Data Mining Final Project

Dayeon Kim, Hyunyoung Shin

12/19/2019

#Final Project ## Dayeon Kim(dk3115), Hyunyoung Shin(hs3158)

Introduction

For our final project, we decided to conduct text analysis and sentiment analysis on the Consumer Complaints dataset from IBM. The dataset involves various consumer complaints received by financial corporations, and are grouped by different product categories and sub-issues. We will be focusing on the 'Consumer Complaints Narrative' column that contains a text narrative of the actual complaints that the consumers submitted. We combined sentiment analysis data with geographical data to explore if certain regions of the country demonstrated higher levels of consumer dissastisfaction, and we also conducted topic modeling with the Latent Dirichlet Allocation method to see if through the narratives we can predict which product category each narrative should be assigned to. The goal of this project is to deliver useful business insights for IBM and companies involved.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidytext)
library(textdata)
library(ggplot2)
library(pacman)
p_load(caret,
       dplyr,
       ggplot2,
       kableExtra,
       stringr,
       tidytext,
       tidyr,
       tm,
       topicmodels,
       wordcloud)
library(tidytext)
library(stringr)
```

Loading Data

```
test <-  read.csv("Consumer_Complaints.csv", header=T, na.strings=c(""))
df <- test[!is.na(test$Consumer.complaint.narrative),]
nrow(test)
```

```
## [1] 1397780
```

```
nrow(df)
```

```
## [1] 445931
```

Because we will be conducting text analysis on the 'Consumer Complaint Narrative' column, we started out by removing rows that have NAs in the column. The initial dataset had 1397780 observations, but after removing th NAs from the column we had 445,931 observations.

```
complaint <- subset(df[1:25000,])
```

We thought that still the dataset with 445,931 observations was too large to conduct text analysis, so we got a sample dataset with 25,000 observations using the subset function.

```
head(complaint)
```

```
##     Date.received
## 649    09/26/2019
## 654    09/26/2019
## 693    09/26/2019
## 768    09/26/2019
## 915    09/25/2019
## 917    09/25/2019
##                                                                                 Product
## 649                                                                    Debt collection
## 654 Credit reporting, credit repair services, or other personal consumer reports
## 693 Credit reporting, credit repair services, or other personal consumer reports
## 768 Credit reporting, credit repair services, or other personal consumer reports
## 915                                                                    Debt collection
## 917 Credit reporting, credit repair services, or other personal consumer reports
##          Sub.product
## 649    I do not know
## 654 Credit reporting
## 693 Credit reporting
## 768 Credit reporting
## 915       Other debt
## 917 Credit reporting
##                                                                                                 Issue
## 649                                                                                Communication tactics
## 654 Problem with a credit reporting company's investigation into an existing problem
## 693                                                                 Incorrect information on your report
## 768                                                                 Incorrect information on your report
## 915                                                                                Communication tactics
```

```
## 917                                          Incorrect information on your report
##                                                Sub.issue
## 649                              Frequent or repeated calls
## 654 Their investigation did not fix an error on your report
## 693                            Account information incorrect
## 768            Old information reappears or never goes away
## 915                              Frequent or repeated calls
## 917                     Public record information inaccurate
##
## 649
## 654
## 693
## 768
## 915
## 917 I was sitting at home and I receive an alert that my credit report changed on Transunion, I look
##                                                                                Company.public.response
## 649 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 654 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 693 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 768 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 915 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 917 Company has responded to the consumer and the CFPB and chooses not to provide a public response
##                                                Company State ZIP.code
## 649                         Valentine and Kebartas, Inc.    FL    334XX
## 654 Fidelity National Information Services, Inc. (FNIS)    CA    937XX
## 693           HCFS Health Care Financial Services, Inc.    TX     <NA>
## 768                      Ability Recovery Services, LLC    TX     <NA>
## 915                                   ProCollect, Inc    TX    770XX
## 917              TRANSUNION INTERMEDIATE HOLDINGS, INC.    FL     <NA>
##     Tags Consumer.consent.provided. Submitted.via Date.sent.to.company
## 649 <NA>           Consent provided           Web           09/26/2019
## 654 <NA>           Consent provided           Web           09/26/2019
## 693 <NA>           Consent provided           Web           09/26/2019
## 768 <NA>           Consent provided           Web           09/26/2019
## 915 <NA>           Consent provided           Web           09/25/2019
## 917 <NA>           Consent provided           Web           09/25/2019
##        Company.response.to.consumer Timely.response. Consumer.disputed.
## 649         Closed with explanation              Yes                N/A
## 654         Closed with explanation              Yes                N/A
## 693         Closed with explanation              Yes                N/A
## 768         Closed with explanation              Yes                N/A
## 915         Closed with explanation              Yes                N/A
## 917 Closed with non-monetary relief              Yes                N/A
##     Complaint.ID
## 649      3386898
## 654      3387731
## 693      3387338
## 768      3386849
## 915      3385404
## 917      3385557
```

```
names(complaint)[names(complaint) == 'Consumer.complaint.narrative'] <- 'Narrative'
colnames(complaint)
```

```
##  [1] "Date.received"                "Product"
##  [3] "Sub.product"                  "Issue"
##  [5] "Sub.issue"                    "Narrative"
##  [7] "Company.public.response"      "Company"
##  [9] "State"                        "ZIP.code"
## [11] "Tags"                         "Consumer.consent.provided."
## [13] "Submitted.via"                "Date.sent.to.company"
## [15] "Company.response.to.consumer" "Timely.response."
## [17] "Consumer.disputed."           "Complaint.ID"
```

Because the column name 'Consumer.complaint.narrative' was too complicated, we renamed it to "Narrative"

Converting the "Narrative" column to character

The "narrative" format was originally factor. But in order to apply text analysis, the column should be in character format, so we converted it to character.

```
complaint$Narrative <- as.character(complaint$Narrative)
```

Tokenizing Narrative Text

```
tidy_complaint <- unnest_tokens(complaint, output = "word", input = "Narrative")
tidy_complaint <- anti_join(tidy_complaint, stop_words)
```

```
## Joining, by = "word"
```

```
test <- tidy_complaint$word
#removing "xxxx" from the narrative column
edit <- str_replace_all(string = test, pattern = "[xxxx]", replacement = "")
edit <- str_squish(edit)
tidy_complaint$word <- edit
```

We tokenized the narrative column using the "unnest_tokens" function, and removed stop words. The new column with tokenized text is named "word". Also, many of the complaints involved "xxxx", which does not have any useful meaning, so we removed "xxxx".

Wordcloud

```
library(wordcloud)

tidy_complaint %>%
  count(word) %>%
  with(wordcloud(word, n, max.words = 100))
```

To get a general idea on which words appear frequently in the narrative column, we created a wordcloud.

Sentiment Analysis with Afinn & NRC

```
library(textdata)
afinn <- get_sentiments("afinn")
head(afinn)
```

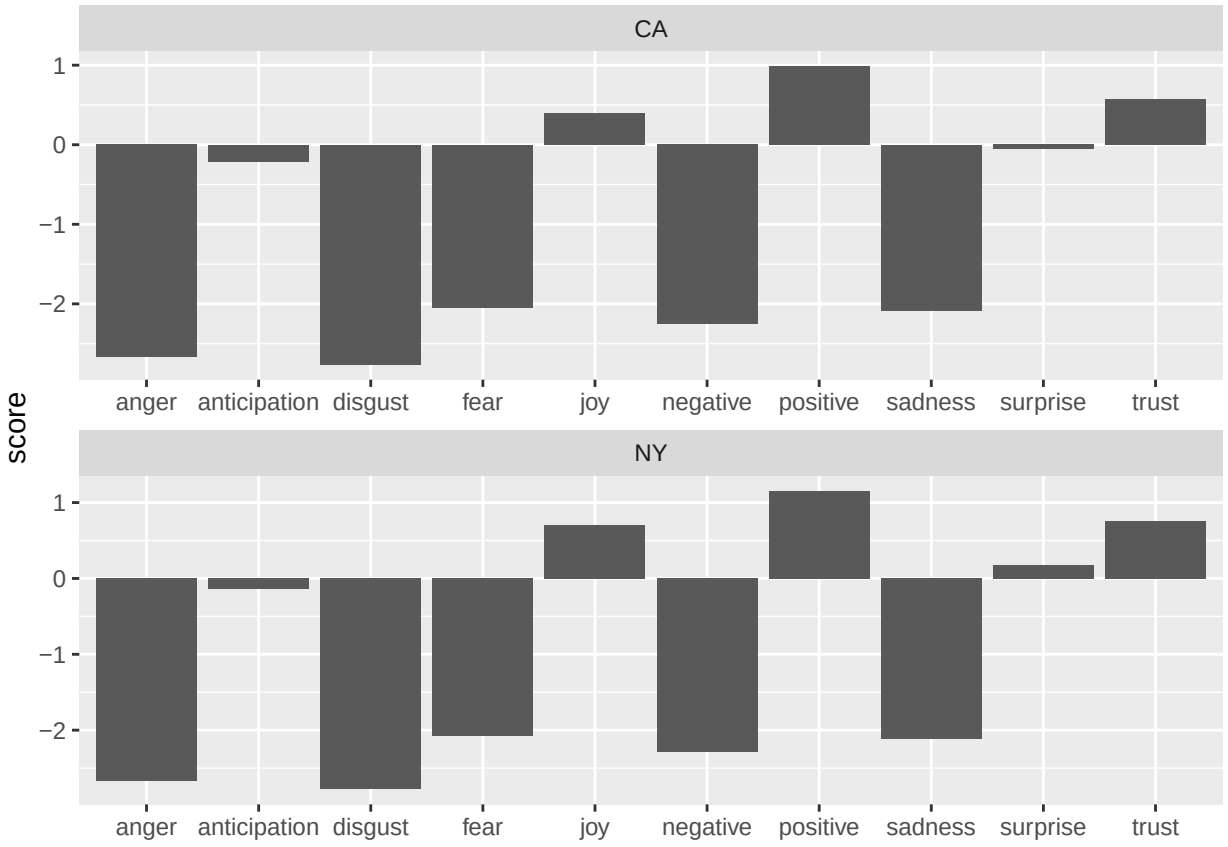
```
## # A tibble: 6 x 2
##   word       value
##   <chr>      <dbl>
## 1 abandon       -2
## 2 abandoned     -2
## 3 abandons      -2
## 4 abducted      -2
## 5 abduction     -2
## 6 abductions    -2
```

```
nrc <- get_sentiments("nrc")
head(nrc)
```

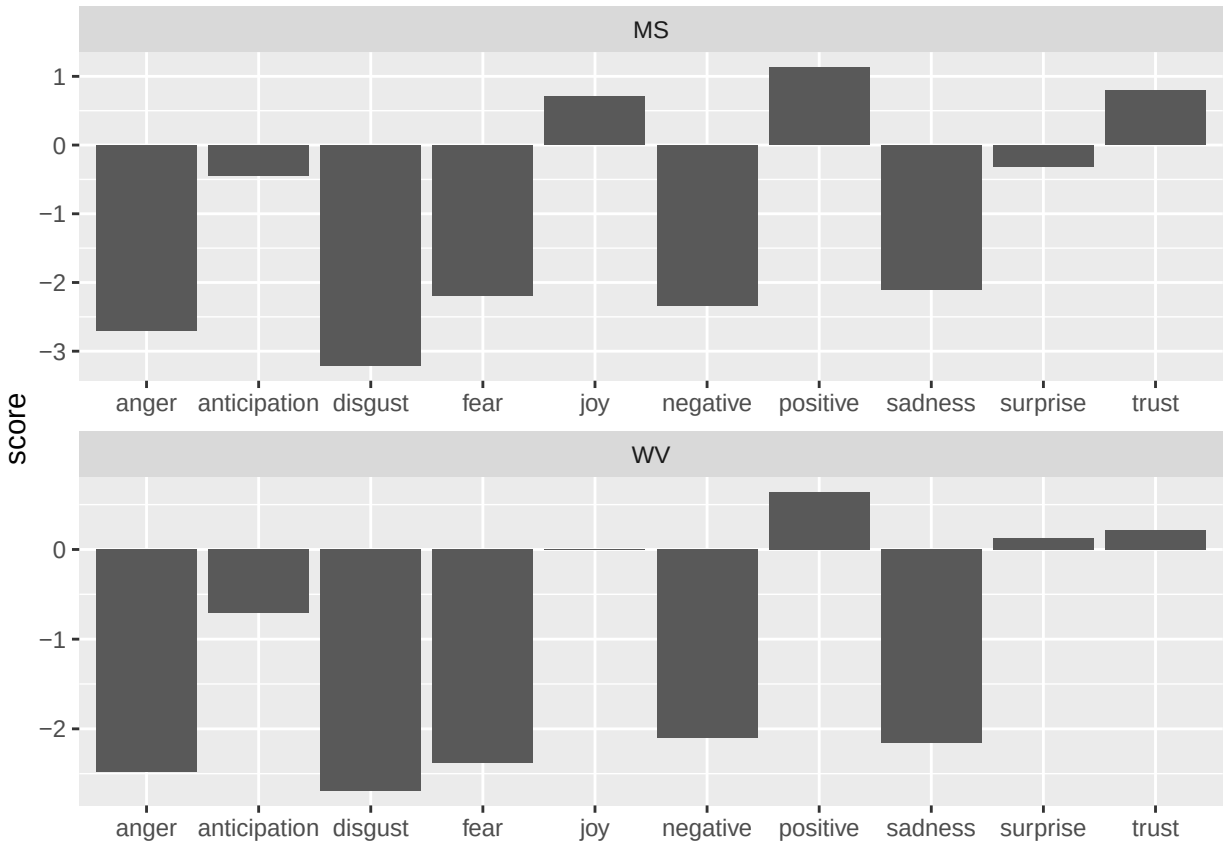
```
## # A tibble: 6 x 2
##   word      sentiment
##   <chr>     <chr>
## 1 abacus    trust
## 2 abandon   fear
## 3 abandon   negative
## 4 abandon   sadness
## 5 abandoned anger
## 6 abandoned fear
```

For our sentiment analysis on the narrative column, we decided to use a dictionary approach, and selected Afinn and NRC. We wanted to assign a "negative sentiment score" to each complaint narrative, and because Afinn is a lexicon of English words rated for valence with an integer between -5 and 5, we thought it was the most suitable dictionary to use. Also, NRC dictionary inclues 8 different categories of emotions instead of the binary distinction between positive and negative, so we decided to try with NRC to see if the complaints showed specific negative emotions such as "anger".

```r
##compare rich and poor
library(ggplot2)

rich <- filter(tidy_complaint, State == "NY" | State == "CA")

rich <- rich %>%
    inner_join(nrc, by = "word") %>% inner_join(afinn, by = "word") %>% group_by(State, sentiment) %>% s

graph_rich <- ggplot(rich, aes(sentiment, score)) + geom_col(show.legend = FALSE) +
  facet_wrap(~State, ncol = 1, scales = "free") + labs(x = NULL)

graph_rich
```




```r
poor <- filter(tidy_complaint, State == "WV" | State == "MS")
poor <- poor %>%
    inner_join(nrc, by = "word") %>% inner_join(afinn, by = "word") %>% group_by(State, sentiment) %>% s
graph_poor <- ggplot(poor, aes(sentiment, score)) + geom_col(show.legend = FALSE) +
  facet_wrap(~State, ncol = 1, scales = "free") + labs(x = NULL)

graph_poor
```



We combined 3 data frames - afinn, nrc, tidy_complaint to see the different emotions in the NRC dicionary shown in the narratives, and their valence defined by Afinn. We wanted to see if different income levels of different states had an impact on the sentiment of the narratives, so we compared the sentiment analysis of the two richest states, New York and California, and the two poorest states, Mississippi and West Virginia. We compared the mean sentiment score of different emotions represented in the complaints submitted in the four states, but unfortunately we were unable to find a pattern between income level and sentiment. All 4 states showed higher levels of anger and disgust.

Combining Afinn and the Complaint Data Frame

```
complaint_afinn <- inner_join(tidy_complaint, afinn)
```

```
## Joining, by = "word"
```

```
head(complaint_afinn)
```

```
##   Date.received
## 1    09/26/2019
## 2    09/26/2019
## 3    09/26/2019
## 4    09/26/2019
## 5    09/26/2019
## 6    09/26/2019
##                                                                 Product
```

```
## 1                                                                 Debt collection
## 2                                                                 Debt collection
## 3                                                                 Debt collection
## 4 Credit reporting, credit repair services, or other personal consumer reports
## 5 Credit reporting, credit repair services, or other personal consumer reports
## 6 Credit reporting, credit repair services, or other personal consumer reports
##       Sub.product
## 1    I do not know
## 2    I do not know
## 3    I do not know
## 4 Credit reporting
## 5 Credit reporting
## 6 Credit reporting
##                                                                                 Issue
## 1                                                                Communication tactics
## 2                                                                Communication tactics
## 3                                                                Communication tactics
## 4 Problem with a credit reporting company's investigation into an existing problem
## 5                                                 Incorrect information on your report
## 6                                                 Incorrect information on your report
##                                                   Sub.issue
## 1                               Frequent or repeated calls
## 2                               Frequent or repeated calls
## 3                               Frequent or repeated calls
## 4 Their investigation did not fix an error on your report
## 5                             Account information incorrect
## 6            Old information reappears or never goes away
##                                                                                                 Company.public.response
## 1 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 2 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 3 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 4 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 5 Company has responded to the consumer and the CFPB and chooses not to provide a public response
## 6 Company has responded to the consumer and the CFPB and chooses not to provide a public response
##                                                   Company State ZIP.code Tags
## 1                          Valentine and Kebartas, Inc.    FL    334XX <NA>
## 2                          Valentine and Kebartas, Inc.    FL    334XX <NA>
## 3                          Valentine and Kebartas, Inc.    FL    334XX <NA>
## 4 Fidelity National Information Services, Inc. (FNIS)    CA    937XX <NA>
## 5           HCFS Health Care Financial Services, Inc.    TX     <NA> <NA>
## 6                       Ability Recovery Services, LLC    TX     <NA> <NA>
##   Consumer.consent.provided. Submitted.via Date.sent.to.company
## 1           Consent provided           Web           09/26/2019
## 2           Consent provided           Web           09/26/2019
## 3           Consent provided           Web           09/26/2019
## 4           Consent provided           Web           09/26/2019
## 5           Consent provided           Web           09/26/2019
## 6           Consent provided           Web           09/26/2019
##   Company.response.to.consumer Timely.response. Consumer.disputed.
## 1      Closed with explanation              Yes                N/A
## 2      Closed with explanation              Yes                N/A
## 3      Closed with explanation              Yes                N/A
## 4      Closed with explanation              Yes                N/A
## 5      Closed with explanation              Yes                N/A
```

```
## 6      Closed with explanation              Yes                N/A
##   Complaint.ID      word value
## 1      3386898   abusive    -3
## 2      3386898 deceitful    -3
## 3      3386898   prevent    -1
## 4      3387731  hardship    -2
## 5      3387338   resolve     2
## 6      3386849      debt    -2
```

Using left_join function, we combined the afinn data frame and the new complaint data frame with tokenized narrative so that each word in the narrative was assigned a valence.

Negative Sentiment Score by State

```
state_sentiment <- group_by(complaint_afinn, State) %>% summarize(score = mean(value, na.rm = TRUE))
```

```
## Warning: Factor `State` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
state_sentiment
```

```
## # A tibble: 58 x 2
##    State  score
##    <fct>  <dbl>
##  1 AE    -0.493
##  2 AK    -0.872
##  3 AL    -0.872
##  4 AP    -0.686
##  5 AR    -0.946
##  6 AZ    -0.926
##  7 CA    -0.906
##  8 CO    -0.947
##  9 CT    -0.755
## 10 DC    -0.954
## # ... with 48 more rows
```

```
arrange(state_sentiment, score)
```

```
## # A tibble: 58 x 2
##    State                                score
##    <fct>                                <dbl>
##  1 UNITED STATES MINOR OUTLYING ISLANDS -1.93
##  2 ND                                   -1.23
##  3 IA                                   -1.14
##  4 NE                                   -1.13
##  5 WV                                   -1.10
##  6 IN                                   -1.08
##  7 KS                                   -1.07
##  8 OH                                   -1.04
```

```
##  9 NH                                    -1.03
## 10 SD                                    -1.02
## # ... with 48 more rows
```

Using the "group_by" and "summarize" function, we computed the mean negative sentiment score by state. It turned out that out of the 50 states, North Dakota showed the highest score of negative sentiment.

Visualizing Negative Sentiment Score by State

We decided to visualize the negative sentiment score on a US map. In order to do so, we loaded R's maps package, which provides us with some pre-drawn map data.

```
library(maps)
us_states <- map_data("state")
head(us_states)
```

```
##        long      lat group order  region subregion
## 1 -87.46201 30.38968     1     1 alabama      <NA>
## 2 -87.48493 30.37249     1     2 alabama      <NA>
## 3 -87.52503 30.37249     1     3 alabama      <NA>
## 4 -87.53076 30.33239     1     4 alabama      <NA>
## 5 -87.57087 30.32665     1     5 alabama      <NA>
## 6 -87.58806 30.32665     1     6 alabama      <NA>
```

In order to get our own negative sentiment by state data on the map, we need to first merge the two data frames (map dataframe and our own data frame) together. For two data frames to merge, we need to have a column of variables that exactly correspond to one another. However, while the "maps" data frame lists states in their full name, our data frame used abbreviations. The function below is simply converting the state abbreviations into their full names in order to have exactly matching variables.

```
 #'x' is the column of a data.frame that holds 2 digit state codes
stateFromLower <-function(x) {
   #read 52 state codes into local variable [includes DC (Washington D.C. and PR (Puerto Rico)]
  st.codes<-data.frame(
                       state=as.factor(c("AK", "AL", "AR", "AZ", "CA", "CO", "CT", "DC", "DE", "FL", "GA
                                         "HI", "IA", "ID", "IL", "IN", "KS", "KY", "LA", "MA", "MD", "M
                                         "MI", "MN", "MO", "MS",  "MT", "NC", "ND", "NE", "NH", "NJ", "
                                         "NV", "NY", "OH", "OK", "OR", "PA", "PR", "RI", "SC", "SD", "T
                                         "TX", "UT", "VA", "VT", "WA", "WI", "WV", "WY")),
                       full=as.factor(c("alaska","alabama","arkansas","arizona","california","colorado",
                                        "connecticut","district of columbia","delaware","florida","georg
                                        "hawaii","iowa","idaho","illinois","indiana","kansas","kentucky"
                                        "louisiana","massachusetts","maryland","maine","michigan","minne
                                        "missouri","mississippi","montana","north carolina","north dakot
                                        "nebraska","new hampshire","new jersey","new mexico","nevada",
                                        "new york","ohio","oklahoma","oregon","pennsylvania","puerto ric
                                        "rhode island","south carolina","south dakota","tennessee","texa
                                        "utah","virginia","vermont","washington","wisconsin",
                                        "west virginia","wyoming"))
                        )
     #create an nx1 data.frame of state codes from source column
```

```
  st.x<-data.frame(state=x)
     #match source codes with codes from 'st.codes' local variable and use to return the full state nam
  refac.x<-st.codes$full[match(st.x$state,st.codes$state)]
     #return the full state names in the same order in which they appeared in the original source
  return(refac.x)

}
```

```
state_sentiment$State <- stateFromLower(state_sentiment$State)
```

Through the code above, we are replacing the State column with full state name variables.

```
head(state_sentiment)
```

```
## # A tibble: 6 x 2
##   State     score
##   <fct>     <dbl>
## 1 <NA>     -0.493
## 2 alaska   -0.872
## 3 alabama  -0.872
## 4 <NA>     -0.686
## 5 arkansas -0.946
## 6 arizona  -0.926
```

```
state_sentiment <- na.omit(state_sentiment)
head(state_sentiment)
```

```
## # A tibble: 6 x 2
##   State       score
##   <fct>       <dbl>
## 1 alaska     -0.872
## 2 alabama    -0.872
## 3 arkansas   -0.946
## 4 arizona    -0.926
## 5 california -0.906
## 6 colorado   -0.947
```

In order to combine two data frames using the left join function, we need to have the combining column with the mathching name. We are simply changing the column name "State" into "region", as the maps dataframe has "region" as its column name.

```
names(state_sentiment)[names(state_sentiment) == 'State'] <- 'region'
head(state_sentiment)
```

```
## # A tibble: 6 x 2
##   region      score
##   <fct>       <dbl>
## 1 alaska     -0.872
## 2 alabama    -0.872
## 3 arkansas   -0.946
## 4 arizona    -0.926
## 5 california -0.906
## 6 colorado   -0.947
```

Now, we are combining the two data frames using left join.

```
sentiment_combined <- left_join(us_states, state_sentiment)
```

```
## Joining, by = "region"
```

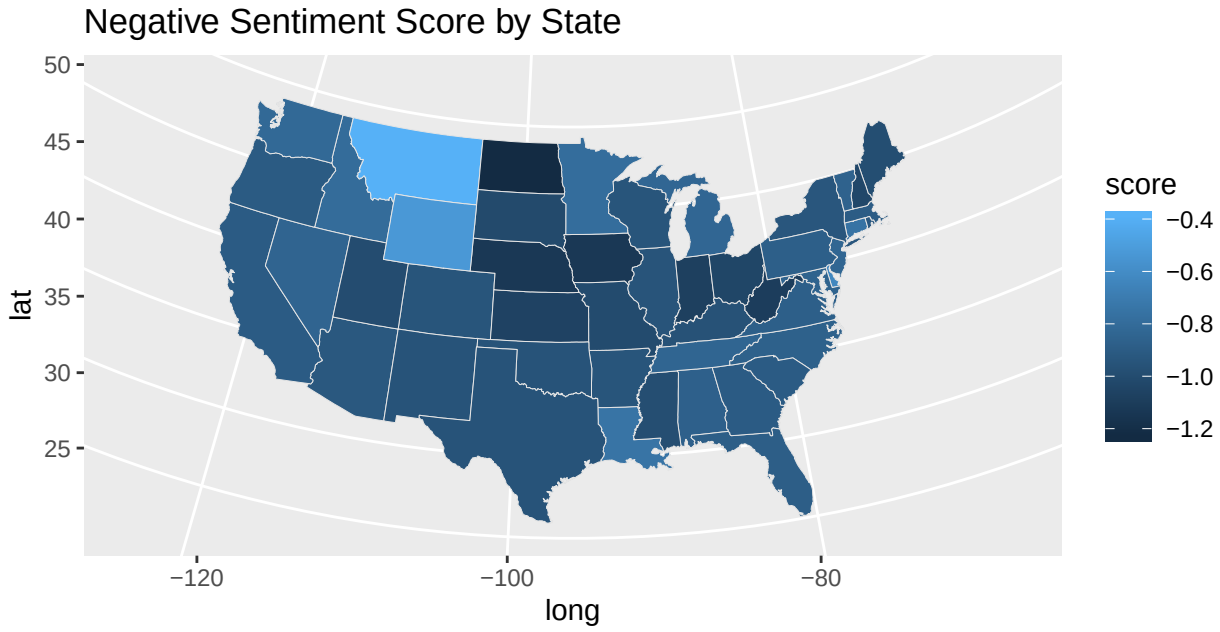
```
## Warning: Column `region` joining character vector and factor, coercing into
## character vector
```

```
head(sentiment_combined)
```

```
##        long      lat group order  region subregion      score
## 1 -87.46201 30.38968     1     1 alabama      <NA> -0.8724944
## 2 -87.48493 30.37249     1     2 alabama      <NA> -0.8724944
## 3 -87.52503 30.37249     1     3 alabama      <NA> -0.8724944
## 4 -87.53076 30.33239     1     4 alabama      <NA> -0.8724944
## 5 -87.57087 30.32665     1     5 alabama      <NA> -0.8724944
## 6 -87.58806 30.32665     1     6 alabama      <NA> -0.8724944
```

GGplot to Create a Map Visualization

```
library(mapproj)
p0 <- ggplot(data = sentiment_combined,
             mapping = aes(x = long, y = lat, group = group, fill = score))
p1 <- p0 + geom_polygon(color = "gray90", size = 0.1) +
  coord_map(projection = "albers", lat0 = 39, lat1 = 45) +
  labs(title = "Negative Sentiment Score by State")
p1
```



The map above shows that states are colored with densities depending on the negative sentiment score; states with higher negative sentiment score is colored more darkly. As our results showed, North Dakota (negative sentiment score of -1.23) has the darkest color, whereas states like Montana (-0.39) and Wyoming(-0.53) with low negative sentiment score were colored lightly. To IBM, this map can provide useful insights about how certain states are showing higher consumer dissatisfaction, and if negative sentiments are associated with issue urgency, complaints that are more urgent. Using this map data, IBM could look into whether CX resources are handled efficiently in states that showed higher negative sentiments.

Negative Sentiment Score by Product

```
product_sentiment <- group_by(complaint_afinn, Product) %>% summarize(score = mean(value, na.rm = TRUE))
k <- arrange(product_sentiment, score)
k
```

```
## # A tibble: 9 x 2
##   Product                                                       score
##   <fct>                                                         <dbl>
## 1 Checking or savings account                                  -0.988
## 2 Debt collection                                              -0.986
## 3 Credit reporting, credit repair services, or other personal consu~ -0.967
## 4 Credit card or prepaid card                                  -0.877
## 5 Money transfer, virtual currency, or money service           -0.866
## 6 Vehicle loan or lease                                        -0.718
## 7 Payday loan, title loan, or personal loan                    -0.656
## 8 Student loan                                                 -0.609
## 9 Mortgage                                                     -0.604
```

We also computed the mean negative sentiment score by product category. Results showed that complaints for "checking or savings account" and "debt collection" showed the highest negative sentiments, whereas those for "student loan" and "mortgage" showed lower negative sentiments.

Negative Sentiment by Company

```
company_sentiment <- group_by(complaint_afinn, Company) %>% summarize(score = mean(value, na.rm = TRUE))
com <- arrange(company_sentiment, score)
com
```

```
## # A tibble: 1,250 x 2
##    Company                          score
##    <fct>                            <dbl>
##  1 Affiliates Management Company    -4
##  2 BCG Equities, LLC                -4
##  3 Data Check of America LLC        -4
##  4 Alliance Recovery Group          -3.5
##  5 Student Loan Finance Corporation -3.5
##  6 MAJR Financial Corp              -3.33
##  7 Meade Recovery Services, LLC     -3
##  8 Skrill USA, Inc.                 -3
```

```
##  9 Southern Financial Systems, Inc. -3
## 10 The Collection Connection        -3
## # ... with 1,240 more rows
```

We attained the mean negative sentiment score by company. According to the results, the top three companies with the highest consumer dissatisfaction turned out to be Affiliates Management Company, BCG Equities, and Data Check of America. Insights like this one can suggest which companies are handling consumer experiences better or worse than their competitors.

Topic Modeling with LDA

We subsetted a sample in order to apply LDA. Also, many complaints in the "Narrative" column included "xxxx" which does not have useful meaning, so we removed "xxxx" from the tokenized complaint data frame "tidy_complaint".

```
small_complaint <- tidy_complaint[1:1000, ]

test <- small_complaint$word
edit <- str_replace_all(string = test, pattern = "[xxxx]", replacement = "")
edit <- str_squish(edit)
```

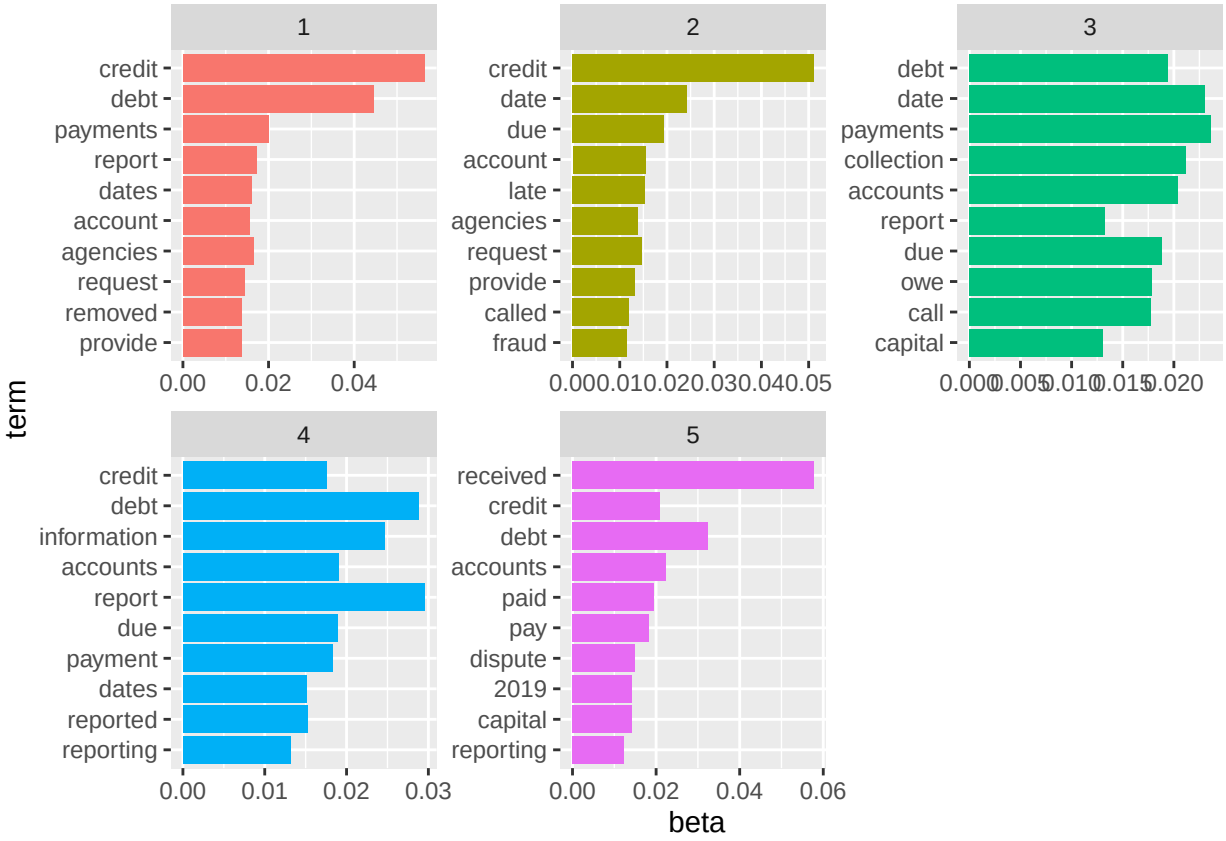
```
Narrative_Corpus <- VCorpus(VectorSource(edit))
text_dm <- DocumentTermMatrix(Narrative_Corpus)
```

```
rowtotal <- apply(text_dm, 1, sum) #Find the sum of words in each Document
dtm_new <- text_dm[rowtotal > 0, ] #remove all docs without words
lda <- LDA(dtm_new, k = 5, control = list(seed = 1234))
topics <- tidy(lda, matrix = "beta")
topics
```

```
## # A tibble: 2,090 x 3
##    topic term          beta
##    <int> <chr>        <dbl>
##  1     1 1692g    0.000934
##  2     2 1692g    0.00230
##  3     3 1692g    0.00154
##  4     4 1692g    0.00144
##  5     5 1692g    0.0000514
##  6     1 19000.00 0.000812
##  7     2 19000.00 0.00128
##  8     3 19000.00 0.000477
##  9     4 19000.00 0.00221
## 10     5 19000.00 0.00148
## # ... with 2,080 more rows
```

In order to conduct topic modeling, we first put the narrative column to be included in the corpus, then we created a document term matrix that has one row for each document (or complaint in this case) in the corpus, one column for each word, and cell for the frequency of words that appear in the document. We then applied the LDA approach to see the probability that each word was generated by one of the 5 topics.

```
top <- topics %>% group_by(topic) %>% top_n(10, beta) %>% ungroup() %>% arrange(topic, -beta)
top %>% mutate(term = reorder(term, beta)) %>% ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) + facet_wrap(~ topic, scales = "free") + coord_flip()
```



We then created a plot to see the top 10 words with the highest probabilities to fall into each of the 5 topics. From the plot above, we were able to link some of the topics to the product categories. For example, in topic 2, top ten words with the highest beta included "credit", "date", "due", and "fraud", suggesting it is potentially related to the "Credit Card" product category. Similarly, topic 3 has "debt" with the highest probability that the word falls into that topic, suggesting that topic 3 is likely related to the product category "Deby collection". Lastly, for topic 4, it includes words such as "reported" and "reporting", which is not so much included in other topics. This can mean that narratives that are included in topic 4 are likely to be related to the product category "Credit reporting, credit repair services, or other personal consumer reports". In this way, we thought that through utilizing the LDA approach, we can classify the complaint narratives into topics, which can then be used to predict which product category the complaint should be assigned to, by looking at the words with high relevance to each category. This led us to think that if a machine learning classifier model can be used to classify narratives into product categories, it can streamline the logistical procedures for handling customer complaints.

Predicting Product Category from Narratives using Machine Learning Classification Models

```
#check the names of products to pick up two products to predict.
unique(complaint$Product)
```

```
## [1] Debt collection
## [2] Credit reporting, credit repair services, or other personal consumer reports
## [3] Checking or savings account
## [4] Vehicle loan or lease
## [5] Mortgage
## [6] Credit card or prepaid card
## [7] Payday loan, title loan, or personal loan
## [8] Student loan
## [9] Money transfer, virtual currency, or money service
## 18 Levels: Bank account or service ... Virtual currency
```

Because there were many types of product categories, for the efficiency of model creation we decided to select two product categories to predict if the narratives fall into any of the two categories. We selected "Checking or savings account" and "Mortgage", and ran models to see if the narratives can be assigned to either one.

```
#create a dataframe for dtm and machinelearning
V_df <- complaint %>% select(Product, Narrative) %>%
  filter(Product == "Checking or savings account" | Product == "Mortgage") %>%
  mutate(Product = ifelse(Product == "Checking or savings account", 1, 0)) %>%
  mutate(Product = factor(Product, levels = 1:0, labels = c("C_S", "M")))
#Product == "Checking or savings account" is 1, named "C_S" and Product == "Mortgage" is 0, named "M"

#make Narrative column as Vector to put it in the VectorSource
V_df_Narrative <- as.vector(V_df$Narrative)
```

In order to run a classification model, we first need to create a new dataframe that includes the narrative columnand the column that shows if the complaint narrative was assigned to either "Checking or Savings account" (labeled as "C_S"), or "Mortgage" (labeled as "M"). After creating a new data frame, we converted the narrative column into vector class in order to apply the VectorSource function.

```
#create a raw corpus for Narrative column using VectorSource
raw_corpus <- Corpus(VectorSource(V_df_Narrative))
```

Using the VectorSource function, we put the Narrative column in the new data frame created above into a corpus, which we will thenmake it into a document term matrix.

```
#to remove xxxx
subSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))

#cleaning the raw_corpus
corpus <- tm_map(raw_corpus, subSpace, "x")
```

```
## Warning in tm_map.SimpleCorpus(raw_corpus, subSpace, "x"): transformation
## drops documents
```

```
corpus <- tm_map(corpus, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(tolower)):
## transformation drops documents
```

```
corpus <- tm_map(corpus, stripWhitespace)
```

```
## Warning in tm_map.SimpleCorpus(corpus, stripWhitespace): transformation
## drops documents
```

```
corpus <- tm_map(corpus, removePunctuation)
```

```
## Warning in tm_map.SimpleCorpus(corpus, removePunctuation): transformation
## drops documents
```

```
corpus <- tm_map(corpus, removeNumbers)
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeNumbers): transformation drops
## documents
```

```
corpus <- tm_map(corpus, removeWords, stopwords("english"))
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords("english")):
## transformation drops documents
```

```
corpus <- tm_map(corpus, stemDocument)
```

```
## Warning in tm_map.SimpleCorpus(corpus, stemDocument): transformation drops
## documents
```

We applied functions above to clean the corpus.

```
#make DocumentTermMatrix with the cleaned corpus
dtm <- DocumentTermMatrix(corpus)
```

After putting into corpus, the vectorized narrative column was turned into a document term matrix.

```
#create a dataframe with Product column and dtm
#make the DocumentTermMatrix as dataframe --> this only contains Narrative information as dtm.
mydtm <- as.data.frame(as.matrix(dtm))
#select Product information --> this is already in the dataframe format.
myproduct <- select(V_df, Product)
#combine these two by cbind --> we don't need to use merge function because these two are in the same o
mydf <- cbind(myproduct, mydtm)

mydf[1:10, 1:9]
```

```
##    Product account advertis bonus check come cuse day deposit
## 1      C_S       1        1     1     1    1    2   1       1
## 2        M       0        0     0     0    0    0   0       0
## 3        M       0        0     0     0    0    0   0       0
## 4        M       0        0     0     0    0    0   0       1
## 5        M       0        0     0     0    1    0   5       0
## 6        M       0        0     0     0    0    0   3       0
## 7        M       0        0     0     0    0    0   6       0
## 8      C_S       1        0     0     0    0    0   0       0
## 9        M       0        0     0     0    0    0   2       0
## 10       M       0        0     0     0    0    0   0       0
```

As our final data frame to run classification models, we created a data frame with the document term matrix and the product column. We converted the dtm into a data frame format, and combined it with the data frame with the product category information. Our final data frame ("mydf") includes the row that indicates each narraitve, the "Product" column that indicates which product category the narrative was assigned to, other columns each with words and the cells as the frequency that each word appeared in the narrative.

```
#Start Classification

#mydf$Product is already factor
set.seed(20191220)
intrain <- createDataPartition(y = mydf$Product, p = 0.8, list = FALSE)
training <- mydf[intrain, ]
testing <- mydf[-intrain, ]
```

We conducted train-test-split prior to running to classification model. For our classification, we selected the following models: 1. Linear Probability Model 2. Logistic Regression 3. Penalized Logistic Regression 4. Discriminant Analysis (LDA) 5. QDA

In our classification model, our independent variable is the frequency of words "account", "deposit", and "direct" that appear in each narrative, and our dependent variable is a binary classification of whether the narrative gets assigned to "C_S" (checking or savings account). The three words that seems to be highly related to the checking and savings account issue were chosen.

Linear Probability Model

```
#Binary Classification via a Linear Probability Model
ols <- lm(Product == "C_S" ~ account + deposit + direct, data = training)

y_hat_ols <- predict(ols, newdata = testing)

Y_yesno <- factor(y_hat_ols > 0.5, levels = c(TRUE, FALSE), labels = c("C_S", "M"))

table(Y_yesno, testing$Product)
```

```
##
## Y_yesno C_S   M
##     C_S 122  37
##     M   165 353
```

#calculated accuracy based on the table Accuracy = (122+353)/(122+37+165+353) [1] 0.7016248

Based on the table above, the accuracy score of the LPM model is 0.70.

```
#Binary classification via logistic regression
logit <- glm(Product == "C_S" ~ account + deposit + direct, data = training, family = binomial(link = "
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
#--> generalized linear model
y_hat_logit <- predict(logit, newdata = testing, type = "response") # these are probabilities
# these are classifications
z_logit <- factor(y_hat_logit > 0.5, levels = c(TRUE, FALSE), labels = c("C_S", "M"))
table(z_logit, testing$Product)
```

```
##
## z_logit C_S   M
##     C_S 163  55
##     M   124 335
```

Accuracy = (163+335)/(163+55+124+335) [1] 0.7355982

Based on the table above, the accuracy score of the logistic regression model is 0.735.

```
#get a penalized version
ctrl <- trainControl(method = "repeatedcv", repeats = 3,
                     classProbs = TRUE, summaryFunction = twoClassSummary)

tune_grid <- expand.grid(.alpha = seq(0, 1, length.out = 10),
                         .lambda = seq(0, 1, length.out = 10))

penalized_logit <- train(Product ~ account + deposit + direct, data = training, method = "glmnet",
                         trControl = ctrl, metric = "ROC", tuneGrid = tune_grid,
                         preProcess = c("center", "scale"))

#y_hat_penalized_logit <- predict(penalized_logit, newdata = testing, type = "prob")$yes
# above are probabilities, below are classifications
z <- predict(penalized_logit, newdata = testing)
defaultSummary(data.frame(obs = testing$Product, pred = z))
```

```
##  Accuracy     Kappa
## 0.7178730 0.3918771
```

```
#LDA version
LDA <- train(Product ~ account + deposit + direct,
             data = training, method = "lda", preProcess = c("center", "scale"))
confusionMatrix(predict(LDA, newdata = testing), reference = testing$Product)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction C_S   M
##        C_S 124  39
##        M   163 351
##
##                Accuracy : 0.7016
##                  95% CI : (0.6656, 0.7359)
##     No Information Rate : 0.5761
##     P-Value [Acc > NIR] : 1.072e-11
##
##                   Kappa : 0.3521
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4321
##             Specificity : 0.9000
##          Pos Pred Value : 0.7607
##          Neg Pred Value : 0.6829
```

```
##              Prevalence : 0.4239
##          Detection Rate : 0.1832
##    Detection Prevalence : 0.2408
##       Balanced Accuracy : 0.6660
##
##        'Positive' Class : C_S
##
```

```
#QDA Version
QDA <- train(Product ~ account + deposit + direct,
             data = training, method = "qda", preProcess = c("center", "scale"))
confusionMatrix(predict(QDA, newdata = testing), reference = testing$Product)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction C_S   M
##        C_S 124  33
##        M   163 357
##
##                Accuracy : 0.7105
##                  95% CI : (0.6747, 0.7444)
##     No Information Rate : 0.5761
##     P-Value [Acc > NIR] : 3.361e-13
##
##                   Kappa : 0.3695
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.4321
##             Specificity : 0.9154
##          Pos Pred Value : 0.7898
##          Neg Pred Value : 0.6865
##              Prevalence : 0.4239
##          Detection Rate : 0.1832
##    Detection Prevalence : 0.2319
##       Balanced Accuracy : 0.6737
##
##        'Positive' Class : C_S
##
```

Among all the classification models, the unpenalized logistic regression model had the highest accuracy score of 0.735.

Conclusion

Through this project, we were able to drive some useful insights that IBM can leverage to improve their customer experiences management. First, through text analysis and sentiment analysis, we were able to assign negative sentiment scores for each complaint, and from that information attain the mean negative sentiment score by state, product category and company. We learned that some states, like North Dakota, showed higher negative sentiments in their complaints compared to other states. We wanted to explore further if such differences in the intensity of negative sentiments can be attributed to other factors such as

the income level of the state. However, it turned out that there was no significant pattern between state income level and the negative sentiment score. But the limitation of this approach is that we did not have the income level data from the IBM data set, but instead we attained from other sources, so the information may not be accurate (This is because not everyone in the state is submitting the complaints to IBM). With the sentiment analysis information, we attained useful insights about which product category, company or states are demonstrating higher levels of customer dissatisfaction. We also utilized the topic modeling approach to see if the narratives can be assigned to 5 different topics. Based on the top words that fall into each topic with high probabilities, we were able to make assumptions about how each topic can symbolize different product categories. Based on the different words that are mentioned frequently in the complaints, we were trying to see if the narratives can be automatically assigned to the corresponding product categories. From this idea, we decided to run classification models to see if we can predict product category based on the text analysis of the narratives. We ran different classification models, which all returned a decently high accuracy score of around 0.70. For the efficiency of creating models and computation, we selected only two product categories as the dependent variables, and the frequency of three words as independent variables. Howevever, if we had the opportunity to further explore on this topic, we would like to include other variables or try with other classification models such as random forests to see if we can improve the accuracy score.