Capstone 3 Project Report:

The Problem:

The problem that I explored in this project was complaint classification. In my career, I used to support a call center organization and I was in conversations with senior leaders about complaints that were coming in. They would personally read dozens of complaints coming in against the bank to try and gain insight into how their team can operate better. The complaints they were reading were all targeted towards the customer service agents and my data in this project has complaints against financial institutions across several different lines of business, but the value proposition remains the same. A model that can classify complaints to the correct line of business while also analyzing the topic of the complaint can not only save executives valuable time, but also increase satisfaction of customers because it gives the business the ability to understand the root of the complaint and who should be driving toward a solution.

Data Collection:

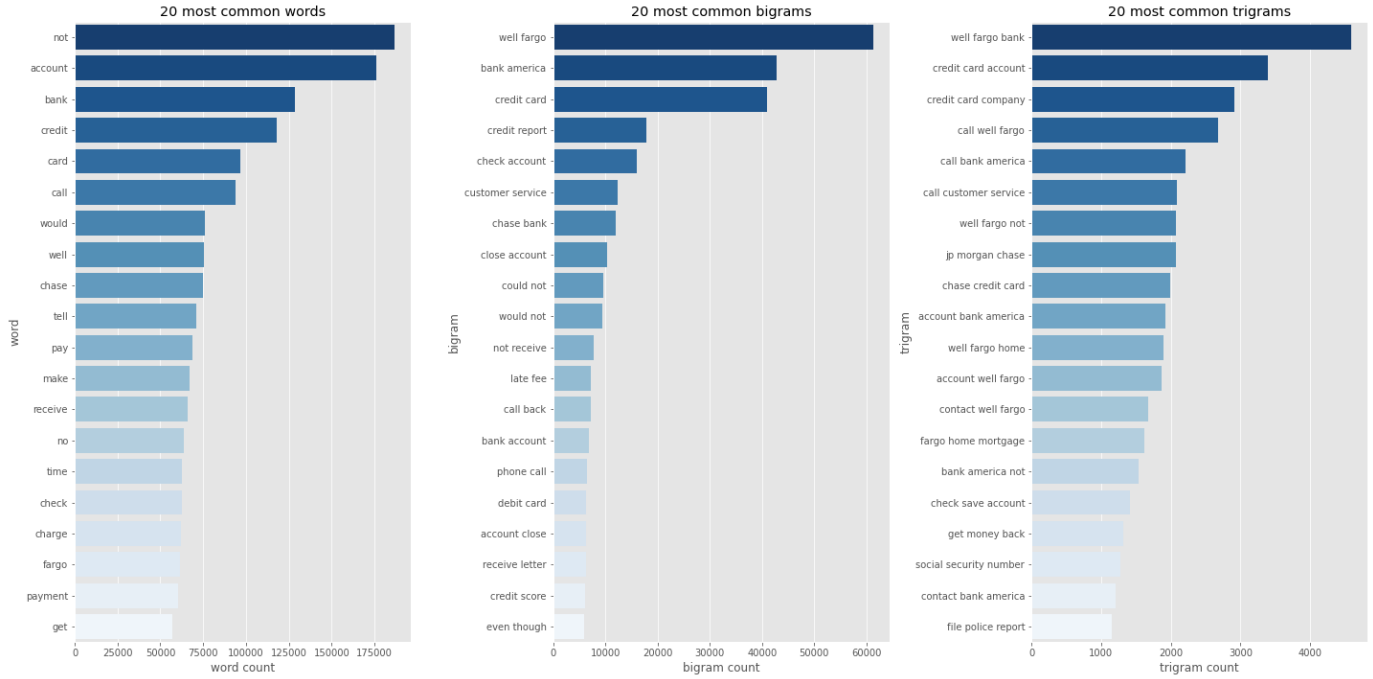
The data I used for this project came from the Consumer Financial Protection Bureau (CFPB) website and consisted of customer complaints across dozens of financial institutions and products. This dataset is quite large with 1.8 million complaints; however, most did not have a written complaint and thus could not be used for this project. Once I filtered those out, I had about 600k records, each with a unique written complaint from a customer. For the scope of this project, I decided to focus on 4 large American banks: JP Morgan Chase, Bank of America, Wells Fargo and Citibank because they offer a more diverse slate of products and thus have more diverse complaints. When I was initially including all of the companies, I found most of the complaints to be related to a customer's credit or their credit report. While only including the banks data, I turned my attention to the products that I was trying to predict. There were initially 17 products, many of which were redundant (i.e. 'Student Loan' and 'Consumer Loan' or 'Money Transfer' and 'Money transfer, virtual currency, or money service') so I consolidated the 17 products down to a much more concise 6 products. After ensuring I did not have any duplicate records, my next step was to preprocess the text data prior to running a vectorizer. I wrote a function that tokenized, lemmatized and removed stop words, then converted the list back to a string so I could use it as input to my Tf-Idf vectorizer later in the process. The remainder of my data wrangling ventures were ultimately not fruitful so I won't go into too much detail, but I will briefly explain some of my aspirations later in this report.

Exploratory Data Analysis:

The analysis of my features and the different variables in my dataset were quite eye-opening for me and revealed characteristics of the complaints that otherwise would go unnoticed. My initial strategy for EDA was fairly basic with the most common words, bigrams and trigrams and what I found was not overly surprising. Most of the results displayed in Figure 1 below were either a name of one of the banks or words, bigrams or trigrams related to one of

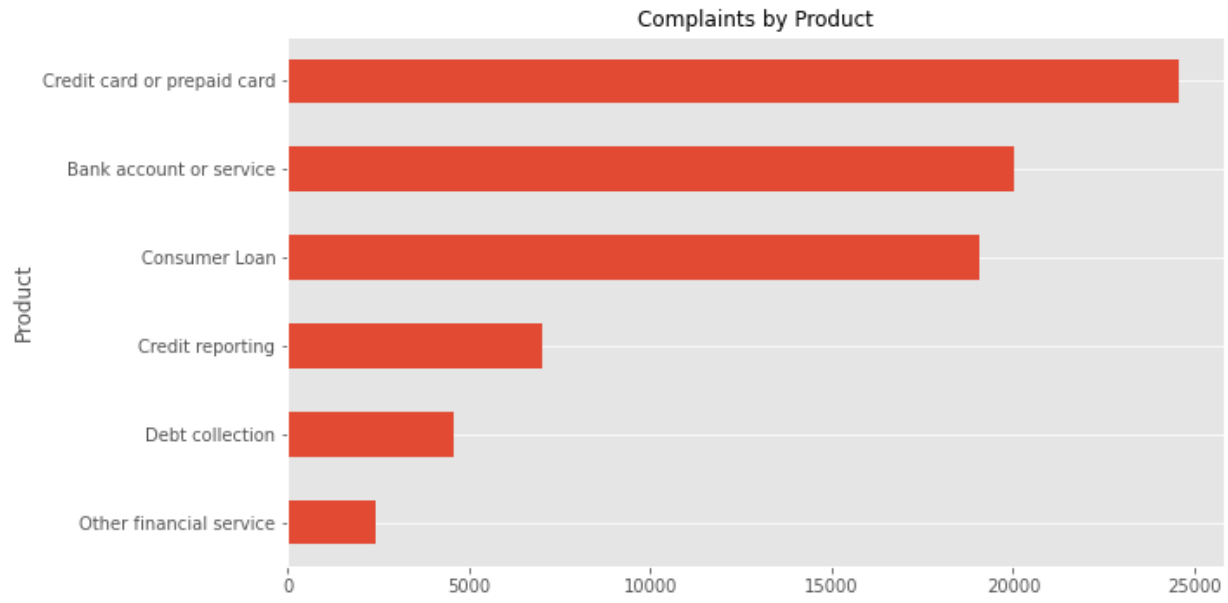
the products I'm trying to predict like 'credit', 'account', 'loan', etc. In the most common words, we can see that the most common word is 'not'. In most cases, this word would be included in our 'stop words' but I left it in to attempt to predict sentiment and I though including words like 'not', 'no' or 'don't' would be valuable to this effort. Ultimately, it was not and I removed words like it for the modeling stage of the project.

Figure 1:



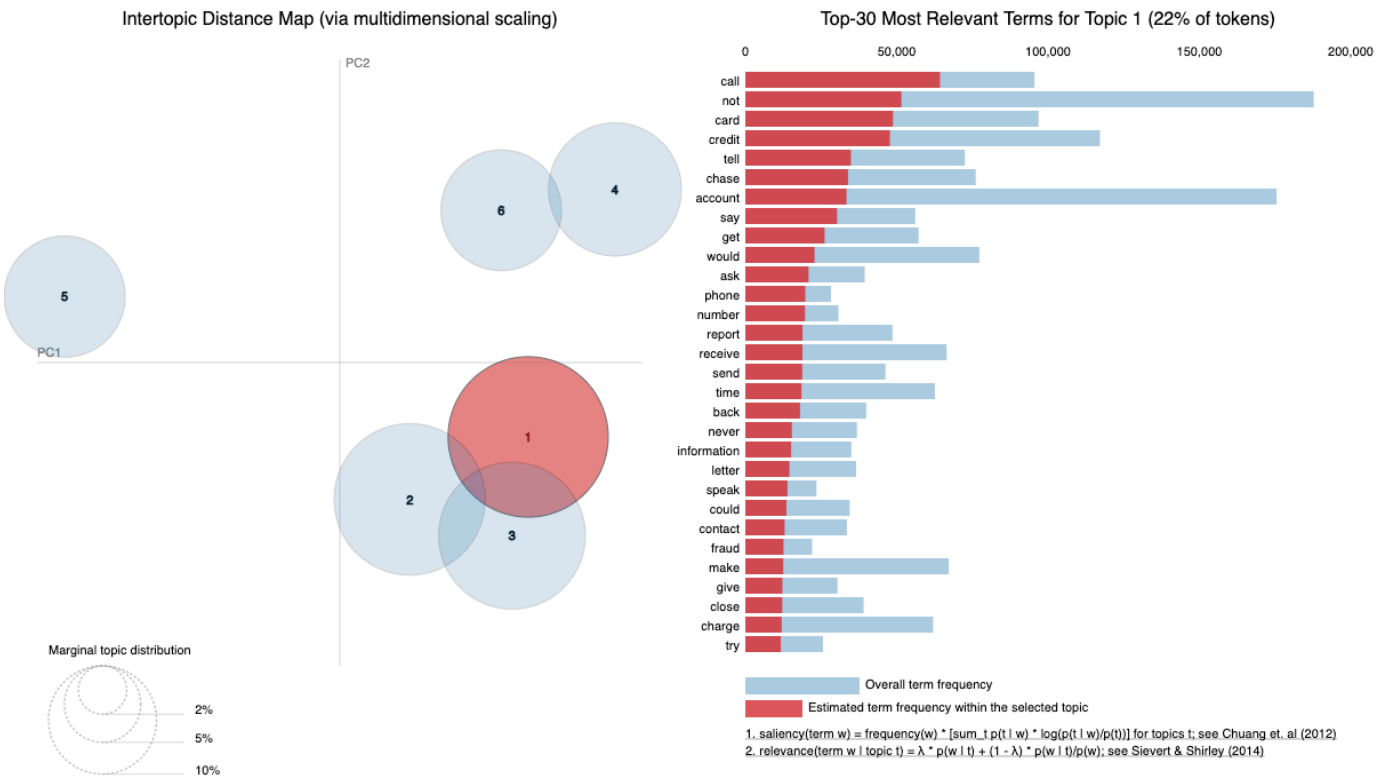
I was also curious about the distribution of complaints across the different banks as well as the products I was trying to predict. What I found was that there was a similar number of complaints against each bank, but I had an imbalanced dataset shown by Figure 2 below.

Figure 2:



Although not all models require a balanced dataset, one that I wanted to use in this project does (Multinomial Naïve Bayes), so to combat the imbalanced dataset, I decided I would oversample the data and balance the classes for prediction. More on this in the modeling section later. The most valuable revelation from my EDA in this project was a topic modeling exercise using a Latent Dirichlet Allocation. This allowed us to group complaints in clusters using a group of common words that were common among that topic. In Figure 3 below, we can see that the largest cluster consists of words like 'call', 'tell', 'say' and 'phone'. This indicates that the topic of the complaint was not entirely focused on the financial product that I am predicting, but was actually more of a reflection of the customer's dissatisfaction with the customer service as the institution when they called in. This discovery could save a company thousands of dollars from trying to uncover the root cause of complaints against a particular line of business when the true issue lies in the customer service agent's ability to work with the customer, help solve their issue and maintain their satisfaction.

Figure 3:



Preprocessing:

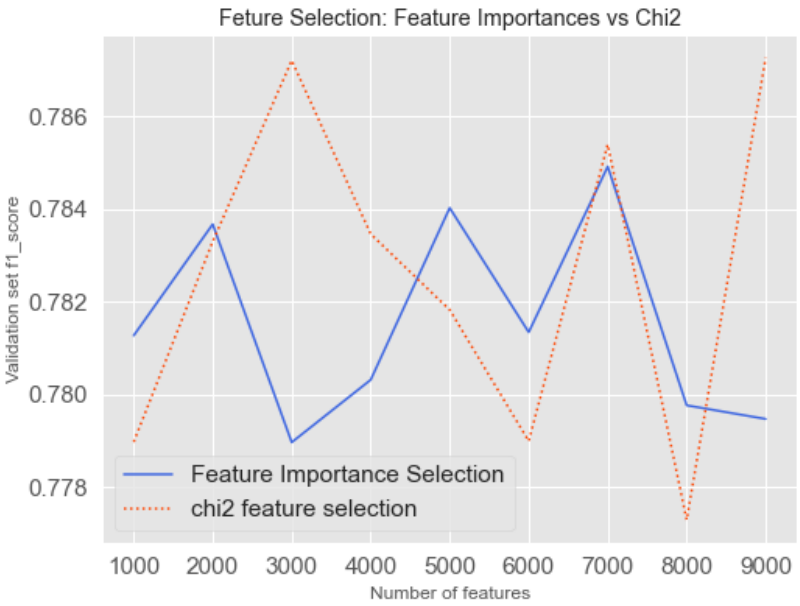
Following my analysis of features and discovery of new possible topics, I moved on to preprocess my data for modeling. To do this, I first split my data to create a test set to be used for validation. I then moved on to create a Tf-Idf vectorizer of single words, bigrams and trigrams. To accommodate constraints on my local machine and not force run time to go through the roof, I limited my vectorizer to 10,000 features. To address the issue of imbalance classes as mentioned above, I wanted to oversample my training dataset so that I had an equal number of complaints from each class. I used the Synthetic Minority Oversampling Technique (SMOTE) and ended up with 30,282 records (5,047 from each class) in my dataset used for modeling. With a complete vectorizer and a balanced dataset, I was ready to work on feature selection to reduce dimensionality.

Feature Selection:

In an effort to reduce dimensionality, I tried a handful of different techniques, but what I chose to do was compare a couple different strategies. One was the SelectFromModel approach which used feature importances or coefficients (depending on the model) to choose n features. I looped over a number of different possibilities to see how I could select the relevant features while maintaining model performance. What I found was that chi2 was the more effective approach, so I used that technique to select a subset of the original 10,000 features from my Tf-Idf vectorizer. Not only was I able to retain the performance of my models which

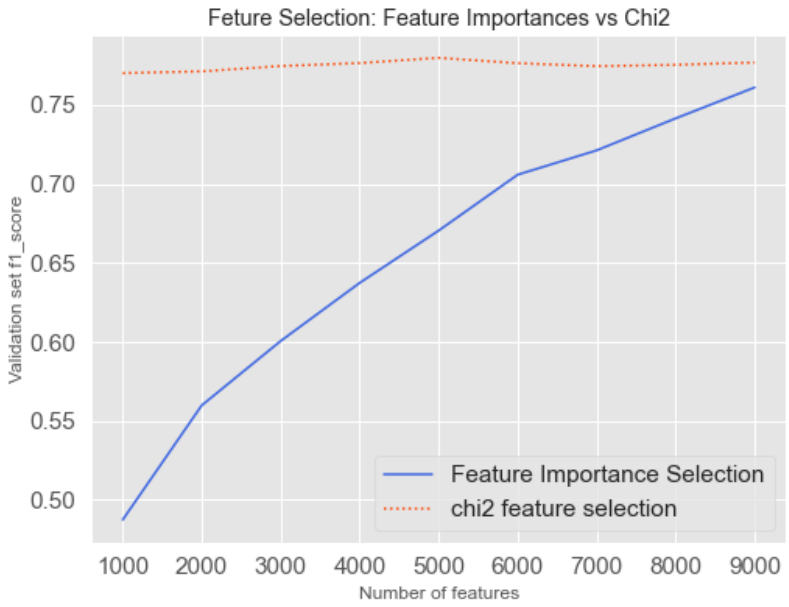
used all features, I was also able to improve runtime substantially especially in the case of the Random Forest Classifier. Figure 4 and 5 demonstrate the weighted f1-score outputs of the Random Forest and Naïve bayes models using chi2 (dotted orange line) vs SelectFromModel (solid blue line) as feature reduction techniques.

Figure 4:



In the case of the Random Forest feature selection, I chose to use the chi2 method and used 3000 features where I saw a peak in performance. Although the performance increase isn't huge, it is still an improvement over the other options.

Figure 5:



With regard to the feature selection with the Naïve Bayes model shown above, it is clear that using the coefficient to select features was not a valuable exercise, so I used the peak of the chi2 analysis which looked to be around 5000 features and used those in my modeling dataset

Modeling:

In this project, the models I used were the Random Forest Classifier, Multinomial Naïve Bayes Classifier and a Logistic Regression. When testing the models, due to the massize dimensionality of the data, the logistic regression was not a good option as it took over 20 minutes to run. So although I include some performance metrics from it, I ruled that model out of my analysis fairly early and did not pursue it due to computational constraints. With the random forest, I decided to use a GridSearchCV to tune my hyperparameters which resulted in slight improvement in my model, although most of the improvement came from the feature selection stage. With the Naïve Bayes, I tested different values for the alpha and determined that a lower alpha was a better fit for this model. I also ran a 3-fold cross validation to ensure I was not overfitting and the results confirmed that.

Evaluation:

The evaluation metric I chose was a weighted f1-score which is just the harmonic mean between precision and recall. The reason I chose this was as opposed to some classification problems that present major repercussions with a false positive or a false negative, the consequences were not as dire for this project which drove my decision to assess based on the average of both weighted across the 6 classes I was predicting. At the end of the day, the Random Forest and Naïve Bayes models had quite similar outputs (weighted f1-score within

0.001 of each other). The recommendation I would make to a client would be to use the Naïve Bayes because although it had a slightly lower evaluation metric, the runtime was incredibly efficient.

Figure 6:

Model	Weighted f1	Runtime (mins)
Random Forest Classification	0.787	1.0
Naive Bayes Classification	0.786	0.0
Logistic Regression	0.786	22.9

Value to Customer:

The initial goal of this project was to classify complaints to the different products then identify which were the most severe to be able to identify and solve the most pressing issues. As I worked through the project, it became clear that the assignment of a severity rating would be beyond the scope of this project. I tried a handful of different strategies which I detail below, but they were not successful. Through my EDA process, I ran a latent Dirichlet allocation which allowed me to assign a topic to the complaints to see if they were in alignment with the classification model. What I found that the most common topic was not focused on one of the products I was predicting. But rather, the underlying issue was with the customer service from the financial institution that drove the complaint. This analysis can be crucial for executive to avoid making false assumptions about a line of business with high complaint volume and dig more into the true root cause.

Aspirations/Failures:

As mentioned above, the biggest disappointment from this project was not being able to effectively predict severity to leverage as an indicator of complaints that should be dealt with. I made several attempts detailed below but the crux of the issue was that it is incredibly difficult to analyze which complaints are “severe” enough to address when every complaint in the dataset is highly negative and deserves attention. I tried things like taking a sample of the data, reading the complaints, and assigning labels to establish ground truth followed by creating numerical features to aid in predicting based on my new found ground truth. I also tried using sentiment analysis packages like TextBlob and VADER to assign a polarity rating to the data. However, I believe my downfall was that all of the complaints were very negative and without a solid ground truth in the form of labels, prediction would be well beyond this project’s scope.

Explore Further:

- Neural network for product classification

- Pyspark to include all data – running locally would have taken hours for simple commands like running a Tf-Idf on all complaints. I ended up having to take a sample of the data so that the code wasn't taking several hours to run.