Configuration

```
#Parameters
PROJECT_NAME = 'ML1010_Weekly'
ENABLE_COLAB = True

#Root Machine Learning Directory. Projects appear underneath
GOOGLE_DRIVE_MOUNT = '/content/gdrive'
COLAB_ROOT_DIR = GOOGLE_DRIVE_MOUNT + '/MyDrive/Colab Notebooks'
COLAB_INIT_DIR = COLAB_ROOT_DIR + '/utility_files'

LOCAL_ROOT_DIR = '/home/magni/Documents/ML_Projects'
LOCAL_INIT_DIR = LOCAL_ROOT_DIR + '/utility_files'
```

Bootstrap Environment

```
#add in support for utility file directory and importing
import sys
import os

if ENABLE_COLAB:
  #Need access to drive
  from google.colab import drive
  drive.mount(GOOGLE_DRIVE_MOUNT, force_remount=True)

  #add in utility directory to syspath to import
  INIT_DIR = COLAB_INIT_DIR
  sys.path.append(os.path.abspath(INIT_DIR))

  #Config environment variables
  ROOT_DIR = COLAB_ROOT_DIR

else:
  #add in utility directory to syspath to import
  INIT_DIR = LOCAL_INIT_DIR
  sys.path.append(os.path.abspath(INIT_DIR))

  #Config environment variables
  ROOT_DIR = LOCAL_ROOT_DIR

#Import Utility Support
from jarvis import Jarvis
jarvis = Jarvis(ROOT_DIR, PROJECT_NAME)

import my_python_utils as myutils
```

```
Mounted at /content/gdrive
Wha...where am I?
I am awake now.

I have set your current working directory to /content/gdrive/MyDrive/Colab Notebooks/ML1
The current time is 11:57
Hello sir. Extra caffeine may help.
```

Setup Runtime Environment

```
if ENABLE_COLAB:
  #!pip install scipy -q
  #!pip install scikit-learn -q
  #!pip install pycaret -q
  #!pip install matplotlib -q
  #!pip install joblib -q
  #!pip install pandasql -q

  display('Google Colab enabled')
else:
  display('Google Colab not enabled')

#Common imports
import json
import gzip
import pandas as pd
import numpy as np
import matplotlib
import re
import nltk
import matplotlib.pyplot as plt

pd.set_option('mode.chained_assignment', None)
nltk.download('stopwords')
%matplotlib inline

import warnings

warnings.filterwarnings("ignore")
```

```
'Google Colab enabled'
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Load Data

```
jarvis.showAllDataFiles()
```

```
Here are all your available data files
[D] /content/gdrive/MyDrive/Colab Notebooks/data [Empty directory]

[D] /content/gdrive/MyDrive/Colab Notebooks/data/Jarvis
---[   gz][  pkl]--> 02_NLP_ReviewTextData.pkl.gz (30.08 MB)
---[   gz][  pkl]--> 02_NLP_SummaryData.pkl.gz (2.88 MB)
---[   gz][  csv]--> movie_reviews_cleaned.csv.gz (14.73 MB)
---[   gz][  csv]--> wk3_task_data.csv.gz (33.47 KB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/Jarvis/01_original [Empty directory]

[D] /content/gdrive/MyDrive/Colab Notebooks/data/Jarvis/02_working [Empty directory]

[D] /content/gdrive/MyDrive/Colab Notebooks/data/Jarvis/03_train
---[   gz][  pkl]--> 02_NLP_SummaryData.pkl.gz (2.88 MB)
---[   gz][  pkl]--> 02_NLP_TitleData.pkl.gz (1.43 MB)
---[   gz][  pkl]--> 03_NLP_ReviewTextData.pkl.gz (10.91 MB)
---[   gz][  pkl]--> 03_NLP_SummaryData.pkl.gz (1.62 MB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/Jarvis/04_test
---[   gz][  csv]--> pima-indians-diabetes.csv.gz (8.53 KB)
---[   gz][  csv]--> wk3_task_data.csv.gz (33.47 KB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project [Empty directory

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/01_original
---[   gz][ json]--> Cell_Phones_and_Accessories_5.json.gz (161.24 MB)
---[   gz][ json]--> meta_Cell_Phones_and_Accessories.json.gz (343.33 MB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/02_working
[*][  pkl]---------> 01_Cellphone_small.pkl (45.46 MB)
---[   gz][  pkl]--> 01_NLP_ReviewText_Narrow_1.pkl.gz (6.88 MB)
---[   gz][  pkl]--> 01_NLP_ReviewText_Narrow_2.pkl.gz (170.55 MB)
---[   gz][  pkl]--> 01_NLP_ReviewText_Narrow_3.pkl.gz (295.59 MB)
[*][  pkl]---------> 01_NLP_ReviewText_small.pkl (28.94 MB)
[*][  pkl]---------> 01_NLP_Summary_small.pkl (3.82 MB)
[*][  pkl]---------> 01_NLP_Title_small.pkl (2.73 MB)
---[   gz][  pkl]--> 01_NL_ReviewText_All(new).pkl.gz (593.23 MB)
---[   gz][  pkl]--> 01_NL_ReviewText_All.pkl.gz (592.92 MB)
---[   gz][  pkl]--> 01_NL_ReviewText_textSplit.pkl.gz (15.78 MB)
[*][  pkl]---------> 02_Cellphone.pkl (46.32 MB)
[*][  pkl]---------> 02_NLP_ReviewTextData.pkl (87.00 MB)
[*][  pkl]---------> 02_NLP_SummaryData.pkl (8.32 MB)
[*][  pkl]---------> 02_NLP_TitleData.pkl (16.71 MB)
[*][  pkl]---------> 03_Cellphone.pkl (46.31 MB)
[*][  pkl]---------> 03_NLP_ReviewTextData.pkl (28.94 MB)
[*][  pkl]---------> 03_NLP_ReviewText_Narrow.pkl (17.13 MB)
[*][  pkl]---------> 03_NLP_SummaryData.pkl (3.82 MB)
```

```
[*][  pkl]---------> 03_NLP_TitleData.pkl (2.73 MB)
[*][  pkl]---------> 04_NLP_ReviewText_Narrow.pkl (16.95 MB)
[*][  pkl]---------> 05_NLP_ReviewText_Narrow.pkl (66.15 MB)
[*][  pkl]---------> 05_NLP_ReviewText_Narrow_full.pkl (207.91 MB)

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/03_train [Empty

[D] /content/gdrive/MyDrive/Colab Notebooks/data/ML1010-Group-Project/04_test [Empty
```

```
df = pd.read_csv(jarvis.DATA_DIR + '/complaints.csv.gz')
```

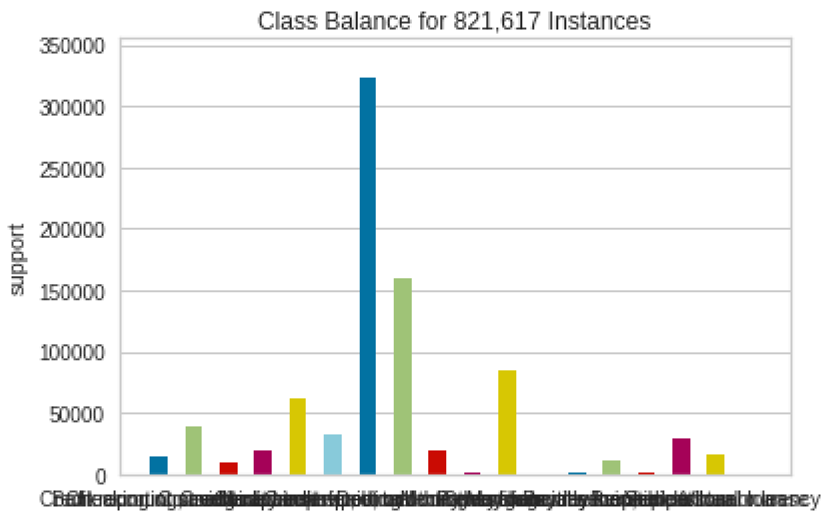
```
mvutils.exploreDataframe(df, numRecords=1)
#df.info(verbose=True)
```

```
dataframe shape: (2355756, 18)

dataframe info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2355756 entries, 0 to 2355755
Data columns (total 18 columns):
 #   Column                        Dtype
---  ------                        -----
 0   Date received                 object
 1   Product                       object
 2   Sub-product                   object
 3   Issue                         object
 4   Sub-issue                     object
 5   Consumer complaint narrative  object
 6   Company public response       object
 7   Company                       object
 8   State                         object
 9   ZIP code                      object
```

```
#delete all null values for narrative
df = df[pd.notnull(df['Consumer complaint narrative'])]
df.reset_index(inplace=True, drop=True)
mvutils.exploreDataframe(df)
```

```
dataframe shape: (821617, 18)

dataframe info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 821617 entries, 0 to 821616
Data columns (total 18 columns):
 #   Column                        Non-Null Count   Dtype 
---  ------                        --------------   ----- 
 0   Date received                 821617 non-null  object
 1   Product                       821617 non-null  object
 2   Sub-product                   769445 non-null  object
 3   Issue                         821617 non-null  object
 4   Sub-issue                     651600 non-null  object
 5   Consumer complaint narrative  821617 non-null  object
 6   Company public response       398420 non-null  object
 7   Company                       821617 non-null  object
 8   State                         817979 non-null  object
 9   ZIP code                      643495 non-null  object
 10  Tags                          131213 non-null  object
 11  Consumer consent provided?    821617 non-null  object
 12  Submitted via                 821617 non-null  object
 13  Date sent to company          821617 non-null  object
 14  Company response to consumer  821616 non-null  object
 15  Timely response?              821617 non-null  object
 16  Consumer disputed?            164062 non-null  object
```

```
mvutils.displayClassBalance(df, 'Product')
```



```
Bottom 1 in dataframe
```

```
df = df.groupby('Product', group_keys=False).apply(lambda x: x.sample(frac=0.01))
```

```
          Product               Issue             complaint      public
```

```
import importlib
importlib.reload(mvutils)
```

```
<module 'mv_python_utils' from '/content/gdrive/MyDrive/Colab Notebooks/utility_files/mv
```

```
col = ['Product', 'Consumer complaint narrative']
df = df[col]
df.columns
```

```
    Index(['Product', 'Consumer complaint narrative'], dtype='object')
```

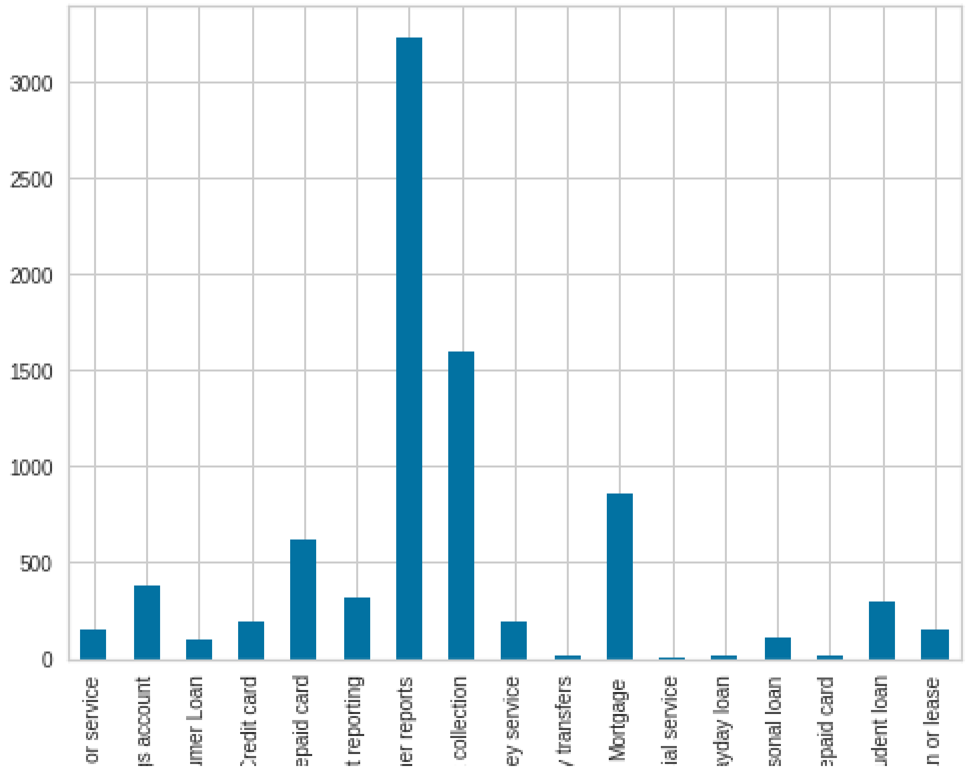
```
df.columns = ['Product', 'Consumer_complaint_narrative']
```

```
df['category_id'] = df['Product'].factorize()[0]
from io import StringIO
category_id_df = df[['Product', 'category_id']].drop_duplicates().sort_values('category_id')
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'Product']].values)
```

```
df.head()
```

	Product	Consumer_complaint_narrative	category_id
593793	Bank account or service	Purchased a Jumbo C.D. from CIT Bank XXXX/XXXX...	0
256852	Bank account or service	I had applied for a CitiGold Checking account ...	0
658921	Bank account or service	RE Case number XXXX On XX/XX/2016, I hired XXX...	0
[illegible]	Bank account or	This is my second complaint against Chase	[illegible]

```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
df.groupby('Product').Consumer_complaint_narrative.count().plot.bar(ylim=0)
plt.show()
```



```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1', ngram_ran

features = tfidf.fit_transform(df.Consumer_complaint_narrative).toarray()
labels = df.category_id
features.shape

     (8215, 22919)
```

edi

```
from sklearn.feature_selection import chi2
import numpy as np

N = 2
for Product, category_id in sorted(category_to_id.items()):
  features_chi2 = chi2(features, labels == category_id)
  indices = np.argsort(features_chi2[0])
  feature_names = np.array(tfidf.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("# '{}':".format(Product))
  print("  . Most correlated unigrams:\n       . {}".format('\n       . '.join(unigrams[-N:])
  print("  . Most correlated bigrams:\n       . {}".format('\n       . '.join(bigrams[-N:])))

     # 'Bank account or service':
       . Most correlated unigrams:
            . citigold
            . overdraft
       . Most correlated bigrams:
            . citigold checking
```

```
          . overdraft fees
      # 'Checking or savings account':
        . Most correlated unigrams:
             . deposit
             . bank
        . Most correlated bigrams:
             . debit card
             . checking account
      # 'Consumer Loan':
        . Most correlated unigrams:
             . car
             . santander
        . Most correlated bigrams:
             . motor finance
             . months loan
      # 'Credit card':
        . Most correlated unigrams:
             . amex
             . card
        . Most correlated bigrams:
             . macy credit
             . credit card
      # 'Credit card or prepaid card':
        . Most correlated unigrams:
             . merchant
             . card
        . Most correlated bigrams:
             . american express
             . credit card
      # 'Credit reporting':
        . Most correlated unigrams:
             . experian
             . equifax
        . Most correlated bigrams:
             . disputed equifax
             . equifax ignored
      # 'Credit reporting, credit repair services, or other personal consumer reports':
        . Most correlated unigrams:
             . inquiries
             . report
        . Most correlated bigrams:
             . identity theft
             . credit report
      # 'Debt collection':
        . Most correlated unigrams:
             . collection
             . debt
        . Most correlated bigrams:
             . debt collection
             . collect debt
      # 'Money transfer, virtual currency, or money service':
        . Most correlated unigrams:
```

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB

X_train, X_test, y_train, y_test = train_test_split(df['Consumer_complaint_narrative'], df['P
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

clf = MultinomialNB().fit(X_train_tfidf, y_train)
```

```
print(clf.predict(count_vect.transform(["This company refuses to provide me verification and
```

```
['Credit reporting, credit repair services, or other personal consumer reports']
```

```
#I trimmed data and used a subset so it doesn't find narrative in the dataframe
df[df['Consumer_complaint_narrative'] == "This company refuses to provide me verification and
```

Product	Consumer_complaint_narrative	category_id

```
df.head()
```

	Product	Consumer_complaint_narrative	category_id
593793	Bank account or service	Purchased a Jumbo C.D. from CIT Bank XXXX/XXXX...	0
256852	Bank account or service	I had applied for a CitiGold Checking account ...	0
658921	Bank account or service	RE Case number XXXX On XX/XX/2016, I hired XXX...	0
[illegible]	Bank account or	This is my second complaint against Chase	[illegible]

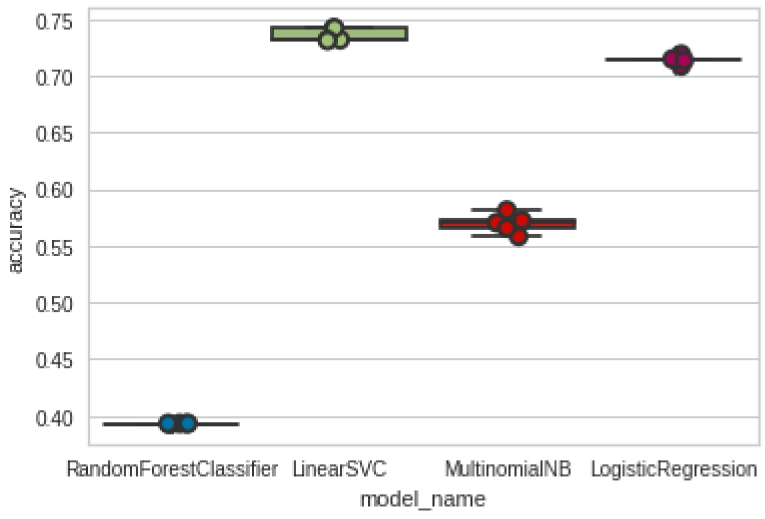
```
print(clf.predict(count_vect.transform(["I am disputing the inaccurate information the Chex-S
```

```
['Credit reporting, credit repair services, or other personal consumer reports']
```

```
#I trimmed data and used a subset so it doesn't find narrative in the dataframe
df[df['Consumer_complaint_narrative'] == "I am disputing the inaccurate information the Chex-
```

Product	Consumer_complaint_narrative	category_id

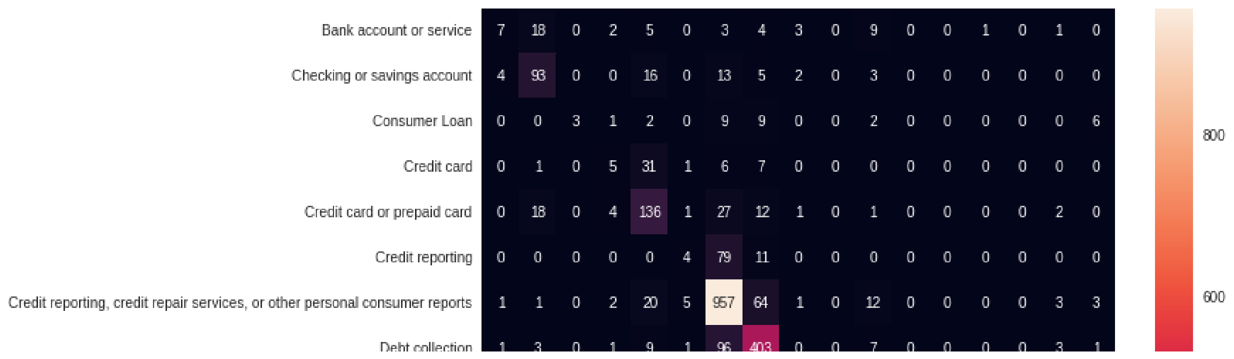
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score


models = [
    RandomForestClassifier(n_estimators=200, max_depth=3, random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),
]
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'])


import seaborn as sns
sns.boxplot(x='model_name', y='accuracy', data=cv_df)
sns.stripplot(x='model_name', y='accuracy', data=cv_df,
              size=8, jitter=True, edgecolor="gray", linewidth=2)
plt.show()
```



```
cv_df.groupby('model_name').accuracy.mean()
```

```
     model_name
     LinearSVC                 0.735606
     LogisticRegression        0.713938
     MultinomialNB             0.569446
     RandomForestClassifier    0.393061
     Name: accuracy, dtype: float64
```

```
model = LinearSVC()
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, la
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
from sklearn.metrics import confusion_matrix
conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=category_id_df.Product.values, yticklabels=category_id_df.Product.val
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



```
category_id_df.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 17 entries, 593793 to 286016
    Data columns (total 2 columns):
     #   Column       Non-Null Count  Dtype 
    ---  ------       --------------  ----- 
     0   Product      17 non-null     object
     1   category_id  17 non-null     int64 
    dtypes: int64(1), object(1)
    memory usage: 408.0+ bytes
```

```
from IPython.display import display

for predicted in category_id_df.category_id:
  for actual in category_id_df.category_id:
    if predicted != actual and conf_mat[actual, predicted] >= 6:
      print("'{}' predicted as '{}' : {} examples.".format(id_to_category[actual], id_to_cate
      display(df.loc[indices_test[(y_test == actual) & (y_pred == predicted)]][['Product', 'C
      print('')
```

```
'Bank account or service' predicted as 'Checking or savings account' : 18 examples
```

	Product	Consumer_complaint_narrative
525836	Bank account or service	checking account was over-drafted. I had, acco...
646988	Bank account or service	Dear Sir, On Friday, XX/XX/XXXX my wallet (CC...
563293	Bank account or service	My boyfriend and I have an account with US Ban...
659868	Bank account or service	XXXX separate Wells Fargo Banks in XXXX, Texas...
652467	Bank account or service	My checking account with PNC Bank had a low ba...
659567	Bank account or service	I got a job offer on XXXX for a administrative...
594970	Bank account or service	Citibank advertised the promotional bonus in X...
584175	Bank account or service	I 'm writing in regards to our experience with...
687637	Bank account or service	I had slightly more than {$1700.00} in my Bank...
624464	Bank account or service	I have been a customer with Bank of America fo...
641811	Bank account or service	At approximately XXXX on XXXX XXXX I went to m...
630554	Bank account or service	I was called out of state for a family emergen...
564049	Bank account or service	I had several accounts at Wells Fargo From XX/...
618502	Bank account or service	I deposited {$7000.00} into my checking accoun...
629807	Bank account or service	In 2011 my debit card was lost/stolen I had ju...
646152	Bank account or service	Ally Bank refused to transfer a matured CD IRA...
575978	Bank account or service	I had XXXX charges that cleared last week, the...
693403	Bank account or service	The bank honored a check that was dated a mont...

```
model.fit(features, labels)
N = 2
for Product, category_id in sorted(category_to_id.items()):
  indices = np.argsort(model.coef_[category_id])
  feature_names = np.array(tfidf.get_feature_names())[indices]
  unigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 1][:N]
  bigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 2][:N]
  print("# '{}':".format(Product))
  print("  . Top unigrams:\n       . {}".format('\n       . '.join(unigrams)))
  print("  . Top bigrams:\n       . {}".format('\n       . '.join(bigrams)))
```

```
# 'Bank account or service':
  . Top unigrams:
       . bank
       . referring
  . Top bigrams:
       . met requirements
       . xxxx 15
# 'Checking or savings account':
```

```
          . Top unigrams:
               . bank
               . branch
          . Top bigrams:
               . overdraft fee
               . debit card
        # 'Consumer Loan':
          . Top unigrams:
               . santander
               . car
          . Top bigrams:
               . months loan
               . new car
        # 'Credit card':
          . Top unigrams:
               . card
               . amex
          . Top bigrams:
               . credit card
               . closed xxxx
        # 'Credit card or prepaid card':
          . Top unigrams:
               . card
               . discover
          . Top bigrams:
               . credit limit
               . use card
        # 'Credit reporting':
          . Top unigrams:
               . equifax
               . experian
          . Top bigrams:
               . xxxx accounts
               . xxxx contract
        # 'Credit reporting, credit repair services, or other personal consumer reports':
          . Top unigrams:
               . experian
               . report
          . Top bigrams:
               . xxxx xxxx
               . disputed xxxx
        # 'Debt collection':
          . Top unigrams:
               . debt
               . collection
          . Top bigrams:
               . company account
               . xxxx filed
        # 'Money transfer, virtual currency, or money service':
```

```
from sklearn import metrics
print(metrics.classification_report(y_test, y_pred, target_names=df['Product'].unique()))
```

```
                                                                                           precision
```

```
                                                     Bank account or service       0.50
                                                 Checking or savings account       0.60
                                                               Consumer Loan       1.00
                                                                 Credit card       0.31
                                                 Credit card or prepaid card       0.58
                                                            Credit reporting       0.29
Credit reporting, credit repair services, or other personal consumer reports       0.76
                                                             Debt collection       0.71
                          Money transfer, virtual currency, or money service       0.77
                                                             Money transfers       0.00
                                                                    Mortgage       0.86
                                                     Other financial service       0.00
                                                                 Payday loan       1.00
                                   Payday loan, title loan, or personal loan       0.80
                                                                Prepaid card       0.00
                                                                Student loan       0.80
                                                       Vehicle loan or lease       0.52

                                                                    accuracy
                                                                   macro avg       0.56
                                                                weighted avg       0.71
```

80777	Checking or savings account	Hi earlier this month, I paid a 5 dollar fee t...
723665	Checking or savings account	I was banking with Wellsfargo my account was c...
504580	Checking or savings account	On XXXX/XXXX/XXXX I reserved a car through XXX...
442567	Checking or savings account	My name is XXXX XXXX and actually this is abou...
220426	Checking or savings account	I get my child support from XXXX on a Direct E...
281422	Checking or savings account	Hello, today I spoke an agent via 5th 3rd bank...
258399	Checking or savings account	On, XX/XX/XXXX all my XXXX transactions where ...
235295	Checking or savings account	I opened a CD account with Marcus Bank. I rece...
306299	Checking or savings account	We have over 90 unauthorized chargers on our U...
307247	Checking or savings account	Dear CFPB, today I received a letter in the US...
63470	Checking or savings account	On XX/XX/2020 my wallet was stolen and the per...

```
'Credit card' predicted as 'Credit card or prepaid card' : 31 examples.
```

	Product	Consumer_complaint_narrative