Analysis and Prediction of Consumer Complaints

A PROJECT REPORT

Submitted by,

Hrishikesh Gadkari	-377116140
Mitanshu Patel	-651884702
Subhiksha Ranganathan	-610521275
Aayush Desai	-277968752
Jeet Patel	-856777509
Ketki Naik	-953706796

CIS 600 Principles of Social Media and Data Mining

Spring 2020

Syracuse University,
Syracuse,
New York - 13244

TABLE OF CONTENTS

1. Problem Definition

1.1 Project Overview

- ➔ In 2011, The Consumer Financial Protection Bureau (CFPB) was created by congress, to entertain and process various consumer complaints related to financial services. CFPB works by submitting a complaint to them, the complaint is then reviewed and routed to the company. The company then reviews the received complaint, contacts the person with the necessary steps taken or that will be taken. The action taken will be completely done in the period of 15 - 60 days. After all these are validated, the complaint is added to the database.

- ➔ Insights regarding the problems that are faced by consumers is provided in every complaint narrative. As there's a significant increase in CFPB complaint narratives, manually handling and reviewing these documents is not feasible. So, there's a need for an intelligent system which automatically analyzes the narratives and provides the experts with insightful knowledge. Therefore, this project provides an intelligent approach to analyze the CFPB complaint narrative data.

- ➔ The goal of our project is to classify consumer complaints into predefined categories based on its content using Machine Learning Algorithms to get accurate predictions. This project also performs a sentiment analysis on the complaint narratives to classify the content as positive, negative or neutral. This project helps in improving customer experience by providing an efficient as well as effective investigations on the narratives. Thus, empowering consumers to make informed financial decisions.

1.2 Problem Statement

"Analysis of Consumer Complaints and classifying them into predefined categories by using various NLP algorithms."

Given a Consumer Complaint dataset, which consists of a million rows and 18 columns, we take the main use of the dataset, the financial product and the complaint narrative into account. We take all the financial products and the corresponding complaint narrative, and apply different classification models on them to classify them into different topics.

We have 18 different categories/ classes in this dataset we are considering. There is a difference between the traditional and very famous multi-class classification, and the one which we will be

using, which is the multi-label classification. In a multi-class classification, each instance is classified into one of three or more classes, whereas, in a multi-label classification, multiple labels are to be predicted for the same instance.

However, it is observed that some classes are contained in others. For instance, ‘Credit card’ and ‘Prepaid card’ are contained in the ‘Credit card or prepaid card’ category. Now, imagine there is a new complaint about a Credit card and we want to classify it. The algorithm can either classify this complaint as 'Credit card' or 'Credit card or prepaid' and it would be correct. Nevertheless, this would affect model performance. In order to avoid this problem, the names of some categories are renamed and reduced to 13 classes to avoid anomalies.

After this step, we apply Multi-Classification models such as Random Forest, Linear Support Vector Machine, Multinomial Naive Bayes, and Logistic Regression. We select the best model based on accuracy and take that into account. When we finalize with building this model, we test this model to check if the prediction works right by categorizing the proposed to the appropriate class.

1.3 Evaluation Metrics

Accuracy: We apply 4 different classification models, Random Forest, Linear Support Vector Machine, Multinomial Naive Bayes, and Logistic Regression to the dataset, and find the mean accuracy of each of the models to find which one fits well. Based on the results, the models are evaluated and the best, more accurate one is taken into account.

Precision: It is a valid choice of evaluation metric to be very sure of our prediction. After applying the model with the highest accuracy, we calculate the precision on all the categorized classes to see, if predicted, which classes will get the correct values, to see which predicted positives are truly positives.

F-1 score: It is possible to observe that the classes with more support (number of occurrences) tend to have a better f1-score. This is because the algorithm was trained with more data.

The best mean accuracy was obtained with LinearSVC with an accuracy rate of 0.779 making it higher than the three other models. We have obtained that the classes with more precision are ‘Mortgage’, ‘Credit reporting, repair, or other’, and ‘Student loan’.

2. Analysis:

2.1 Data Extraction:

The most important and time consuming task in any machine learning project or data science is to extract a good quality of data for further processing and storage. First step includes loading data and importing packages. CFPB consumer complaint database is available publicly on the CFPB website. Data set includes real world complaints received about financial products and services. The dataset consists of a total of 615,273 complaint records. These records have multiple columns which include the date of submission, product and its issues, the company that was sent the response about the complaint, etc. Each complaint has been labeled with a specific product.

Important packages used :

pandas - It offers data structures and operations for manipulating numerical tables and time series.

numpy - Support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

matplotlib - It's a multi platform data visualization library used for 2D plots of arrays.

Sklearn - Machine learning library that includes various classification, regression and clustering algorithms.

Steps for loading the data and importing packages :

```
import pandas as pd
import numpy as np
from scipy.stats import randint
from io import StringIO
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import chi2
from IPython.display import display
from sklearn.model_selection import train_test_split
import seaborn as sns # used for plot interactive graph.
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

```
# Data loading from csv
dataframe = pd.read_csv('consumercomplaints.csv')
dataframe.shape
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718:
  interactivity=interactivity, compiler=compiler, result=result)
(1282355, 18)
```

In the dataset, we have more than 1 million instances (rows) and 18 features (columns).

2.2 Data Exploration

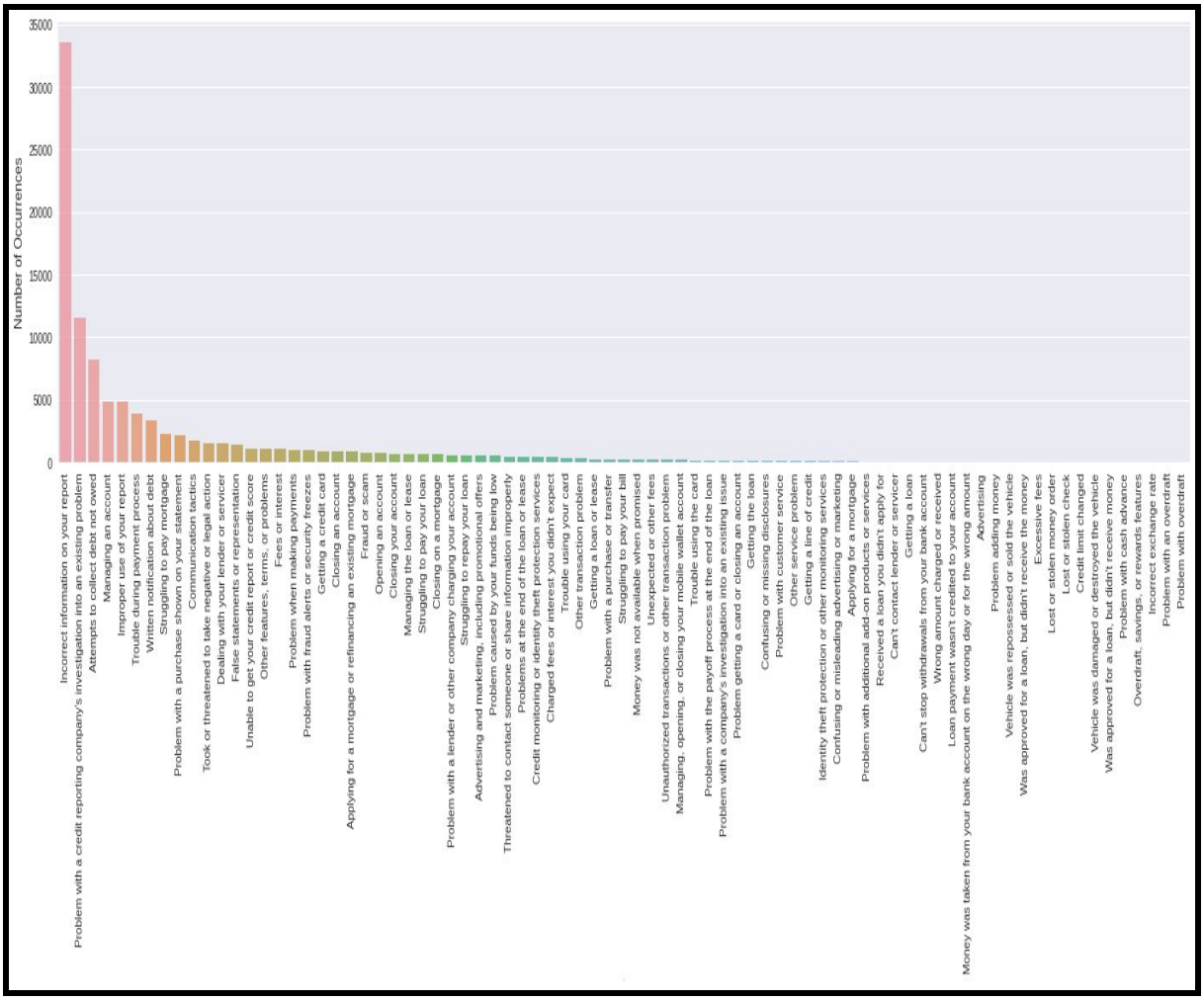
The data set consists of the following fields :

Out of these fields, the consumer complaint narrative field will be preprocessed and fitted into different classifiers to predict which product category the complaint belongs to.

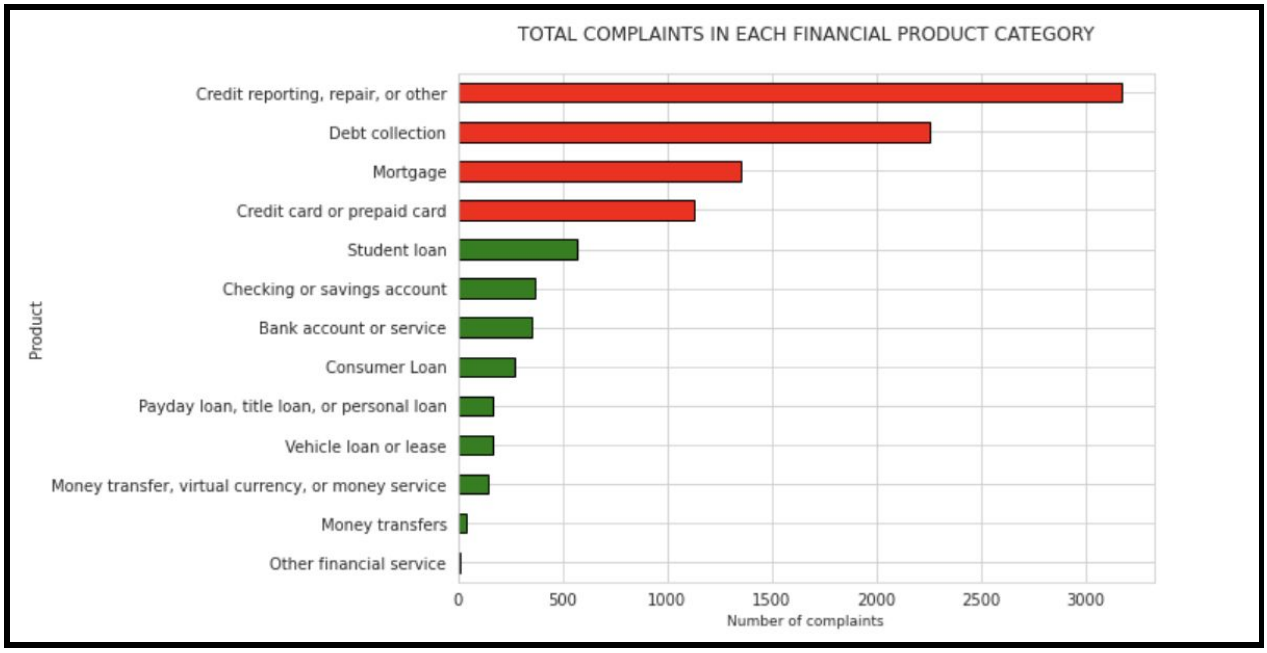
We have a total of more than 1 million rows and 18 columns, which can be loaded from consumercomplaints.csv file.

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	Consumer consent provided?	Submitted via	Date sent to company	Company response to consumer	Timely response?	Consumer disputed?	Complaint ID
0	05/10/2019	Checking or savings account	Checking account	Managing an account	Problem using a debit or ATM card	NaN	NaN	NAVY FEDERAL CREDIT UNION	FL	328XX	Older American	NaN	Web	05/10/2019	In progress	Yes	NaN	3238275
1	05/10/2019	Checking or savings account	Other banking product or service	Managing an account	Deposits and withdrawals	NaN	NaN	BOEING EMPLOYEES CREDIT UNION	WA	98204	NaN	NaN	Referral	05/10/2019	Closed with explanation	Yes	NaN	3238228
2	05/10/2019	Debt collection	Payday loan debt	Communication tactics	Frequent or repeated calls	NaN	NaN	CURO Intermediate Holdings	TX	751XX	NaN	NaN	Web	05/10/2019	Closed with explanation	Yes	NaN	3237964
3	05/10/2019	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Old information reappears or never goes away	NaN	NaN	Ad Astra Recovery Services Inc	LA	708XX	NaN	NaN	Web	05/10/2019	Closed with explanation	Yes	NaN	3238479
4	05/10/2019	Checking or savings account	Checking account	Managing an account	Banking errors	NaN	NaN	ALLY FINANCIAL INC.	AZ	85205	NaN	NaN	Postal mail	05/10/2019	In progress	Yes	NaN	3238460

2.3 Exploratory Visualization



The above plot between various issues mentioned in the dataset and their number of occurrences. We can observe from the plot that 'Incorrect information on your report' has the maximum number of occurrences followed by the 'Problem with a credit reporting company's investigation into an existing problem'.



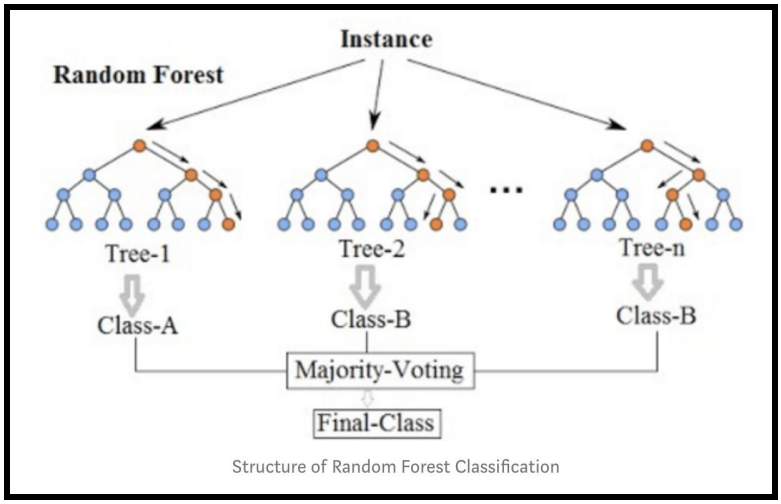
The above plot is about the number of complaints in each product category. It is between the products and the number of occurrences. Here, we can observe that 'Credit reporting, repair or other', 'Debt Collection', 'Mortgage' and 'Credit card or prepaid card' are the Products which have a majority of occurrences in the CFPB consumer complaint dataset.

2.4 Algorithms & Techniques

1. Random Forest
2. Linear Support Vector Machine
3. Multinomial Naive Bayes
4. Logistic Regression

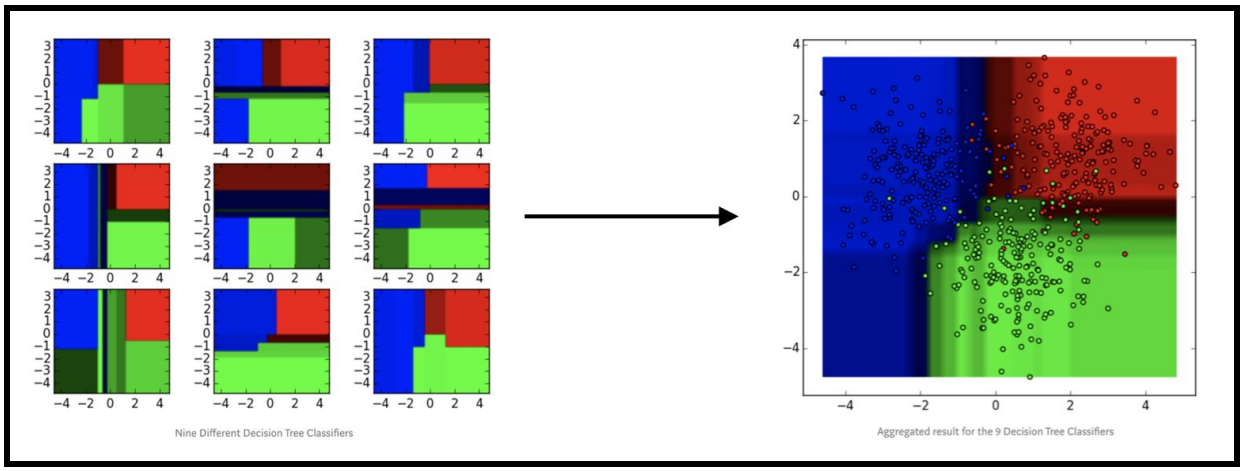
1. **Random Forest** : It is an **ensemble tree-based learning** algorithm. The Random Forest Classifier is a set of decision trees from a randomly selected subset of training set. It **aggregates the votes from different decision trees** to decide the final class of the test object. Ensemble algorithms are those which **c**ombine more than one algorithms of the same or different kind for classifying objects.

Structure of the algorithm :



Structure of Random Forest Classification

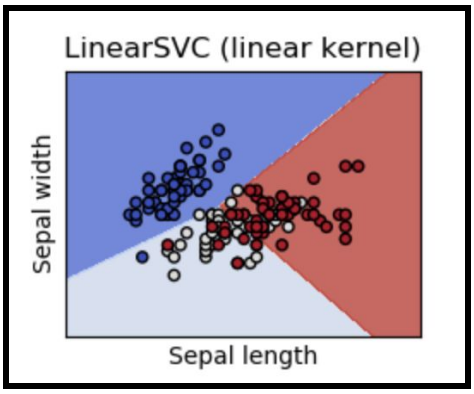
The random forest is composed of the generated trees, and we use the random forest to classify the electrical signal test data. The classification result is decided by the decision trees. There needs to be some actual signal in our features so that models built using those features do better than random guessing. The predictions (and therefore the errors) made by the individual trees need to have low correlations with each other.


Nine Different Decision Tree Classifiers

Aggregated result for the 9 Decision Tree Classifiers

2. Linear Support Vector Machine :

Linear SVC is another implementation of Support Vector Classification. The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide, returning a "best fit" hyperplane that divides, or categorizes, your data. From there, after getting the hyperplane, you can then feed some features to your classifier to see what the "predicted" class is.



Linear SVC takes 2 arrays as input. an array X of size [n_samples, n_features] holding the training samples, and an array Y of class labels (strings or integers), size [n_samples].

3. **Multinomial Naive Bayes :**

Multinomial implements the naive Bayes algorithm for multinomial distributed data, and is one of the two classic naive Bayes variants used in text classification (where the data are typically represented as word vector counts, although tf-idf vectors are also known to work well in practice). The distribution is parametrized by vectors $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ for each class **y,** where **n** is the number of features (in text classification, the size of the vocabulary) and θ_{yi} is the probability $P(x_i \mid y)$ of features appearing in a sample belonging to class **y**.

The parameter θ_y is estimated by a smoothed version of maximum likelihood, i.e. relative frequency counting :

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n}$$

Where $N_{yi} = \sum_{x \in T} x_i$ is the number of times feature i appears in a sample of class y in the training set T and $N_y = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class **y**.

4. **Logistic Regression :**

Logical Regression algorithm is used for assigning observations to the discrete set classes of classes. It transforms its output to probability value using a logistic sigmoid function which can be mapped to more than one discrete class.
Types of Logical Regression:

- **Binary**
  Result is predicted in two values i.e. pass or fail and using sigmoid function predicted values are mapped to the probabilities. Equation for that is as follows:
  $$S(z) = \frac{1}{1 + e^{-z}}$$
  Where, S(z) = Output between 0 and 1
  Z = Input to function

- **Multi**
  Binary Logistic regression is rerun for multiple time , one for each class to achieve Multi Logistic Regression

- **Ordinal**
  Ordinal Logistic Regression is used to predict the ordinal dependent variable based on given one or more independent variables. It is the generalization of Multiple Linear Regression. It uses interaction between the independent variables for predictions of dependent variables.

Logical Regression models the probability of default or first class i.e. the probability that input (X) belongs to default class(Y)

- P(X) = P(Y=1 | X)

Data for Logical Regression is prepared using parameters like Binary Output Variable, Remove Noise, Gaussian Distribution, Removing Correlated inputs and fail to converge.

2.5 Benchmark Model

We are using four different models which are LinearSVC, Logistic Regression, MultinomialNB and Random Forest Classifier with 5-fold cross validation, we came to a conclusion that LinearSVC has better mean accuracy compared to others and a better option. Therefore, we will be using LinearSVC for our classifications of various consumer complaints.

Logistic Regression or MultinomialNB may also work in this case but since we are using such a huge dataset and our results require precision, using LinearSVC will be beneficial.A detailed comparison between the algorithms with their mean accuracy and standard deviation will be provided.

3. Methodology

3.1 Data pre-processing

The CFPB consumer complaint dataset has over 1 million rows but this data has a lot of irrelevant and noisy data which needs to be removed. For this model, the 'Product' and 'Consumer complaint Narrative' columns are important and will be required for analysis. Hence, we created a new dataframe which consists of only these two columns.

```
# Creating a new dataframe with only the two features we need
dataframe1 = dataframe[['Product', 'Consumer complaint narrative']].copy()

# NaN values are removed (null)
dataframe1 = dataframe1[pd.notnull(dataframe1['Consumer complaint narrative'])]

# Consume complaint narrative is renamed as Complaint
dataframe1.columns = ['Product', 'Complaint']

dataframe1.shape
```

```
(438660, 2)
```

```
# Percentage of actual complaints (not null)
total = dataframe1['Complaint'].notnull().sum()
round((total/len(dataframe)*100),1)
```

```
31.7
```

As said earlier, the dataset is noisy and has a lot of redundant data, we removed Null values (NaN) from the dataset and that helped in reducing the size of the dataset. Now, the data which we have obtained is of size 438660 rows and 2 columns. In this data, approximately 32% comprises text which is still a good amount of data required for analysis. Next, we need to distinguish different Products from the given datasets that will be required for classification of various consumer complaints.

```
pd.DataFrame(dataframe.Product.unique()).values
```

```
array([['Debt collection'],
       ['Payday loan, title loan, or personal loan'],
       ['Credit reporting, credit repair services, or other personal consumer reports'],
       ['Mortgage'],
       ['Money transfer, virtual currency, or money service'],
       ['Credit card or prepaid card'],
       ['Vehicle loan or lease'],
       ['Checking or savings account'],
       ['Student loan'],
       ['Credit reporting'],
       ['Credit card'],
       ['Bank account or service'],
       ['Other financial service'],
       ['Consumer Loan'],
       ['Payday loan'],
       ['Prepaid card'],
       ['Money transfers'],
       ['Virtual currency']], dtype=object)
```

There are a total of 18 different categories of Product but after close observation we get to know that there is still some redundancy which can be removed. Some categories were found out to be contained into other categories. So, for a new complaint which we want to classify may have multiple options in which it can classify. Since, these multiple options resemble the same categories, our classification will be correct but it will hamper the performance of the model.

```
# Renaming categories
df2.replace({'Product':
             {'Credit reporting, credit repair services, or other personal consumer reports':
              'Credit reporting, repair, or other',
              'Credit reporting': 'Credit reporting, repair, or other',
             'Credit card': 'Credit card or prepaid card',
             'Prepaid card': 'Credit card or prepaid card',
             'Payday loan': 'Payday loan, title loan, or personal loan',
             'Money transfer': 'Money transfer, virtual currency, or money service',
             'Virtual currency': 'Money transfer, virtual currency, or money service'}},
            inplace= True)
```

```
pd.DataFrame(df2.Product.unique())
```

0	Debt collection
1	Student loan
2	Checking or savings account
3	Consumer Loan
4	Mortgage
5	Credit card or prepaid card
6	Credit reporting, repair, or other
7	Vehicle loan or lease
8	Money transfer, virtual currency, or money ser...
9	Bank account or service
10	Payday loan, title loan, or personal loan
11	Money transfers
12	Other financial service

So, in order to avoid this problem we renamed the similar categories and merged them. Due to which, the number of categories were reduced from 18 to 13 and we labelled these categories with numbers so that our predictive model understands different categories better. We renamed the labels with numbers as 'category_id'.

```
# A new column 'CategoryId' is created to maintain categories
df2['CategoryId'] = df2['Product'].factorize()[0]
category_id_df = df2[['Product', 'CategoryId']].drop_duplicates()
#del df2['CategoryId']

# Dictionaries to make search easier later
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['CategoryId', 'Product']].values)

# Resulted dataframe with categories
df2.head(10)
```

	Product	Complaint	CategoryId
88396	Debt collection	on XX/XX/19 commenity capital bank called a re...	0
672838	Student loan	I co signed XXXX loans for my son years ago wi...	1
450389	Checking or savings account	I was involved in an online relationship with ...	2
649714	Consumer Loan	I purchased a XXXX XXXX XXXX truck from XXXX X...	3
907024	Debt collection	A man called from a company called RMB - which...	0
408205	Debt collection	" Credence Resource Management " has called m...	0
1000587	Debt collection	I am victim of fraud and someone stole my iden...	0
72085	Debt collection	I received a notice from The Advantage Group, ...	0
18312	Mortgage	Account # XXXXThrough Freedom MortgageMy accou...	4
255813	Credit card or prepaid card	On XX/XX/2018 I used my credit card to rent a ...	5

For predictions the text which we have right now needs to be converted into vectors. In this case, we used Term Frequency - Inverse Document Frequency (TFIDF) weights. Using this we can evaluate the importance of a word in a document which is in a collection of documents. But first, we will have to remove punctuation from the words and also lower case the words. Only after performing these operations, we are able to find the importance of words with its frequency. Basically, TF-IDF gets the scores for the weights by multiplying Term Frequency (TF) with Inverse Document Frequency (IDF). Term Frequency summarizes how often a particular word appears in a document. Term Frequency is calculated as (Number of times the term appears in the document) / (Total number of words in the document). Inverse Document Frequency downscales the words that appear frequently across the document. A IDF score is obtained when a particular term appears in a few documents. Also, if a term is very common in a document then it will have a low IDF score. IDF is calculated as ln(Number of Docs / Number docs the term appears in). So, after calculating the TF-IDF scores we get the words which are more interesting. The higher the TF-IDF score, the rarer the term is. We are using TfidfVectorizer which is predefined in python.

```
tfidf_score = TfidfVectorizer(sublinear_tf=True, min_df=5,ngram_range=(1, 2), stop_words='english')

# Transforming complaint into a vector
features = tfidf_score.fit_transform(df2.Complaint).toarray()

labels = df2.CategoryId

print("Every sample (%d) complaints is represented by %d features (TF-IDF score of unigrams and bigrams)" %(features.shape))
```

```
Every sample (10000) complaints is represented by 27507 features (TF-IDF score of unigrams and bigrams)
```

It has different parameters which allows us to remove words from the vocabulary that have occurred less than a specified number (min_df), remove the words from the vocabulary which have occurred more than a specified number (max_df), scaling the term frequency in logarithmic scale, removing the predefined stopwords in English and to consider unigrams and bigrams. So, with this we can find the most correlated unigrams and bigrams with each of the product categories.

```
# Three most correlated terms in each of the product categories
N = 3
for Product, CategoryId in sorted(category_to_id.items()):
  features_chi2 = chi2(features, labels == CategoryId)
  indices = np.argsort(features_chi2[0])
  feature_names = np.array(tfidf_score.get_feature_names())[indices]

  unigrams = [v for v in feature_names if len(v.split(' ')) == 1]
  bigrams = [v for v in feature_names if len(v.split(' ')) == 2]

  print("\n==> %s:" %(Product))
  print("  # Top Correlated Unigrams are: %s" %(', '.join(unigrams[-N:])))
  print("  # Top Correlated Bigrams are: %s" %(', '.join(bigrams[-N:])))
```

To find the three most correlated terms with each of the product categories, we get the most correlated unigrams and bigrams of the product categories as shown below:

```
==> Bank account or service:
  # Top Correlated Unigrams are: overdraft, bank, scottrade
  # Top Correlated Bigrams are: citigold checking, debit card, checking account

==> Checking or savings account:
  # Top Correlated Unigrams are: checking, branch, overdraft
  # Top Correlated Bigrams are: 00 bonus, overdraft fees, checking account

==> Consumer Loan:
  # Top Correlated Unigrams are: dealership, vehicle, car
  # Top Correlated Bigrams are: car loan, vehicle loan, regional acceptance

==> Credit card or prepaid card:
  # Top Correlated Unigrams are: express, citi, card
  # Top Correlated Bigrams are: balance transfer, american express, credit card

==> Credit reporting, repair, or other:
  # Top Correlated Unigrams are: report, experian, equifax
  # Top Correlated Bigrams are: credit file, equifax xxxx, credit report

==> Debt collection:
  # Top Correlated Unigrams are: collect, collection, debt
  # Top Correlated Bigrams are: debt collector, collect debt, collection agency

==> Money transfer, virtual currency, or money service:
  # Top Correlated Unigrams are: ethereum, bitcoin, coinbase
  # Top Correlated Bigrams are: account coinbase, coinbase xxxx, coinbase account

==> Money transfers:
  # Top Correlated Unigrams are: paypal, moneygram, gram
  # Top Correlated Bigrams are: sending money, western union, money gram

==> Mortgage:
  # Top Correlated Unigrams are: escrow, modification, mortgage
  # Top Correlated Bigrams are: short sale, mortgage company, loan modification

==> Other financial service:
  # Top Correlated Unigrams are: meetings, productive, vast
  # Top Correlated Bigrams are: insurance check, check payable, face face

==> Payday loan, title loan, or personal loan:
  # Top Correlated Unigrams are: astra, ace, payday
  # Top Correlated Bigrams are: 00 loan, applied payday, payday loan

==> Student loan:
  # Top Correlated Unigrams are: student, loans, navient
  # Top Correlated Bigrams are: income based, student loan, student loans

==> Vehicle loan or lease:
  # Top Correlated Unigrams are: honda, car, vehicle
  # Top Correlated Bigrams are: used vehicle, total loss, honda financial
```

With the help of the highly correlated unigrams and bigrams that are extracted from the consumer complaints, we can classify a new complaint into its highly correlated category.

3.2 Implementation

In our data, the features (X) are the consumer complaints and the target (Y) are the Products which we want to predict. We have splitted the data into training which consists of 75% and the rest for testing which consist of 25%.

```
X = df2['Complaint'] # Collection of documents
y = df2['Product'] # Target or the labels we want to predict (i.e., the 13 different complaints of products)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state = 0)
```

We start to implement by first trying out different models and comparing their accuracies. Also, we are performing a 5 fold cross validation on these models.

```
models = [
    RandomForestClassifier(n_estimators=100, max_depth=5, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]

# 5 Cross-validation
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))

entries = []
for model in models:
  Model_Name = model.__class__.__name__
  accuracies = cross_val_score(model, features, labels, scoring='accuracy', cv=CV)
  for fold_idx, accuracy in enumerate(accuracies):
    entries.append((Model_Name, fold_idx, accuracy))

cv_df = pd.DataFrame(entries, columns=['Model_Name', 'fold_idx', 'accuracy'])
```

Sentiment Analysis:

Sentiment Analysis is the process of statistically working out whether a given text is positive, neutral or negative. It does this job in two forms: Polarity-based, in which the pieces of texts are classified as either negative or positive or Valence-based, where the words ‘bad’ and ‘worst’ will

treated different, ‘worst’ will have more impact as compared with ‘bad’ but in polarity-based approach ‘bad’ and ‘worst’ will be treated as the same.
For our project, We have performed Sentiment Analysis using
. VADER is the type of sentiment analysis that works on lexicons of sentiment-related words. VADER gives sentiment metrics from the word ratings. So, we can measure the degree of the text to see which ones are more negative. VADER works with these four features: Emoticons, Capitalization, Degree Modifiers, Shift in polarity due to but. Emoticons like “!” have value while calculating the metrics. Capitalization also plays a good role in calculating the metrics, like “BAD” will have a higher score than “bad”. Degrees of words are also taken into account as well as changes in polarity because of ‘but’ while calculating the metrics score. Due to all such features VADER makes a great option for working with consumer complaints dataset.
Following is the implementation of VADER for Sentiment Analysis:

```
nltk.download('vader_lexicon')

# SentimentIntensityAnalyser object is loaded
from nltk.sentiment.vader import SentimentIntensityAnalyzer

analyzer = SentimentIntensityAnalyzer()
```

We defined the following functions to get the negative, neutral and compound scores from the given texts.

```
# function to get the negative score of a complaint
def negative_score(text):
    neg_value = analyzer.polarity_scores(text)['neg']
    return neg_value

# function to get the neutral score of a complaint
def neutral_score(text):
    neu_value = analyzer.polarity_scores(text)['neu']
    return neu_value

# function to get the compound score of a complaint
def compound_score(text):
    comp_value = analyzer.polarity_scores(text)['compound']
    return comp_value
```

```
dataframe1['negative_sentiment'] = dataframe1['Complaint'].apply(negative_score)
dataframe1['neutral_sentiment'] = dataframe1['Complaint'].apply(neutral_score)
dataframe1['compound_sentiment'] = dataframe1['Complaint'].apply(compound_score)
```

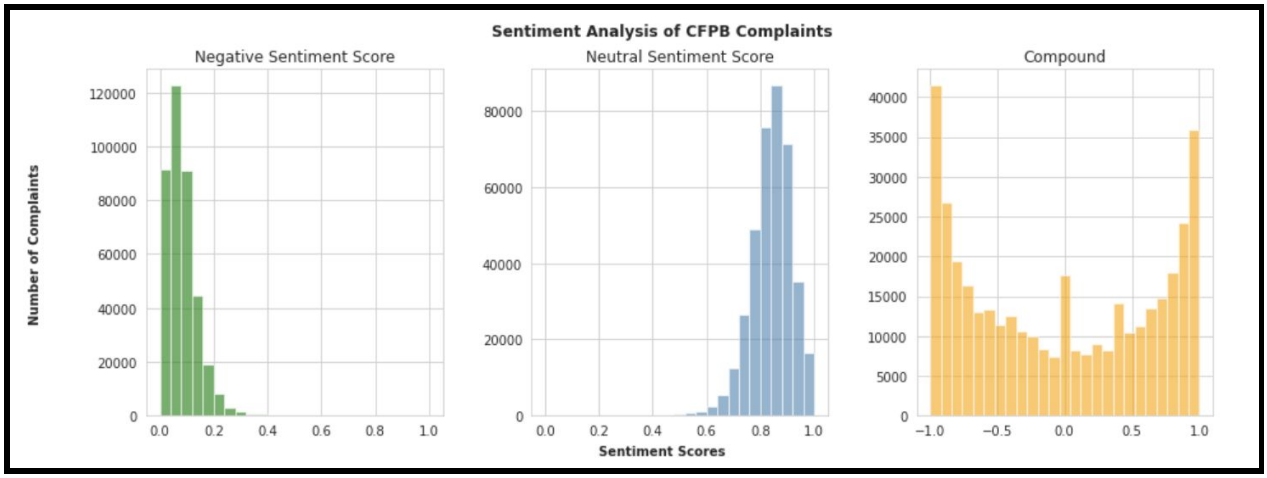
Below the plot for number of complaints vs the sentiment scores is shown for negative, neutral and compound scores.

```
fig, axes = plt.subplots(1, 3, figsize=(15,5))

# plotting all histograms for positive, negative, neutral, compound
dataframe1.hist('neg_sentiment', bins=25, ax=axes[0], color='green', alpha=0.6)
axes[0].set_title('Negative Sentiment Score')
dataframe1.hist('neu_sentiment', bins=25, ax=axes[1], color='steelblue', alpha=0.6)
axes[1].set_title('Neutral Sentiment Score')
dataframe1.hist('comp_sentiment', bins=25, ax=axes[2], color='orange', alpha=0.6)
axes[2].set_title('Compound')

# plot common x- and y-label
fig.text(0.5, 0.04, 'Sentiment Scores',  fontweight='bold', ha='center')
fig.text(0.04, 0.5, 'Number of Complaints', fontweight='bold', va='center', rotation='vertical')

# plot title
plt.suptitle('Sentiment Analysis of CFPB Complaints\n\n', fontsize=12, fontweight='bold');
```

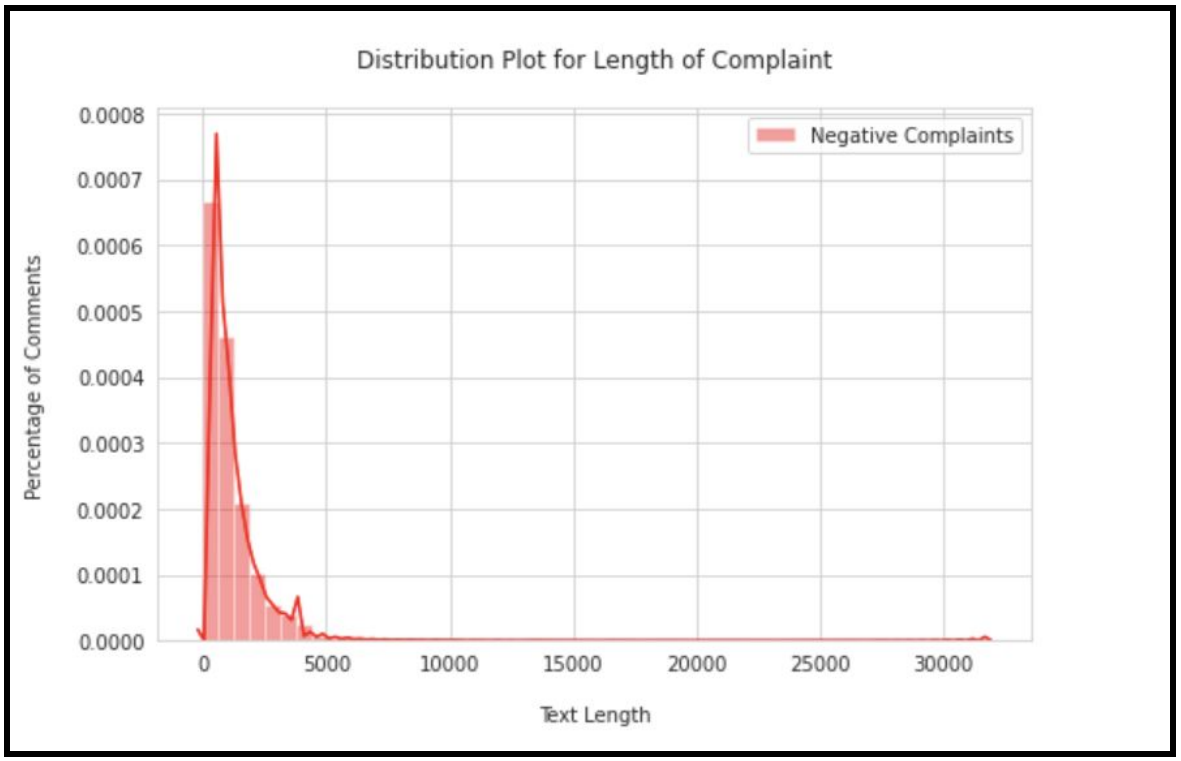


To analyze the negative complaints based on the text length we have plotted the following. Mostly, the complaints with length between 0 and 5000 have a maximum percentage of comments.

```
# all NEGATIVE comments in the dataframe
df_neg = dataframe1.loc[dataframe1.comp_sentiment < 0.0]

# only corpus of NEGATIVE comments
neg_comments = df_neg['Complaint'].tolist()
```

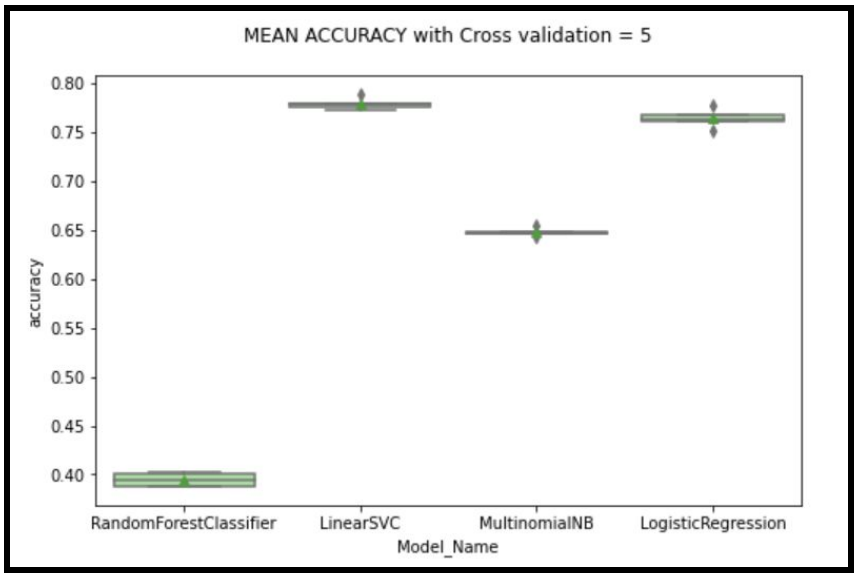
```
df_neg['text_length'] = df_neg['Complaint'].apply(len)
```



3.3 Refinement :

After comparing different models which are LinearSVC, Logistic Regression, MultinomialNB and Random Forest Classifier with 5-fold cross validation, we found that LinearSVC has better mean accuracy compared to others and a better option. Therefore, we will be using LinearSVC for our classifications of various consumer complaints. Comparison of different models is shown in the figure below.

Model_Name	Mean Accuracy	Standard deviation
LinearSVC	0.7807	0.004791
LogisticRegression	0.7633	0.005574
MultinomialNB	0.6385	0.007616
RandomForestClassifier	0.3813	0.002775



4. Results

4.1 Model Evaluation

For model evaluation, we can generate a classification report which consists of precision scores, recall, F1-scores and support for different categories. With the help of the following code we can generate the classification report:

```
X_train, X_test, y_train, y_test,indices_train,indices_test = train_test_split(features,
                                                              labels,
                                                              df2.index, test_size=0.25,
                                                              random_state=1)
model = LinearSVC()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

The output for the above code is as follows:

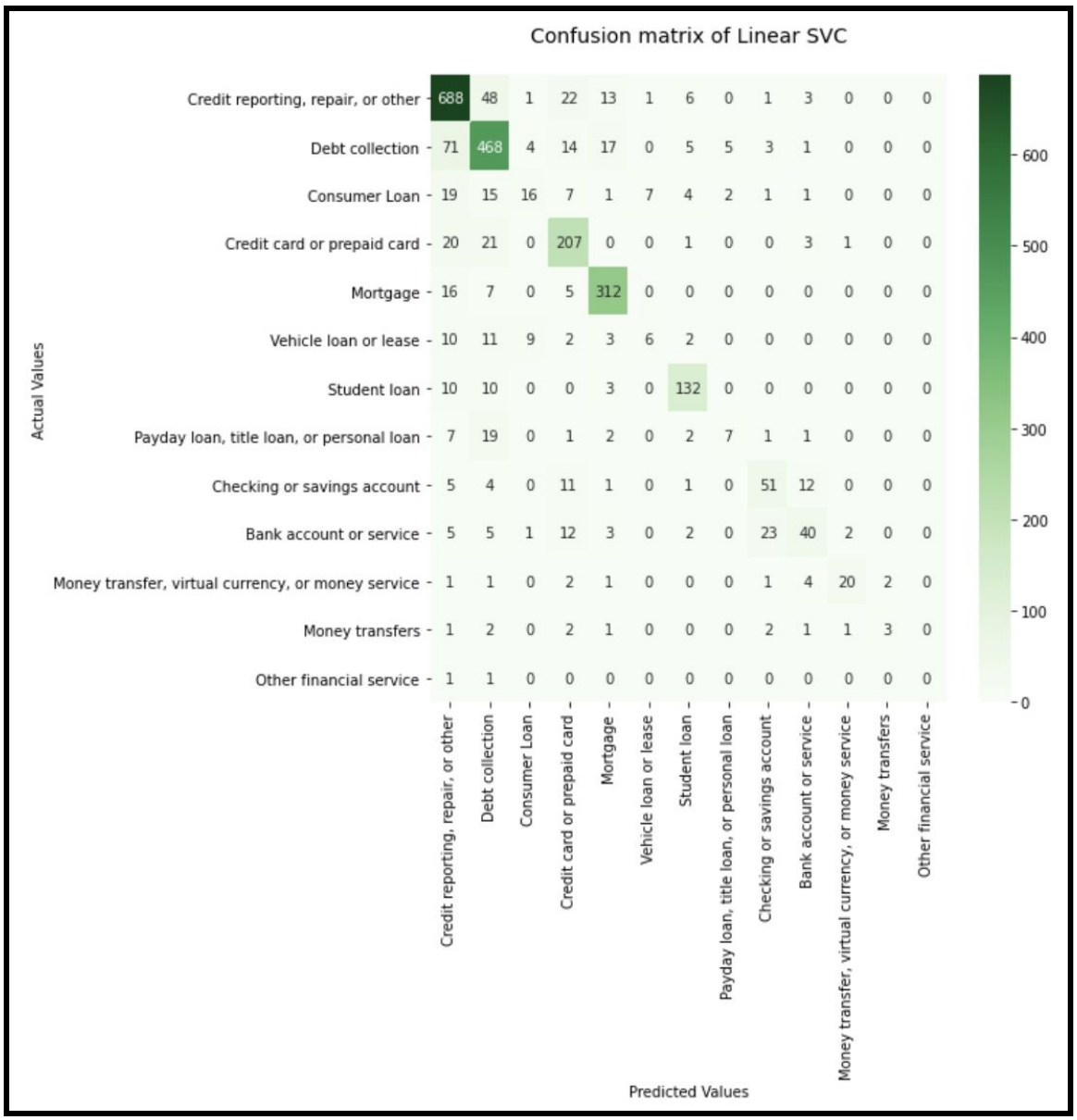
CLASSIFICATIION METRICS

	precision	recall	f1-score	support
Credit reporting, repair, or other	0.81	0.88	0.84	783
Debt collection	0.76	0.80	0.78	588
Consumer Loan	0.52	0.22	0.31	73
Credit card or prepaid card	0.73	0.82	0.77	253
Mortgage	0.87	0.92	0.90	340
Vehicle loan or lease	0.43	0.14	0.21	43
Student loan	0.85	0.85	0.85	155
Payday loan, title loan, or personal loan	0.50	0.17	0.26	40
Checking or savings account	0.61	0.60	0.61	85
Bank account or service	0.61	0.43	0.50	93
Money transfer, virtual currency, or money service	0.83	0.62	0.71	32
Money transfers	0.60	0.23	0.33	13
Other financial service	0.00	0.00	0.00	2
accuracy			0.78	2500
macro avg	0.62	0.51	0.54	2500
weighted avg	0.77	0.78	0.77	2500

We can observe that the categories which have higher values of support tend to have a better f1-score because the model is trained with more data. Also, from the above metrics we can see that ‘Credit reporting, repair or other’, ‘Mortgage’ and ‘Student Loan’ have good precision. So, they can be classified with very few errors. Considering the F1-scores of all categories, we get an accuracy of our model as 78% which can be considered as pretty good for classification.

Confusion Matrix :

```
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8,8))
sns.heatmap(conf_mat, annot=True, cmap="Greens", fmt='d',
            xticklabels=category_id_df.Product.values,
            yticklabels=category_id_df.Product.values)
plt.ylabel('Actual Values')
plt.xlabel('Predicted Values')
plt.title("Confusion matrix of Linear SVC\n", size=14);
```



A confusion matrix is a table in which the columns represent the predicted class and the rows represent the actual class. For a perfect model that classifies correctly a new complaint, the confusion matrix will have values in the diagonals only i.e predicted label = actual label. Overall, our model's confusion matrix looks good as its values are more concentrated in the diagonals but there's some misclassifications as well. Following are some misclassifications that our model has predicted:

```
for predicted in category_id_df.CategoryId:
  for actual in category_id_df.CategoryId:
    if predicted != actual and conf_mat[actual, predicted] >= 20:
      print("'{}' predicted as '{}' : {} examples.".format(id_to_category[actual],
                                                          id_to_category[predicted],
                                                          conf_mat[actual, predicted]))

      display(df2.loc[indices_test[(y_test == actual) & (y_pred == predicted)]][['Product',
                                                                                 'Complaint']])
      print('')
```

```
'Credit card or prepaid card' predicted as 'Debt collection' : 21 examples.
'Credit reporting, repair, or other' predicted as 'Credit card or prepaid card' : 22 examples.'Debt collection' predicted as 'Credit reporting, repair, or other' : 71 examples.
'Credit card or prepaid card' predicted as 'Credit reporting, repair, or other' : 20 examples
'Credit reporting, repair, or other' predicted as 'Debt collection' : 48 examples.

'Bank account or service' predicted as 'Checking or savings account' : 23 examples.
```

4.2 Justification

After selecting the best model, we'll try to predict and see how our model works. The required code used for prediction is as follows:

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state = 0)

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5,
                        ngram_range=(1, 2),
                        stop_words='english')

fitted_vectorizer = tfidf.fit(X_train)
tfidf_vectorizer_vectors = fitted_vectorizer.transform(X_train)

model = LinearSVC().fit(tfidf_vectorizer_vectors, y_train)
```

So, after our model is ready we must enter a new complaint and check how our model classifies it.

```
new_complaint = """I am Subhiksha, I am studying at Syracuse University. Recently, i have been harassed by \
Navient for the last month. I have taken a loan and paid it back with interests. I have faxed in paperwork providing them with everything they needed. And yet I am still getting \
phone calls for payments. Furthermore, Navient is now reporting to the credit bureaus that I am late. At this point, \
Navient needs to get their act together to avoid me taking further action. I have been enrolled the entire time and my \
deferment should be valid with my planned graduation date being the XX/XX/XXXX."""
print(model.predict(fitted_vectorizer.transform([new_complaint])))
```

For the given new complaint, our model has successfully predicted that this complaint belongs to the category of ‘Student Loans’ as the unigram ‘Navient’ is highly correlated with the category of Student Loans and this example has mentioned ‘Navient’ three times which has a high TF-IDF score. Hence, we can say that our model does a good job in classifying new complaints although it is also true that it won’t be correct all the time.

4.3. Model Comparison

After looking at various resources, papers, websites and comparing different text classification algorithms, we found that for a huge dataset like CFPB, the Linear Support Vector Classifier works efficiently with an accuracy range of 75-80% based on the data provided. In a few cases, Neural Networks also does a good job in classifying texts with an accuracy range of 76-82% but it depends on the data. We can say that the text classification algorithms have a benchmark accuracy of approximately 75-80% and given that our model gives 78% accuracy which makes it reliable.

5. Conclusion

5.1 Reflection

These are the summary of things which we have done in this project :

- Collection of data, filtering out the required features and cleansing the data
- Renaming product categories to remove ambiguity and taking only the unique products
- Vectorizing the data and finding the highly correlated terms using unigrams and bigrams
- Finding accuracy scores of multiple models on cross-validated data to choose the best model
- Using the model with best accuracy(LinearSVC) for predictions and calculating the classification metrics(78% accuracy)
- Defining functions to calculate negative, neutral and compound scores for sentiment analysis using VADER

5.2 Future Scope

- Using Sentiment Analysis, to predict companies making the most and the least profit in a year by evaluating the company's response to financial product complaints.
- Evaluate company's response to consumer's complaints to filter the best companies.
- Categorize the companies according to the state, to see which state performs well.
- Increasing the accuracy with the help of trigrams, making it a better model but should have to consider the time consumed too.

6. References

1. F. M. Yildirim *et al*, "A real-world text classification application for an E-commerce platform," in 2019,.
2. S. Das, "Sentiment Analysis of Twitter Data using various Classification Algorithms." , ProQuest Dissertations Publishing, 2018.
3. E. A. Emon *et al*, "A deep learning approach to detect abusive bengali text," in 2019,.
4. Li Susan. "Multi-Class Text Classification with Scikit-Learn". https://towardsdatascience.com/multi-class-text-classification-with-scikit-learn-12f1e60e0a9f
5. P. Rivas Perea. "Algorithms for Training Large-Scale Linear Programming Support Vector Regression and Classification". Order No. 3457758, The University of Texas at El Paso, Ann Arbor, 2011.
6. Pandey Parul. "Simplifying Sentiment Analysis using VADER in Python". https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f
7. "Using VADER to handle sentiment analysis with social media text". http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html
8. Gandhi Rohit. "Support Vector Machine - Introduction to Machine Learning Algorithms". https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47
9. Ray Sunil. "Understanding Support Vector Machine(SVM) algorithm from examples". https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/
10. Yiu Tony. "Understanding Random Forest: how the algorithm works and why it is so effective". https://towardsdatascience.com/understanding-random-forest-58381e0602d2
11. Nazrul Syed. "Multinomial Naive Bayes Classifier for Text Analysis (Python)". https://towardsdatascience.com/multinomial-naive-bayes-classifier-for-text-analysis-python-8dd6825ece67
12. Gupta Shashank. "Sentiment Analysis: Concept, Analysis and Applications". https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17
13. Malkin Cory. "TF IDF | TF IDF Python Example". https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76