In [1]:

```
1 import numpy as np
2 import pandas as pd
```

```
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/pandas/core/computation/expressions.py:20
: UserWarning: Pandas requires version '2.7.3' or newer of 'numexpr' (version '2.7.1' currently install
ed).
  from pandas.core.computation.check import NUMEXPR_INSTALLED
```

In [2]:
```
full_data = pd.read_csv("new_complaints.csv")
full_data.head()
```

Out[2]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	C p
0	3/12/14	Mortgage	Other mortgage	Loan modification,collection,foreclosure	NaN	NaN	NaN	M&T BANK CORPORATION	MI	48382	NaN	
1	1/19/17	Student loan	Federal student loan servicing	Dealing with my lender or servicer	Received bad information about my loan	When my loan was switched over to Navient i wa...	NaN	Navient Solutions, LLC.	LA	NaN	NaN	
2	4/6/18	Credit card or prepaid card	General-purpose credit card or charge card	Other features, terms, or problems	Other problem	I tried to sign up for a spending monitoring p...	NaN	CAPITAL ONE FINANCIAL CORPORATION	VA	NaN	Older American	
3	6/8/14	Credit card	NaN	Bankruptcy	NaN	NaN	NaN	AMERICAN EXPRESS COMPANY	ID	83854	Older American	
4	9/13/14	Debt collection	Credit card	Communication tactics	Frequent or repeated calls	NaN	NaN	CITIBANK, N.A.	VA	23233	NaN	

In [3]:
```
full_data.shape
```

Out[3]: (1669, 18)

In [4]:

```
# subset on data

data = full_data[['Consumer complaint narrative','Product']]
data.columns = ['X','y']
data.head()
```

Out[4]:

	X	y
0	NaN	Mortgage
1	When my loan was switched over to Navient i wa...	Student loan
2	I tried to sign up for a spending monitoring p...	Credit card or prepaid card
3	NaN	Credit card
4	NaN	Debt collection

In [5]:

```
print(data['X'][1])
```

```
When my loan was switched over to Navient i was never told that i had a deliquint balance because with
XXXX i did not. When going to purchase a vehicle i discovered my credit score had been dropped from the
XXXX into the XXXX. I have been faithful at paying my student loan. I was told that Navient was the com
pany i had delinquency with. I contacted Navient to resolve this issue you and kept being told to just
contact the credit bureaus and expalin the situation and maybe they could help me. I was so angry that
i just hurried and paid the balance off and then after tried to dispute the delinquency with the credit
bureaus. I have had so much trouble bringing my credit score back up.
```

In [6]:
```
print(data['y'].value_counts())
```

```
y
Mortgage                                                                  424
Debt collection                                                           308
Credit reporting                                                          245
Credit reporting, credit repair services, or other personal consumer reports    169
Credit card                                                               156
Bank account or service                                                   153
Student loan                                                               61
Consumer Loan                                                              55
Credit card or prepaid card                                                33
Checking or savings account                                                28
Money transfers                                                             8
Payday loan                                                                 8
Vehicle loan or lease                                                       7
Money transfer, virtual currency, or money service                          5
Payday loan, title loan, or personal loan                                   5
Other financial service                                                     2
Prepaid card                                                                2
Name: count, dtype: int64
```

In [7]:
```
print(data['y'].unique())
```

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

In [8]:
```
import nltk
# nltk.download('punkt')
```

Word Tokenize

In [9]:
```
data.dropna(inplace = True)
```

```
<ipython-input-9-48a623a66630>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  data.dropna(inplace = True)
```

In [10]:
```
# first complaint

first_complaint = data.iloc[0][0]
first_complaint
```

Out[10]:
```
'When my loan was switched over to Navient i was never told that i had a deliquint balance because with XXXX i did not. When going to purchase a vehicle i discovered my credit score had been dropped from the XXXX into the XXXX. I have been faithful at paying my student loan. I was told that Navient was the company i had delinquency with. I contacted Navient to resolve this issue you and kept being told to just contact the credit bureaus and expalin the situation and maybe they could help me. I was so angry that i just hurried and paid the balance off and then after tried to dispute the delinquency with the credit bureaus. I have had so much trouble bringing my credit score back up.'
```

In [11]:
```
# apply word tokenizer
from nltk.tokenize import word_tokenize
word_token = word_tokenize(first_complaint)
print(word_token)
```

```
['When', 'my', 'loan', 'was', 'switched', 'over', 'to', 'Navient', 'i', 'was', 'never', 'told', 'that',
'i', 'had', 'a', 'deliquint', 'balance', 'because', 'with', 'XXXX', 'i', 'did', 'not', '.', 'When', 'go
ing', 'to', 'purchase', 'a', 'vehicle', 'i', 'discovered', 'my', 'credit', 'score', 'had', 'been', 'dro
pped', 'from', 'the', 'XXXX', 'into', 'the', 'XXXX', '.', 'I', 'have', 'been', 'faithful', 'at', 'payin
g', 'my', 'student', 'loan', '.', 'I', 'was', 'told', 'that', 'Navient', 'was', 'the', 'company', 'i',
'had', 'delinquency', 'with', '.', 'I', 'contacted', 'Navient', 'to', 'resolve', 'this', 'issue', 'you'
, 'and', 'kept', 'being', 'told', 'to', 'just', 'contact', 'the', 'credit', 'bureaus', 'and', 'expalin'
, 'the', 'situation', 'and', 'maybe', 'they', 'could', 'help', 'me', '.', 'I', 'was', 'so', 'angry', 't
hat', 'i', 'just', 'hurried', 'and', 'paid', 'the', 'balance', 'off', 'and', 'then', 'after', 'tried',
'to', 'dispute', 'the', 'delinquency', 'with', 'the', 'credit', 'bureaus', '.', 'I', 'have', 'had', 'so
', 'much', 'trouble', 'bringing', 'my', 'credit', 'score', 'back', 'up', '.']
```

Sent tokenize

In [12]:
```
# apply word tokenizer
from nltk.tokenize import sent_tokenize
sent_token = sent_tokenize(first_complaint)
print(sent_token)
```

```
['When my loan was switched over to Navient i was never told that i had a deliquint balance because wit
h XXXX i did not.', 'When going to purchase a vehicle i discovered my credit score had been dropped fro
m the XXXX into the XXXX.', 'I have been faithful at paying my student loan.', 'I was told that Navient
was the company i had delinquency with.', 'I contacted Navient to resolve this issue you and kept being
told to just contact the credit bureaus and expalin the situation and maybe they could help me.', 'I wa
s so angry that i just hurried and paid the balance off and then after tried to dispute the delinquency
with the credit bureaus.', 'I have had so much trouble bringing my credit score back up.']
```

Stemming

In [13]:
```
text = "Natural language processing is really fun and i want to study it more"
tokens = word_tokenize(text)

porter = nltk.PorterStemmer()

stem = [porter.stem(i) for i in tokens]

print(stem)
```

```
['natur', 'languag', 'process', 'is', 'realli', 'fun', 'and', 'i', 'want', 'to', 'studi', 'it', 'more']
```

Lemmatizer

In [14]:
```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/kunalshriwas/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[14]:
```
True
```

In [15]:
```
from nltk.stem import WordNetLemmatizer
text = "Natural language processing is really fun and i want to study it more"
tokens = word_tokenize(text)
lemma = WordNetLemmatizer()
lemma_word = [lemma.lemmatize(i) for i in tokens]
print(lemma_word)
```

```
['Natural', 'language', 'processing', 'is', 'really', 'fun', 'and', 'i', 'want', 'to', 'study', 'it', '
more']
```

Count vectorizer

In [16]:
```
# method 1
```

In [17]:
```
from collections import Counter
count_vectorizer = Counter(word_token)
print(count_vectorizer)
```

```
Counter({'the': 8, '.': 7, 'i': 6, 'was': 5, 'to': 5, 'I': 5, 'and': 5, 'my': 4, 'had': 4, 'credit': 4,
'Navient': 3, 'told': 3, 'that': 3, 'with': 3, 'XXXX': 3, 'When': 2, 'loan': 2, 'a': 2, 'balance': 2, '
score': 2, 'been': 2, 'have': 2, 'delinquency': 2, 'just': 2, 'bureaus': 2, 'so': 2, 'switched': 1, 'ov
er': 1, 'never': 1, 'deliquint': 1, 'because': 1, 'did': 1, 'not': 1, 'going': 1, 'purchase': 1, 'vehic
le': 1, 'discovered': 1, 'dropped': 1, 'from': 1, 'into': 1, 'faithful': 1, 'at': 1, 'paying': 1, 'stud
ent': 1, 'company': 1, 'contacted': 1, 'resolve': 1, 'this': 1, 'issue': 1, 'you': 1, 'kept': 1, 'being
': 1, 'contact': 1, 'expalin': 1, 'situation': 1, 'maybe': 1, 'they': 1, 'could': 1, 'help': 1, 'me': 1
, 'angry': 1, 'hurried': 1, 'paid': 1, 'off': 1, 'then': 1, 'after': 1, 'tried': 1, 'dispute': 1, 'much
': 1, 'trouble': 1, 'bringing': 1, 'back': 1, 'up': 1})
```

In [18]:
```
# method 2
```

In [19]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer()

txt = [data['X'].iloc[0]]

cv.fit(txt)

vector = cv.transform(txt)

vector_value = vector.toarray()

print("vector_value",vector_value)
print("=====================================================================")
print("Vocab : ", cv.vocabulary_)
```

```
vector_value [[1 5 1 1 1 2 1 2 1 1 2 1 1 1 1 4 2 1 1 1 1 1 1 1 1 4 2 1 1 1 1 2 1 2 1
  1 1 4 3 1 1 1 1 1 1 1 1 2 1 2 1 1 3 8 1 1 1 5 3 1 1 1 1 5 2 3 3 1]]
=====================================================================
Vocab :  {'when': 65, 'my': 38, 'loan': 34, 'was': 64, 'switched': 52, 'over': 43, 'to': 58, 'navient':
39, 'never': 40, 'told': 59, 'that': 53, 'had': 26, 'deliquint': 17, 'balance': 5, 'because': 6, 'with'
: 66, 'xxxx': 67, 'did': 18, 'not': 41, 'going': 25, 'purchase': 46, 'vehicle': 63, 'discovered': 19, '
credit': 15, 'score': 48, 'been': 7, 'dropped': 21, 'from': 24, 'the': 54, 'into': 30, 'have': 27, 'fai
thful': 23, 'at': 3, 'paying': 45, 'student': 51, 'company': 11, 'delinquency': 16, 'contacted': 13, 'r
esolve': 47, 'this': 57, 'issue': 31, 'you': 68, 'and': 1, 'kept': 33, 'being': 8, 'just': 32, 'contact
': 12, 'bureaus': 10, 'expalin': 22, 'situation': 49, 'maybe': 35, 'they': 56, 'could': 14, 'help': 28,
'me': 36, 'so': 50, 'angry': 2, 'hurried': 29, 'paid': 44, 'off': 42, 'then': 55, 'after': 0, 'tried':
60, 'dispute': 20, 'much': 37, 'trouble': 61, 'bringing': 9, 'back': 4, 'up': 62}
```

In [20]:
```
txt
```

Out[20]:
```
['When my loan was switched over to Navient i was never told that i had a deliquint balance because wit
h XXXX i did not. When going to purchase a vehicle i discovered my credit score had been dropped from t
he XXXX into the XXXX. I have been faithful at paying my student loan. I was told that Navient was the
company i had delinquency with. I contacted Navient to resolve this issue you and kept being told to ju
st contact the credit bureaus and expalin the situation and maybe they could help me. I was so angry th
at i just hurried and paid the balance off and then after tried to dispute the delinquency with the cre
dit bureaus. I have had so much trouble bringing my credit score back up.']
```

Classification using BOW

In [21]:
```
from sklearn.metrics import accuracy_score,roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.preprocessing import LabelEncoder
```

In [22]:

```
all_data = data[['X']]

all_data['X'] = all_data['X'].str.lower()

cv = CountVectorizer()

vector = cv.fit_transform(all_data['X'])

X = vector.toarray()

labels = data[['y']]

le = LabelEncoder()

labels['y'] = le.fit_transform(labels['y'])

X_train, X_test, y_train, y_test = train_test_split(X,labels['y'],test_size=0.3,random_state=42)

log_reg = LogisticRegression(random_state=42)

log_reg.fit(X_train,y_train)

acc = log_reg.score(X_test,y_test)

print(acc)
```

```
<ipython-input-22-3dac13234bd7>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  all_data['X'] = all_data['X'].str.lower()
<ipython-input-22-3dac13234bd7>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
```

```
html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy)
  labels['y'] = le.fit_transform(labels['y'])
```

```
0.5148514851485149
```

```
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:458: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules
/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn
.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Stopwords

In [23]:

```
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
print(set(stopwords.words('english')))
```

```
{'d', 'under', 'y', 'than', 'only', 'shan', "wasn't", 'themselves', 'her', 'on', 'with', 'weren', "shou
ld've", "wouldn't", 'hadn', 'hers', 'their', 'you', 'whom', 'so', 'where', 'same', 'isn', 'during', 'or
', 'have', 'that', 'some', 'our', 'no', 'll', 'your', "that'll", 'o', 'just', 'haven', 'being', 'up', '
few', 'most', "mustn't", 'this', 'other', 'by', 'do', 'to', 'but', 'aren', 'of', 'nor', 'not', 'needn',
'ma', 'further', 'each', 'between', 'then', 'doing', "couldn't", 'too', 'don', 'an', "hadn't", 'yoursel
ves', 'hasn', "hasn't", 'from', 'above', "you've", 'were', 'at', 'how', "doesn't", 'did', "mightn't", '
me', "won't", 'she', 'these', 'is', 's', 'they', 'yours', 'into', 'be', 'wouldn', 'if', 'against', 'mig
htn', 'more', "it's", 'ourselves', 'been', 'the', "aren't", 'doesn', "you'd", "weren't", 'itself', 'onc
e', "you'll", 'until', 'why', 'as', 've', 'we', 'herself', 'those', 't', 'are', "didn't", 'about', 'hav
ing', 'before', 'a', 'wasn', 'them', 'any', 'myself', "needn't", "she's", "shan't", 'does', 'am', 'afte
r', 'couldn', 'because', 'its', 'should', 'for', 'off', 'i', 'my', 'himself', 'all', 'very', 'mustn', '
will', "isn't", 'out', 'was', 'he', 'which', 'his', 'here', 'while', 'had', 'when', 'again', 'down', "s
houldn't", 'now', 'both', 'm', 'what', 'him', "don't", 'own', 'through', 'such', 'can', 'didn', 'who',
'and', 'ain', 'it', 'theirs', 'shouldn', 'over', 're', 'ours', 'in', 'there', "haven't", 'below', "you'
re", 'yourself', 'has', 'won'}

[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/kunalshriwas/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Stopword removal from sample data

In [24]:

```python
stop_words = set(stopwords.words('english'))
print("First_complaint : ", first_complaint)
print("=========================================================================")
bow = word_tokenize(first_complaint)
print("Words before performing stopword removal",bow)
print("Lenght of words before performing stopword removal : --------->",len(bow))
print("=========================================================================")

bow_stop_word_removed = [x for x in bow if x not in stop_words ]
print("Words after performing stopword removal",bow_stop_word_removed)
print("Lenght of words after performing stopword removal : --------->",len(bow_stop_word_removed))
```

```
First_complaint :  When my loan was switched over to Navient i was never told that i had a deliquint ba
lance because with XXXX i did not. When going to purchase a vehicle i discovered my credit score had be
en dropped from the XXXX into the XXXX. I have been faithful at paying my student loan. I was told that
Navient was the company i had delinquency with. I contacted Navient to resolve this issue you and kept
being told to just contact the credit bureaus and expalin the situation and maybe they could help me. I
was so angry that i just hurried and paid the balance off and then after tried to dispute the delinquen
cy with the credit bureaus. I have had so much trouble bringing my credit score back up.
=========================================================================
Words before performing stopword removal ['When', 'my', 'loan', 'was', 'switched', 'over', 'to', 'Navie
nt', 'i', 'was', 'never', 'told', 'that', 'i', 'had', 'a', 'deliquint', 'balance', 'because', 'with', '
XXXX', 'i', 'did', 'not', '.', 'When', 'going', 'to', 'purchase', 'a', 'vehicle', 'i', 'discovered', 'm
y', 'credit', 'score', 'had', 'been', 'dropped', 'from', 'the', 'XXXX', 'into', 'the', 'XXXX', '.', 'I'
, 'have', 'been', 'faithful', 'at', 'paying', 'my', 'student', 'loan', '.', 'I', 'was', 'told', 'that',
'Navient', 'was', 'the', 'company', 'i', 'had', 'delinquency', 'with', '.', 'I', 'contacted', 'Navient'
, 'to', 'resolve', 'this', 'issue', 'you', 'and', 'kept', 'being', 'told', 'to', 'just', 'contact', 'th
e', 'credit', 'bureaus', 'and', 'expalin', 'the', 'situation', 'and', 'maybe', 'they', 'could', 'help',
'me', '.', 'I', 'was', 'so', 'angry', 'that', 'i', 'just', 'hurried', 'and', 'paid', 'the', 'balance',
'off', 'and', 'then', 'after', 'tried', 'to', 'dispute', 'the', 'delinquency', 'with', 'the', 'credit',
'bureaus', '.', 'I', 'have', 'had', 'so', 'much', 'trouble', 'bringing', 'my', 'credit', 'score', 'back
', 'up', '.']
Lenght of words before performing stopword removal : ---------> 137
=========================================================================
Words after performing stopword removal ['When', 'loan', 'switched', 'Navient', 'never', 'told', 'deliq
uint', 'balance', 'XXXX', '.', 'When', 'going', 'purchase', 'vehicle', 'discovered', 'credit', 'score',
'dropped', 'XXXX', 'XXXX', '.', 'I', 'faithful', 'paying', 'student', 'loan', '.', 'I', 'told', 'Navien
t', 'company', 'delinquency', '.', 'I', 'contacted', 'Navient', 'resolve', 'issue', 'kept', 'told', 'co
ntact', 'credit', 'bureaus', 'expalin', 'situation', 'maybe', 'could', 'help', '.', 'I', 'angry', 'hurr
```

```
ied', 'paid', 'balance', 'tried', 'dispute', 'delinquency', 'credit', 'bureaus', '.', 'I', 'much', 'tro
uble', 'bringing', 'credit', 'score', 'back', '.']
Lenght of words after performing stopword removal : ---------> 68
```

Classification using BOW with stop word removal

In [25]:

```
all_data = data[['X']]

all_data['X'] = all_data['X'].str.lower()

cv_stop = CountVectorizer(stop_words='english')

vector_stop = cv_stop.fit_transform(all_data['X'])

X_stop = vector_stop.toarray()

labels = data[['y']]

le = LabelEncoder()

labels['y'] = le.fit_transform(labels['y'])

X_train, X_test, y_train, y_test = train_test_split(X_stop,labels['y'],test_size=0.3,random_state=42

log_reg_stop = LogisticRegression(random_state=42)

log_reg_stop.fit(X_train,y_train)

acc_stop = log_reg_stop.score(X_test,y_test)

print(acc_stop)
```

```
<ipython-input-25-5c49cb680785>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
```

```
html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy)
  all_data['X'] = all_data['X'].str.lower()
<ipython-input-25-5c49cb680785>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy)
  labels['y'] = le.fit_transform(labels['y'])

0.5445544554455446

/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/sklearn/linear_model/_logistic.py:458: Co
nvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-learn.org/stable/modules
/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (https://scikit-learn
.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

TF - IDF

In [26]:

```
complaint_1 = data['X'].iloc[0]
complaint_2 = data['X'].iloc[1]
complaint_3 = data['X'].iloc[2]

print("Complaint 1 : ",complaint_1)
print("\nComplaint 2 : ",complaint_2)
print("\nComplaint 3 : ",complaint_3)

```

```
Complaint 1 :  When my loan was switched over to Navient i was never told that i had a deliquint balanc
e because with XXXX i did not. When going to purchase a vehicle i discovered my credit score had been d
ropped from the XXXX into the XXXX. I have been faithful at paying my student loan. I was told that Nav
ient was the company i had delinquency with. I contacted Navient to resolve this issue you and kept bei
ng told to just contact the credit bureaus and expalin the situation and maybe they could help me. I wa
s so angry that i just hurried and paid the balance off and then after tried to dispute the delinquency
with the credit bureaus. I have had so much trouble bringing my credit score back up.

Complaint 2 :  I tried to sign up for a spending monitoring program and Capital One will not let me acc
ess my account through them

Complaint 3 :  My mortgage is with BB & T Bank, recently I have been investigating ways to pay down my
mortgage faster and I came across Biweekly Mortgage Calculator on BB & T 's website. It's a nice, easy
to use calculator that you plug in your interest rate, mortgage amount, mortgage term, and payment type
and it calculates your accelerated bi-weekly payment for you and shows you how much quicker you can pay
down your loan. Ours figured out to pay off a 30 year mortgage in 26.4 years ... quite a savings!
I called BB & T 's customer service number to inquire how I get set up on this payment plan. I was told
they do not offer that type of payment plan, but I could send in my payments bi-weekly but it would not
be applied until the full amount was received. ( the money would sit in a " holding account '' until th
e full payment amount was collected ). I ended up calling back a few days later thinking the rep I was
talking to didn't understand what I wanted to do or was not knowledgeable of this program. I got the SA
ME ANSWER!
I then asked for the corporate BB & T office number where I could speak to someone that was knowledgeab
le of this product. After 3 days I received a phone call back from a corporate manager stating they do
not offer this product, and they were " checking into why this is on their website ''. She stated they
do have a few customers that make bi-weekly payments, but they no longer offer this service.
I don't understand how they can have this active link on their website under their Financial Planning C
enter tab to mislead customers when all they say is " I'm sorry, I know you're upset about this '' Soun
ds like false advertising to me!
https : //www.bbt.com/XXXX
```

In [28]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

sents = [complaint_1,complaint_2,complaint_3]

vectorizer = TfidfVectorizer()

vectorizer.fit(sents)

vector = vectorizer.transform(sents)

print("shape of vectorized sentense : ",vector.shape)

vector_values = vector.toarray().tolist()[0]

print("TFIDF score of first 10 elements:  ", vector_values[:10])
```

```
shape of vectorized sentense :  (3, 214)
TFIDF score of first 10 elements:   [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.05253580411952334]
```

In [42]:

```
# Convert tfidf score with word into a dictionary

import operator

sorted_x = sorted(vectorizer.vocabulary_.items(),key = operator.itemgetter(1))

words = [x[0] for x in sorted_x]

d = dict(zip(words,vector_values))
#print("Dictionary of words with TFIDF score : ", d)
# sorting this dictionary by value in the descending order to see ranking
print("sorting dictionary  : \n")
print(sorted(d.items(), key = operator.itemgetter(1),reverse = True))
```

```
sorting dictionary  :

[('the', 0.42028643295618673), ('credit', 0.27631307611219824), ('had', 0.27631307611219824), ('was', 0
.2626790205976167), ('navient', 0.20723480708414868), ('and', 0.20399369240069398), ('to', 0.2039936924
0069398), ('my', 0.1631949539205552), ('that', 0.15760741235857004), ('told', 0.15760741235857004), ('w
```

ith', 0.15760741235857004), ('xxxx', 0.15760741235857004), ('balance', 0.13815653805609912), ('bureaus', 0.13815653805609912), ('delinquency', 0.13815653805609912), ('just', 0.13815653805609912), ('score', 0.13815653805609912), ('so', 0.13815653805609912), ('been', 0.10507160823904668), ('have', 0.10507160823904668), ('loan', 0.10507160823904668), ('when', 0.10507160823904668), ('angry', 0.06907826902804956), ('at', 0.06907826902804956), ('because', 0.06907826902804956), ('being', 0.06907826902804956), ('bringing', 0.06907826902804956), ('company', 0.06907826902804956), ('contact', 0.06907826902804956), ('contacted', 0.06907826902804956), ('deliquint', 0.06907826902804956), ('did', 0.06907826902804956), ('discovered', 0.06907826902804956), ('dispute', 0.06907826902804956), ('dropped', 0.06907826902804956), ('expalin', 0.06907826902804956), ('faithful', 0.06907826902804956), ('going', 0.06907826902804956), ('help', 0.06907826902804956), ('hurried', 0.06907826902804956), ('issue', 0.06907826902804956), ('kept', 0.06907826902804956), ('maybe', 0.06907826902804956), ('never', 0.06907826902804956), ('over', 0.06907826902804956), ('paid', 0.06907826902804956), ('paying', 0.06907826902804956), ('purchase', 0.06907826902804956), ('resolve', 0.06907826902804956), ('situation', 0.06907826902804956), ('student', 0.06907826902804956), ('switched', 0.06907826902804956), ('trouble', 0.06907826902804956), ('vehicle', 0.06907826902804956), ('after', 0.05253580411952334), ('back', 0.05253580411952334), ('could', 0.05253580411952334), ('from', 0.05253580411952334), ('into', 0.05253580411952334), ('much', 0.05253580411952334), ('off', 0.05253580411952334), ('then', 0.05253580411952334), ('they', 0.05253580411952334), ('this', 0.05253580411952334), ('tried', 0.05253580411952334), ('you', 0.05253580411952334), ('me', 0.0407987384801388), ('not', 0.0407987384801388), ('up', 0.0407987384801388), ('26', 0.0), ('30', 0.0), ('about', 0.0), ('accelerated', 0.0), ('access', 0.0), ('account', 0.0), ('across', 0.0), ('active', 0.0), ('advertising', 0.0), ('all', 0.0), ('amount', 0.0), ('answer', 0.0), ('applied', 0.0), ('asked', 0.0), ('bank', 0.0), ('bb', 0.0), ('bbt', 0.0), ('be', 0.0), ('bi', 0.0), ('biweekly', 0.0), ('but', 0.0), ('calculates', 0.0), ('calculator', 0.0), ('call', 0.0), ('called', 0.0), ('calling', 0.0), ('came', 0.0), ('can', 0.0), ('capital', 0.0), ('center', 0.0), ('checking', 0.0), ('collected', 0.0), ('com', 0.0), ('corporate', 0.0), ('customer', 0.0), ('customers', 0.0), ('days', 0.0), ('didn', 0.0), ('do', 0.0), ('don', 0.0), ('down', 0.0), ('easy', 0.0), ('ended', 0.0), ('false', 0.0), ('faster', 0.0), ('few', 0.0), ('figured', 0.0), ('financial', 0.0), ('for', 0.0), ('full', 0.0), ('get', 0.0), ('got', 0.0), ('holding', 0.0), ('how', 0.0), ('https', 0.0), ('in', 0.0), ('inquire', 0.0), ('interest', 0.0), ('investigating', 0.0), ('is', 0.0), ('it', 0.0), ('know', 0.0), ('knowledgeable', 0.0), ('later', 0.0), ('let', 0.0), ('like', 0.0), ('link', 0.0), ('longer', 0.0), ('make', 0.0), ('manager', 0.0), ('mislead', 0.0), ('money', 0.0), ('monitoring', 0.0), ('mortgage', 0.0), ('nice', 0.0), ('no', 0.0), ('number', 0.0), ('of', 0.0), ('offer', 0.0), ('office', 0.0), ('on', 0.0), ('one', 0.0), ('or', 0.0), ('ours', 0.0), ('out', 0.0), ('pay', 0.0), ('payment', 0.0), ('payments', 0.0), ('phone', 0.0), ('plan', 0.0), ('planning', 0.0), ('plug', 0.0), ('product', 0.0), ('program', 0.0), ('quicker', 0.0), ('quite', 0.0), ('rate', 0.0), ('re', 0.0), ('received', 0.0), ('recently', 0.0), ('rep', 0.0), ('same', 0.0), ('savings', 0.0), ('say', 0.0), ('send', 0.0), ('service', 0.0), ('set', 0.0), ('she', 0.0), ('shows', 0.0), ('sign', 0.0), ('sit', 0.0), ('someone', 0.0), ('sorry', 0.0), ('sounds', 0.0), ('speak', 0.0), ('spending', 0.0), ('stated', 0.0), ('stating', 0.0), ('tab', 0.0), ('talking', 0.0), ('term', 0.0), ('their', 0.0), ('them', 0.0), ('thinking', 0.0), ('through', 0.0), ('type', 0.0), ('under', 0.0), ('understand', 0.0), ('until', 0.0), ('upset',

```
0.0), ('use', 0.0), ('wanted', 0.0), ('ways', 0.0), ('website', 0.0), ('weekly', 0.0), ('were', 0.0), (
'what', 0.0), ('where', 0.0), ('why', 0.0), ('will', 0.0), ('would', 0.0), ('www', 0.0), ('year', 0.0),
('years', 0.0), ('your', 0.0)]
```

In [48]:

```
# apply same on multiple docs

sents_100 = []

for x in range(101):
    sents_100.append(data['X'].iloc[x])

vectorizer = TfidfVectorizer()
vectorizer.fit(sents_100)
vector = vectorizer.transform(sents_100)
print("shape of vectorized sentense : ",vector.shape)
vector_values = vector.toarray().tolist()[0]
sorted_x100 = sorted(vectorizer.vocabulary_.items(),key = operator.itemgetter(1))
words_100 = [x[0] for x in sorted_x100]
d1 = dict(zip(words_100,vector_values))
#print("Dictionary of words with TFIDF score : ", d)
# sorting this dictionary by value in the descending order to see ranking
print("sorting dictionary  : \n")
print(sorted(d1.items(), key = operator.itemgetter(1),reverse = True))
```

```
shape of vectorized sentense :  (101, 2315)
sorting dictionary  :

[('navient', 0.3547748473050244), ('the', 0.23987035295364448), ('delinquency', 0.23651656487001627), (
'had', 0.21048139845175112), ('told', 0.19512462262839433), ('bureaus', 0.19224258840025857), ('was', 0
.18359992404858105), ('score', 0.1640783270390704), ('credit', 0.15996027451433503), ('just', 0.1524183
7508811337), ('balance', 0.14587820745037827), ('to', 0.1441121098032926), ('and', 0.13992032156143958)
, ('loan', 0.132405826069357), ('angry', 0.12885169562236412), ('deliquint', 0.12885169562236412), ('ex
palin', 0.12885169562236412), ('faithful', 0.12885169562236412), ('hurried', 0.12885169562236412), ('sw
itched', 0.12885169562236412), ('when', 0.12785920812359747), ('discovered', 0.11825828243500813), ('tr
ouble', 0.11825828243500813), ('so', 0.1179707188085681), ('my', 0.1175866217649111), ('been', 0.114485
8848637017), ('with', 0.11132115662871929), ('dropped', 0.11074213616462766), ('situation', 0.110742136
16462766), ('vehicle', 0.11074213616462766), ('bringing', 0.10491216018914912), ('issue', 0.10491216018
914912), ('kept', 0.10491216018914912), ('maybe', 0.10491216018914912), ('that', 0.10459877759431659),
('paying', 0.09612129420012928), ('xxxx', 0.08995138235761668), ('much', 0.08955530978991567), ('purcha
```

```
se', 0.08955530978991567), ('resolve', 0.08955530978991567), ('student', 0.08955530978991567), ('contac
t', 0.0820391635195352), ('dispute', 0.07801173474239283), ('into', 0.07801173474239283), ('tried', 0.0
7801173474239283), ('going', 0.07620918754405669), ('then', 0.07293910372518914), ('have', 0.0719252797
0288275), ('after', 0.0714457503321792), ('paid', 0.0714457503321792), ('help', 0.07003315920323935), (
```

In [ ]:
```
1
```

Classification using TF-IDF

In [71]:
```
1 df = pd.read_pickle("a2.pkl")
2 df.head()
```

Out[71]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags
0	03/12/2014	Mortgage	Other mortgage	Loan modification,collection,foreclosure	NaN	NaN	NaN	M&T BANK CORPORATION	MI	48382	NaN
1	01/19/2017	Student loan	Federal student loan servicing	Dealing with my lender or servicer	Received bad information about my loan	When my loan was switched over to Navient i wa...	NaN	Navient Solutions, LLC.	LA	NaN	NaN
2	04/06/2018	Credit card or prepaid card	General-purpose credit card or charge card	Other features, terms, or problems	Other problem	I tried to sign up for a spending monitoring p...	NaN	CAPITAL ONE FINANCIAL CORPORATION	VA	NaN	Older American

In [72]:
```
df.columns
```

Out[72]:
```
Index(['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'],
      dtype='object')
```

In [73]:
```
df = df[['Consumer complaint narrative','Product']]
df = df.dropna()
df.columns = ['X','y']
df = df.iloc[:2000]
df.shape
```

Out[73]:
```
(1806, 2)
```

In [74]:
```
all_text = df[['X']]
all_text['X']= all_text['X'].str.lower()
```

```
<ipython-input-74-81d7ef621500>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  all_text['X']= all_text['X'].str.lower()
```

In [ ]:
```

```

In [79]:

```
tfidf = TfidfVectorizer(stop_words = 'english')
vector = tfidf.fit_transform(all_text['X'])
X_tfidf = vector.toarray()
labels = df[['y']]
le = LabelEncoder()
labels['y'] = le.fit_transform(labels['y'])
X_train, X_test, y_train, y_test = train_test_split(X_tfidf,labels['y'],test_size=0.3,random_state=4
log_reg_tfidf = LogisticRegression(random_state=42)
log_reg_tfidf.fit(X_train,y_train)
acc_tfidf = log_reg_tfidf.score(X_test,y_test)
print(acc_tfidf)
```

```
<ipython-input-79-767b0c039ec8>:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.
html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.h
tml#returning-a-view-versus-a-copy)
  labels['y'] = le.fit_transform(labels['y'])

0.5940959409594095
```

In [85]:
```
from imblearn.over_sampling import RandomOverSampler
from sklearn.naive_bayes import MultinomialNB

ros = RandomOverSampler()

X_ros, y_ros = ros.fit_resample(X_train,y_train)

nb = MultinomialNB()

nb.fit(X_ros, y_ros)

ros_score = nb.score(X_test,y_test)

print(ros_score)
```

```
0.6014760147601476
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```