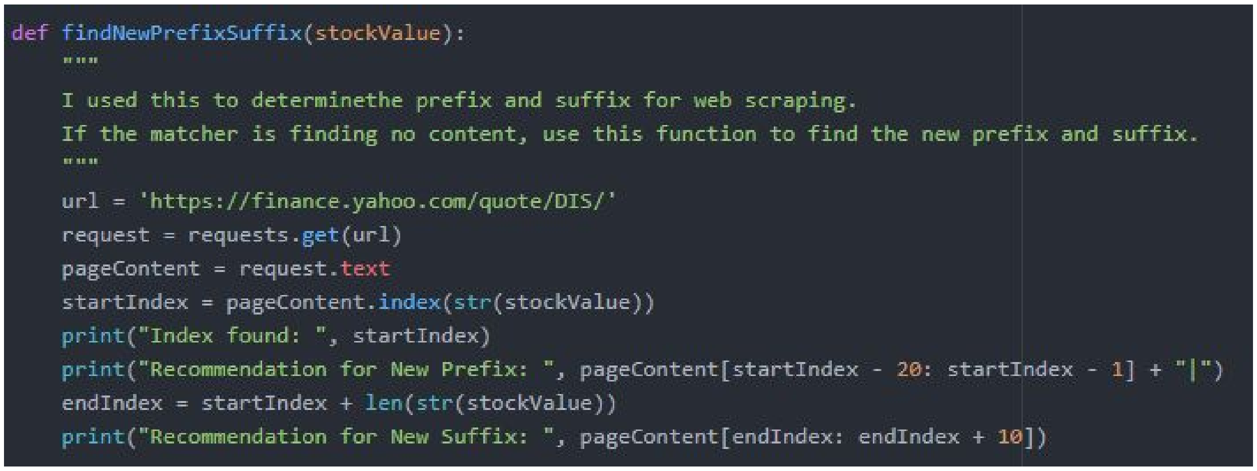
```
for message in consumer:
    loaded_message = loads(message.value)
    print("Loaded message: ", message.value)
    try:
        adj_close = loaded_message["adjusted_close"]
        volume = loaded_message["volume"]
        high = loaded_message["high"]
        low = loaded_message["low"]
        close = loaded_message["close"]
        open = loaded_message["open"]

        hlp = getHLP(high, low, close)
        change = getChange(close, open)
        df2 = pandas.DataFrame([[adj_close, volume, hlp, change]], columns=["Adj Close", "Volume", "hlp", "change"])
        scaled_df2 = scaler.transform(df2)
        print("Predicting with given data from Kafka stream: ", df2)
        print(model.predict(np.array(scaled_df2)))
    except:
        print("Error: Data pushed to Kafka topic numtest does not conform.")
```

In the Kafka Host file, the primary function is to take a set of prefixes and suffixes and use those with a Regex Expression to pull out the appropriate financial data to push onto the Kafka topic. The prefixes and suffixes are hardcoded, as the values are prone to change and need to be manually tweaked if new situations arise. Thus far, I have ironed out initial bugs by making adjustments to the Regex Expression, but this is definitely a spot that I would like to focus on if I had more time on the project.

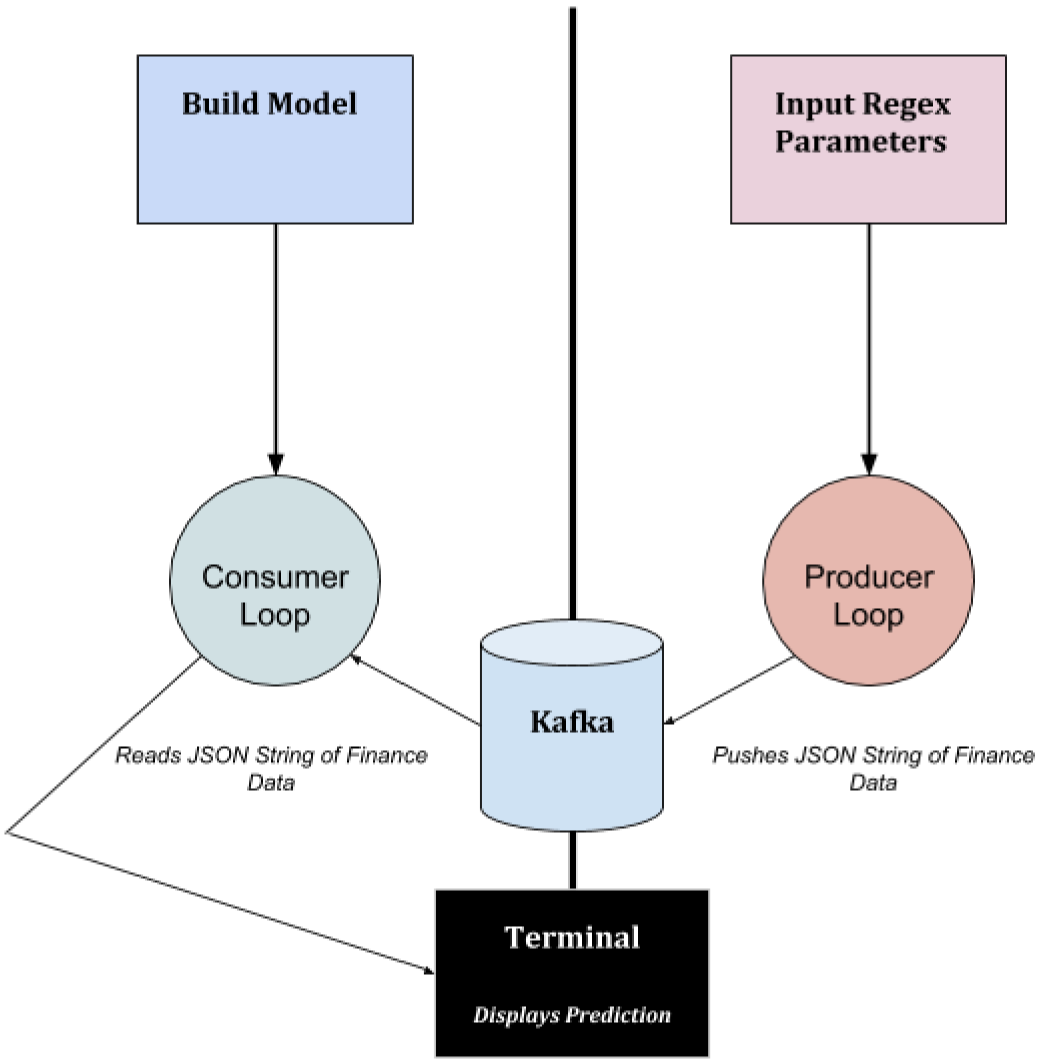


```python
# Scrape values desired using supplied prefixes and suffixes
close = getValueWithPrefixAndSuffix(psDict["closePrefix"], psDict["closeSuffix"], content, 0)
open = getValueWithPrefixAndSuffix(psDict["openPrefix"], psDict["openSuffix"], content, close)
low = getValueWithPrefixAndSuffix(psDict["lowPrefix"], psDict["lowSuffix"], content, close)
high = getValueWithPrefixAndSuffix(psDict["highPrefix"], psDict["highSuffix"], content, close)
volume = getValueWithPrefixAndSuffix(psDict["volumePrefix"], psDict["volumeSuffix"], content, 13000000)

# If the close is the same value, skip instead of repushing to Kafka
if skip_same_values and old_value is not None and old_value == close:
    print("Value has not changed. Sleeping.")
    continue
old_value = close

# Attempt to push to Kafka. Using the numtest topic from the example to reduce Kafka's
# temper with Windows and not unleash its wrath
try:
    data = {"adjusted_close": close, "volume": volume, "high": high, "low": low, "close": close, "open": open}
    print("Sending data: ", data, ". Sleeping.")
    producer.send('numtest', value=data)
except:
    print("Prefixes and Suffixes are not set correctly. Exiting.")
    break
```

There is a convenience method that takes a provided value and searches the content of the URL for it, and will give the surrounding characters. The key flaw is the reliance on knowing the current values that you need, and those values are prone to change as there is no guarantee of the format on the page. Those surrounding characters are outputted, and can then be pasted into the main block.



```python
def findNewPrefixSuffix(stockValue):
    """
    I used this to determinethe prefix and suffix for web scraping.
    If the matcher is finding no content, use this function to find the new prefix and suffix.
    """
    url = 'https://finance.yahoo.com/quote/DIS/'
    request = requests.get(url)
    pageContent = request.text
    startIndex = pageContent.index(str(stockValue))
    print("Index found: ", startIndex)
    print("Recommendation for New Prefix: ", pageContent[startIndex - 20: startIndex - 1] + "|")
    endIndex = startIndex + len(str(stockValue))
    print("Recommendation for New Suffix: ", pageContent[endIndex: endIndex + 10])
```



The primary test input during development was the initial data obtained from Yahoo! Finance's historical data of Disney's Stock (DIS). This CSV had data on the low, high, volume, and adjusted close for the stock going back nearly a century. That data was used to test Pandas dataframes and creation of the Linear Regression model. Yahoo! Finance's web page that I found the downloadable historical data on also contained the values the model needed to predict on, and so that url ('https://finance.yahoo.com/quote/DIS/') was used to test against for scraping and then providing new values to the created model.

One of the most helpful things I pulled from the class into this project was the experience with Apache Spark hands on. Though I did not use Spark for this project, that was due to Pandas providing the same ability to manage dataframes in a very well-integrated Python library. The Apache Spark homework helped with my understanding of how to use dataframes, as my previous usages of Scikit Learn I avoided them. This plays into a second point that the more theoritical questions about Apache Spark and Map Reduce gave me additional background knowledge on the purpose and incentives for using dataframes to manipulate data.

While not direct topics from the syllabus, there were some repeated elements of the class that I felt played into the development and structure of the project. One I kept heavily in mind was minimization of I/O even with databases, and that encouraged my program loops to avoid unnecessary I/O and other operations against my earlier assumptions that databases would manage I/O more effectively on their own. Additionally, we discussed data analytics often in class, and putting that into practice in this project with Pandas, Apache Kafka, and Scikit Learn helped to provide a practical use for larger amounts of data that is a concern in this class.

Considering the rapid startup that this project had, I believe six months of full-time work on the project would yield effective natural language processing for the feature set, multiple stocks incorporated into the models, and possibly even a web-based user interface using Django. That could likely be done within the first two months if focused on, and the following four months would be tuning the model and the feature set to try and increase accuracy. With those goals in mind, I am mostly satisfied with what I created in these short weeks, but would be excited to return to the project again. Apache Kafka, Pandas, and webscraping were all new technologies and concepts to me, and I believe I would enjoy using them again for future projects or to reach the goals listed above.

Sources

Xiaodong Li et. al., “Empirical analysis: stock market prediction via extreme learning Machine” EXTREME LEARNING MACHINE AND APPLICATIONS. Springer-Verlag. London. 2014.

Schumaker, R., & Chen, H., “Textual Analysis of Stock Market Prediction Using Financial News Articles”. 12th Americas Conference on Information Systems (AMCIS) 2006.

Jae Won Lee, “Stock Price Prediction using Reinforcement Learning”. Industrial Electronics. IEEE International Symposium Proceedings. 2001.

Stack Overflow: Artificial intelligence - “source of historical stock data” [closed]:
https://stackoverflow.com/questions/754593/source-of-historical-stock-data
Lucas Kohorts, Towards Data Science - “Predicting Stock Prices with Python”
https://towardsdatascience.com/predicting-stock-prices-with-python-ec1d0c9bece1
Vincent Tatan, Towards Data Science - “In 12 Minutes: Stocks Analysis with Pandas and Scikit Learn”
https://towardsdatascience.com/in-12-minutes-stocks-analysis-with-pandas-and-scikit-learn-a8d8a7b50ee7