**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:**

**Chart, bar chart

Description automatically generated**

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:**

Chart, bar chart, funnel chart

Description automatically generated

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:**

Chart, bubble chart

Description automatically generated

**Preprocessing:**

Tf-Idf, SMOTE

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.

Aspirations/Failures:

* Tried to do sentiment analysis – harder with no labels and when everything is negative (can’t tell negative against very negative, computer doesn’t understand how some banking products are more severe than others)
  + Create additional numerical features (word count, sentence count, special characters, etc) to try to aid in predicting severity
  + Tried to manually assign labels to a subset of the data to use as ‘ground truth’ for sentiment analysis
  + Using TextBlob and Vader, tried to analyze polarity and subjectivity to gain insight into which complaints should be addressed

EXPLORE FURTHER:

* Neural network
* Pyspark to include all data – running locally would have taken hours for simple commands