Data Analytics Report

Consumer Complaints Free Text Classification

1. **Executive Summary**

This analytics is a machine-learning project. It uses Python scikit-learn, an open source machine learning library, to classify “Consumer Complaints” data (publicly-available open dataset). The result can be used to automatically classify any new consumer complaints or similar free text.

1. **Introduction**

(Note to Coursera reviewer: This project does not use Foursquare data, due to the fact that in my current job, we do not deal with location data. Rather, we deal with great amount of free texts. Therefore, I would like to use this opportunity to develop an analytical tool that can be leveraged in my daily job. Thank you for your consideration.)

It is estimated that the majority (80% ~ 90%) of current data is un-structured data; that is, there is no pre-defined hierarchy of the data structure and data format. Such un-structured data include emails, text messages, reviews and comments, and other free texts. These data are of great value, since they can be used to provide deep insights and to reveal covered facts that are critical to the business. However, they are highly challenging to be processed for further analysis.

This project aims to develop a preliminary tool to interpret and classify free text. It reads “Consumer Complaints” data, a publicly-available open dataset, undergoes data cleaning process (including removal of stop words, tokenizing and lemmatizing), and uses part of the datasets as the training dataset. Then it will go through the training process to find typical keywords for each Complaint Category (termed as “Product” in the original data source). In the testing process, the tool will evaluate the keywords in the new data entry and classify it into an appropriate category.

1. **Data Source**

The data is from data.world (<https://data.world/data-society/consumer-complaint-data> ). See snapshot below.

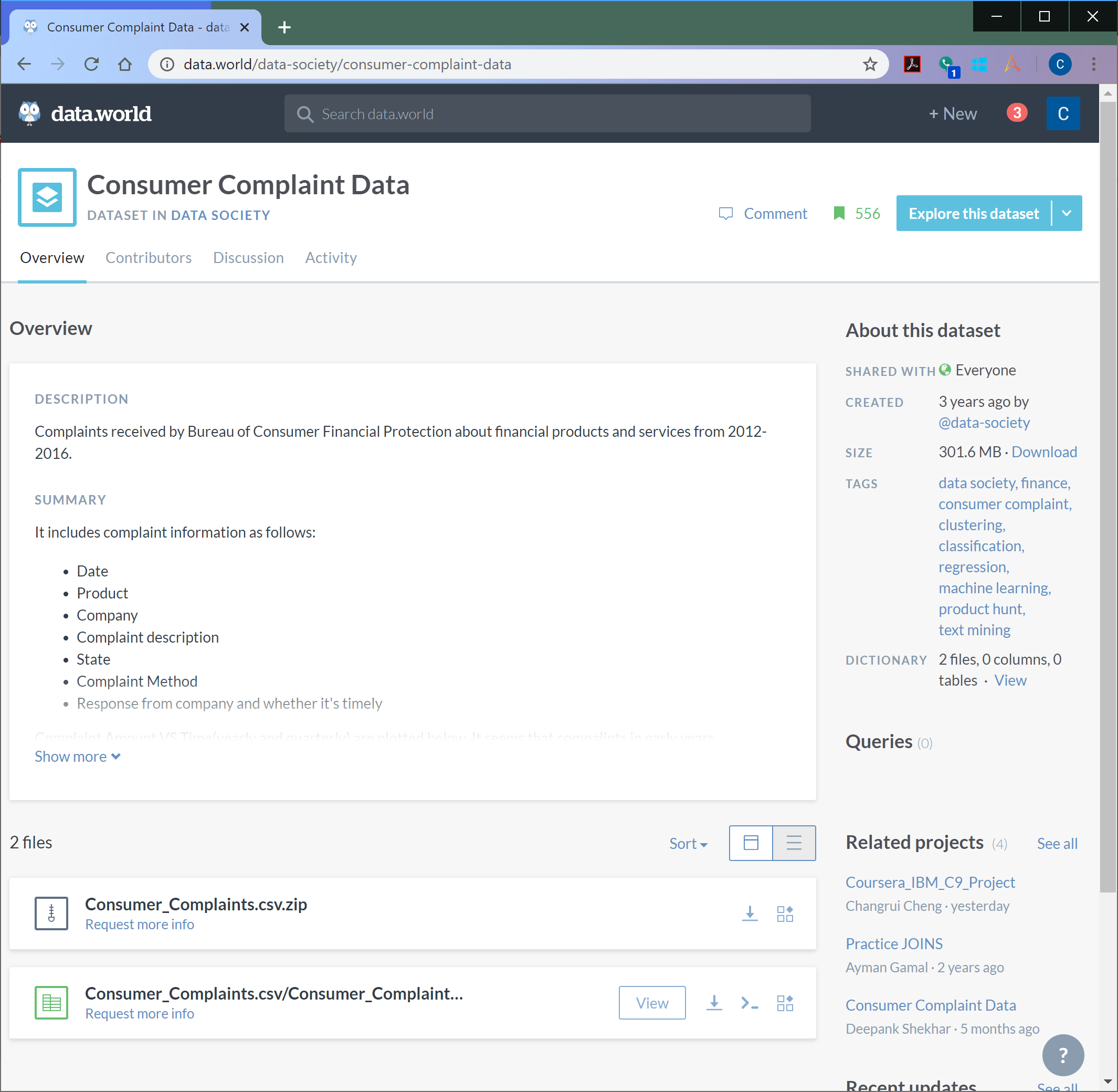


Figure 1. Data.world snapshot of Consumer Complaints data

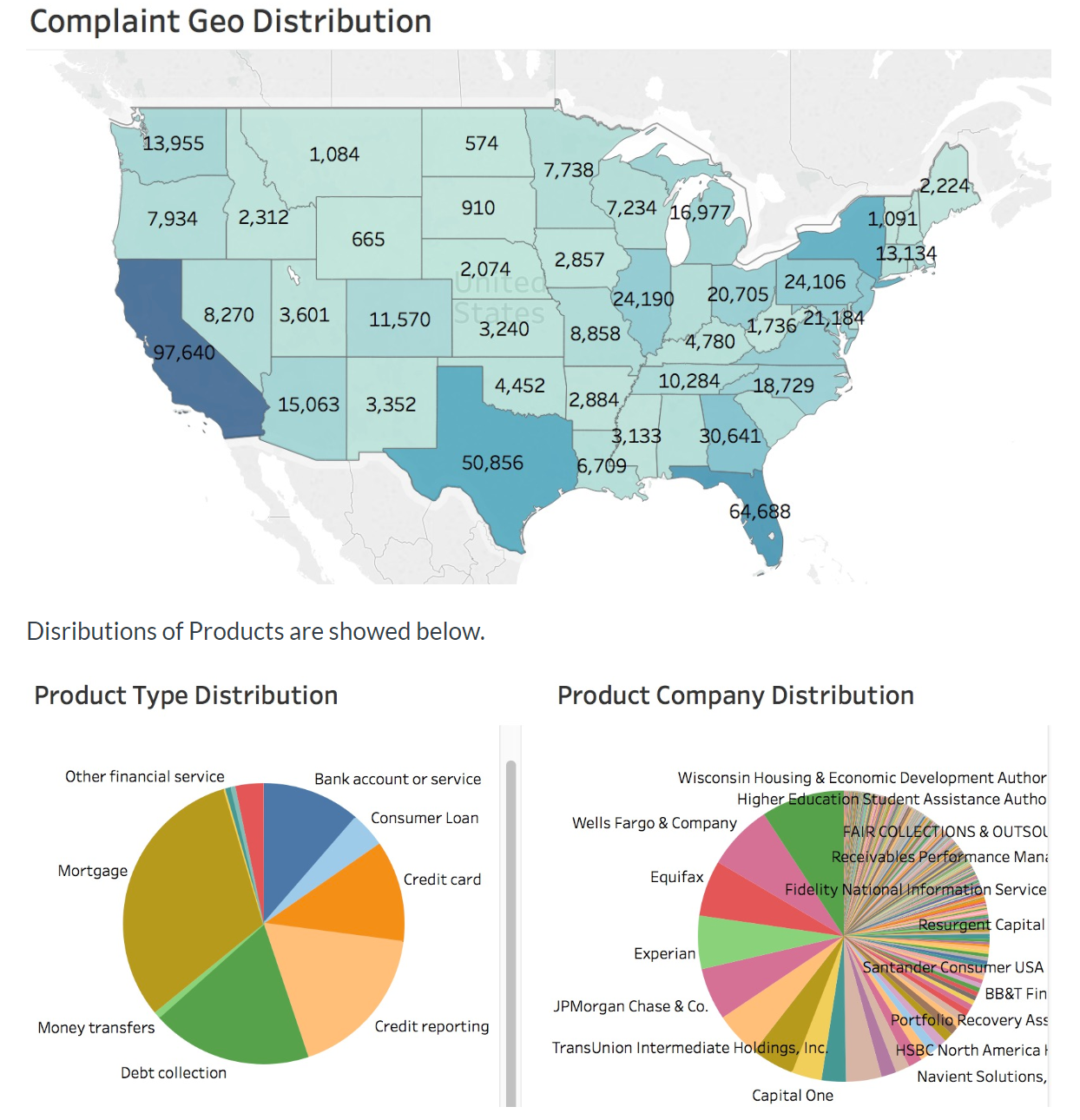


Figure 2. Data.world snapshot of Consumer Complaints data, Plots

The original data can be found at Consumer Financial Protection Bureau’s website (<https://data.consumerfinance.gov/dataset/Consumer-Complaints/s6ew-h6mp> ). See snapshot below. It also contains a brief description of the data column.

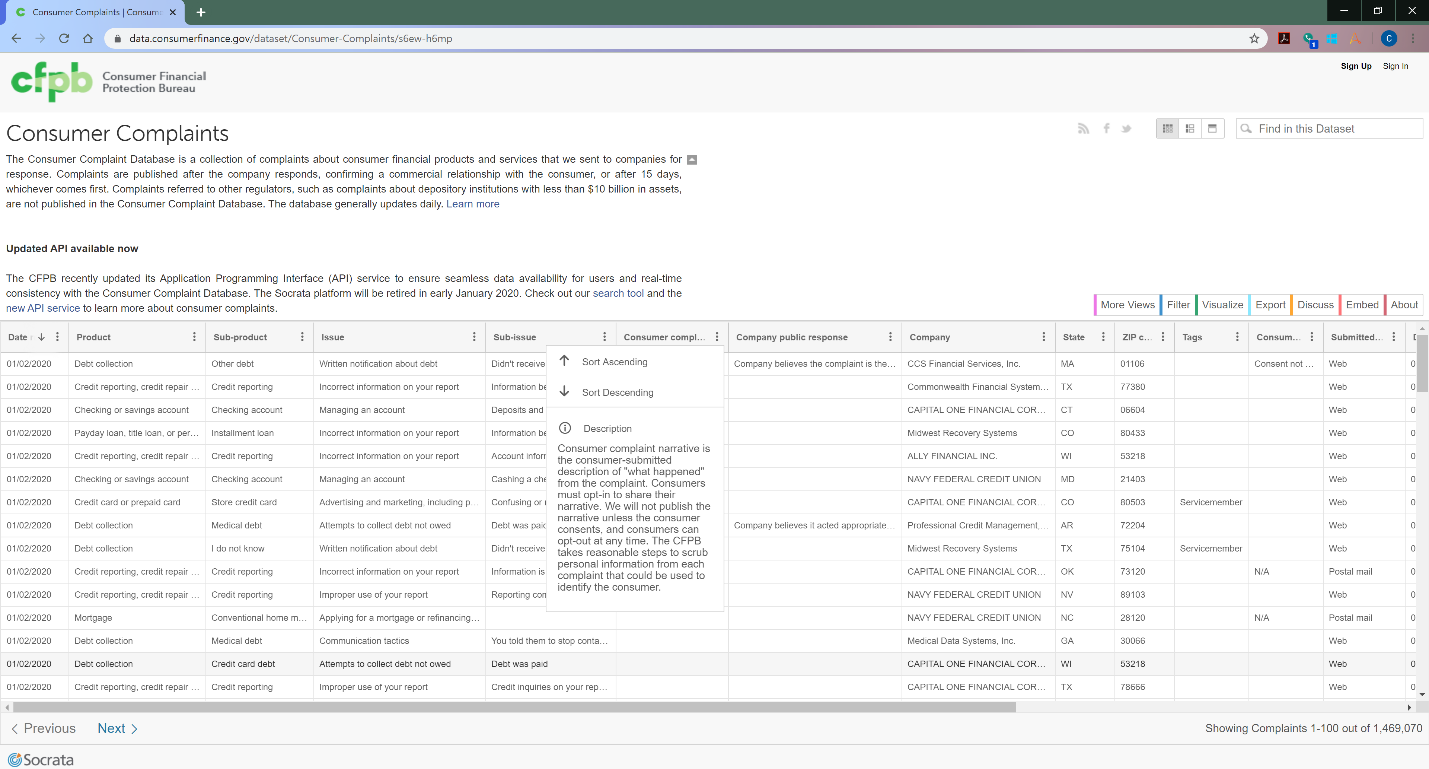
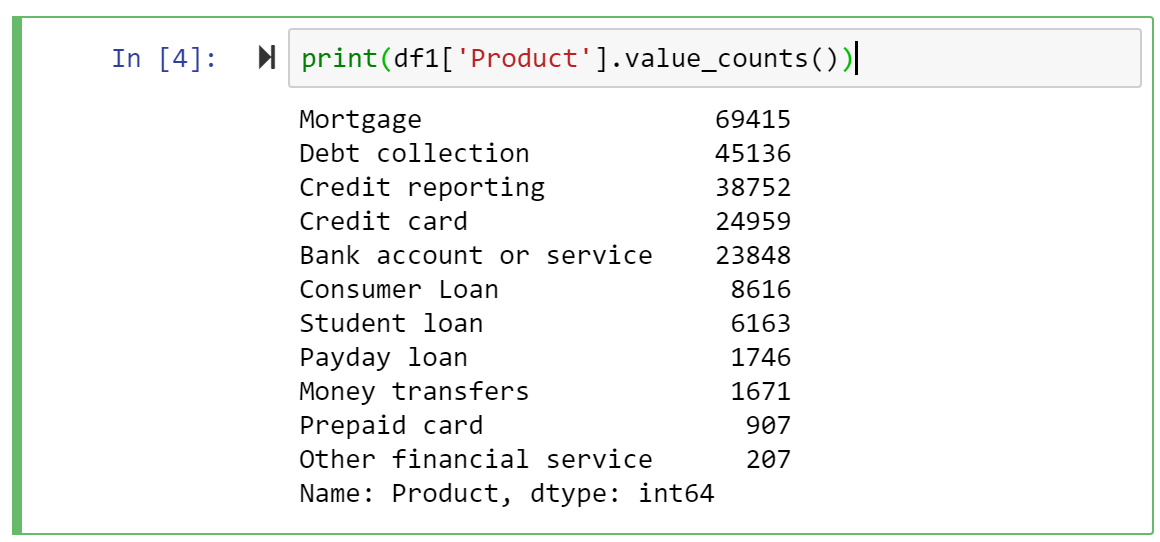


Figure 3. Original data source: Consumer Complaint Database, from Consumer Financial Protection Bureau

This project will only use two data fields (columns): “Product” (the type of products or services); and “”. All unique values in “Product” is below:



1. **Data Preparation**

This includes:

. Understanding date range :

df1['Date received'] = pd.to\_datetime(df1['Date received']) # Convert to Date

print('Data from: ', df1['Date received'].min(), 'to:', df1['Date received'].max()) # (2011-12-01 - 2016-06-28 )

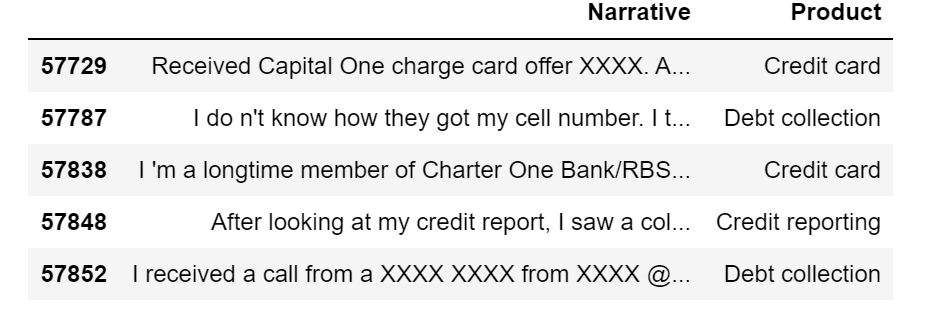
. Rename data column:

df1.rename(columns = {'Consumer complaint narrative':'Narrative'}, inplace = True)

df1 = df1[['Narrative','Product']]

df1 = df1[pd.notnull(df1['Narrative'])]

After that, the dataset looks like:



import matplotlib.pyplot as plt

import seaborn as sns

product\_count = df1['Product'].value\_counts()

plt.figure(figsize=(12,4))

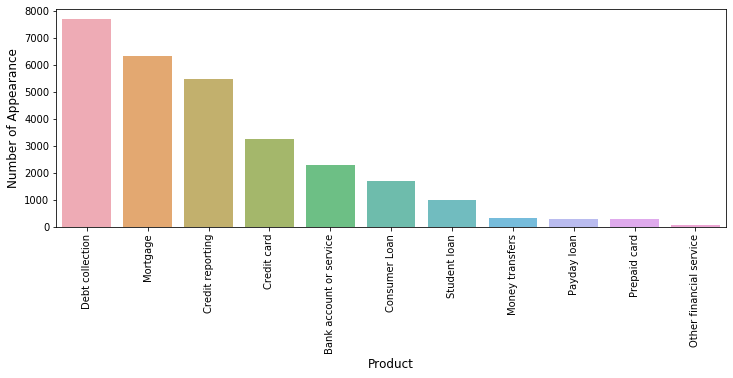
sns.barplot(product\_count.index, product\_count.values, alpha=0.8)

plt.ylabel('Number of Appearance', fontsize=12)

plt.xlabel('Product', fontsize=12)

plt.xticks(rotation=90)

plt.show();



. Remove Stop Words; Tokenize and Lemmatize words; remove "XX":

stopWords = set(nltk.corpus.stopwords.words('english'))

allStops = stopWords | set(string.punctuation) | set(('•','-'))

tokenizer = nltk.TweetTokenizer()

lemmatizer = WordNetLemmatizer()

df1['Narrative\_simple'] = ""

for index, row in df1.iterrows():

tokens = tokenizer.tokenize(row['Narrative'])

tokens = [lemmatizer.lemmatize(token) for token in tokens]

tokens = [t.lower() for t in tokens if t.lower() not in allStops and len(t) >= 2]

words = ""

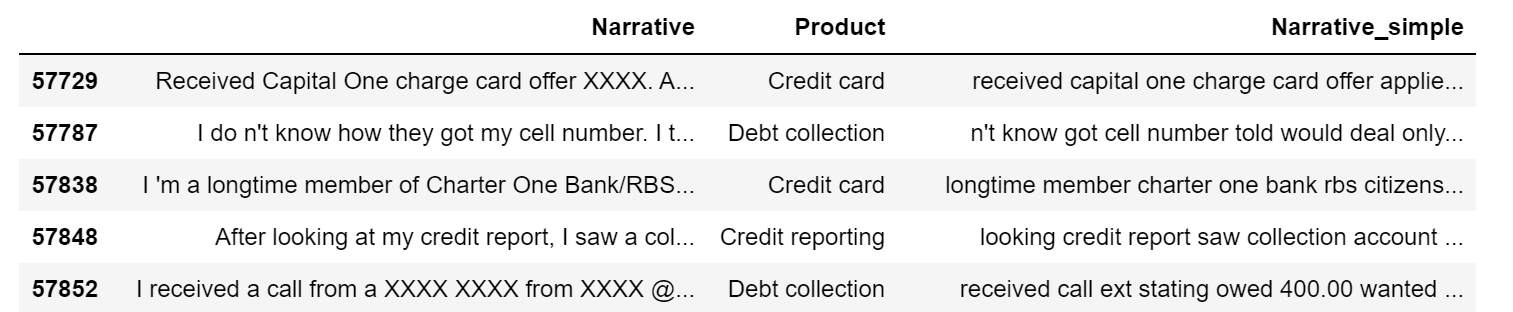
for w in tokens:

if w.find('xx')<0:

words += " " + w

row['Narrative\_simple'] = words

df1.head(20)



Now the data is ready for analysis.

1. **Analytics Methodology**

First, TFIDF is used to find tf-idf vector for each record.

from sklearn.feature\_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(sublinear\_tf=True, min\_df=5, norm='l2', encoding='latin-1', ngram\_range=(1,3), stop\_words='english')

features = tfidf.fit\_transform(df1['Narrative\_simple'])

labels = df1['Product']

features.shape #(28629, 91639)

Then the “Product” column is re-index for further analysis:

df\_product = df1['Product'].drop\_duplicates().reset\_index()

df\_product.reset\_index(drop=True)

Next, the tool will use Scikit-Learn’s Chi2 to find the feature vector for each record, and try to identify the most frequent uni-gram, bi-gram and tri-gram:

from sklearn.feature\_selection import chi2

N = 5

for Product in df\_product['Product']:

features\_chi2 = chi2(features, labels == Product)

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]

trigrams = [v for v in feature\_names if len(v.split(' ')) == 3]

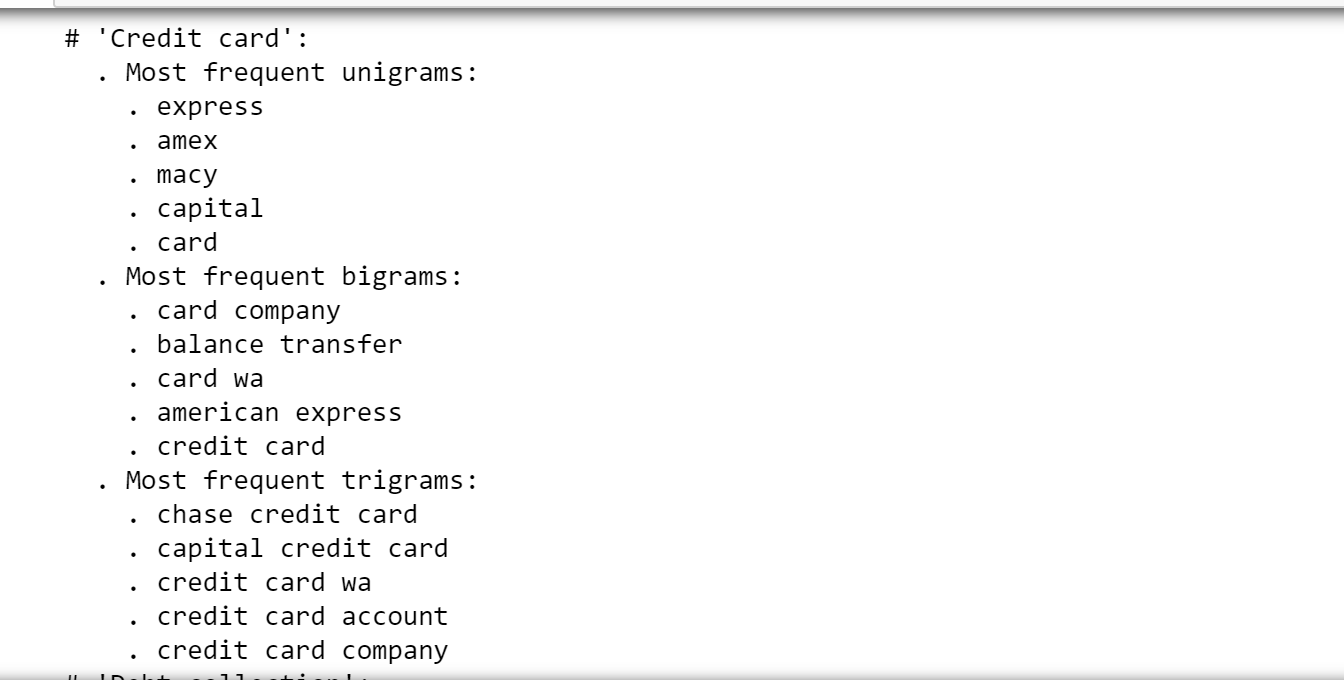
print("# '{}':".format(Product))

print(" . Most frequent unigrams:\n . {}".format('\n . '.join(unigrams[-N:])))

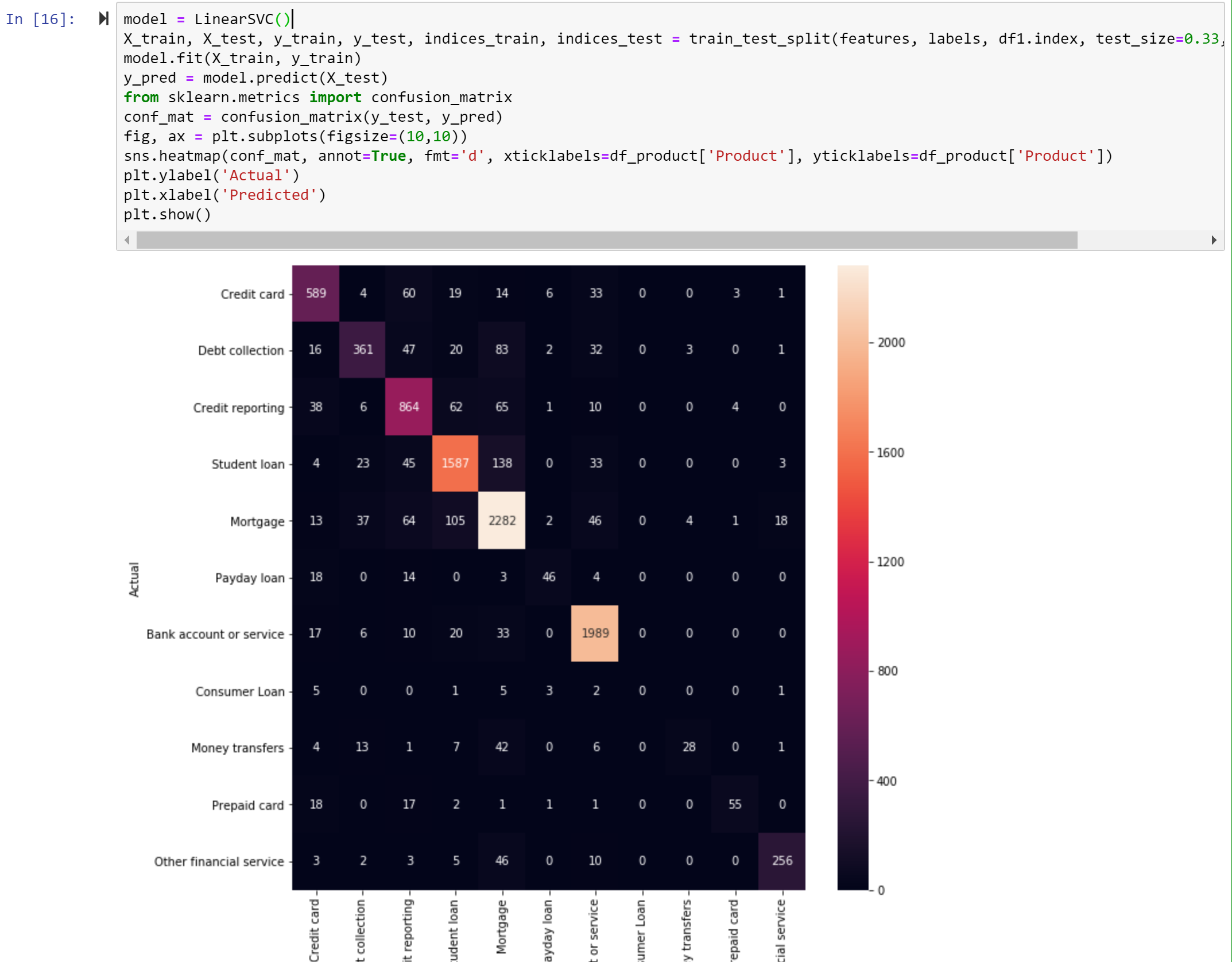
print(" . Most frequent bigrams:\n . {}".format('\n . '.join(bigrams[-N:])))

print(" . Most frequent trigrams:\n . {}".format('\n . '.join(trigrams[-N:])))

so the outcome is below:

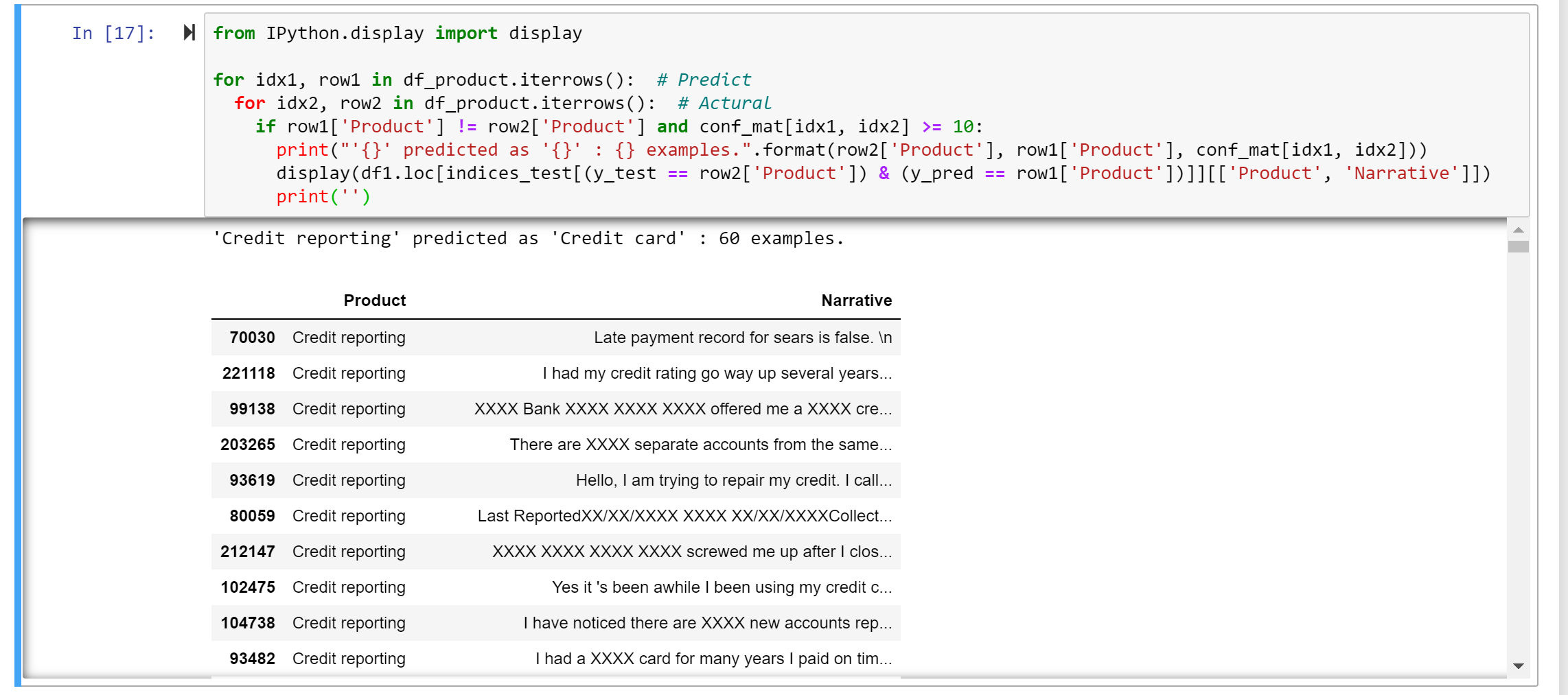


Then the tool uses Scikit-Learn’s LinearSVC to first train the model. After that, the trained model is used to predict the test model, and the prediction results is used to compare with the original test dataset, as shown below.

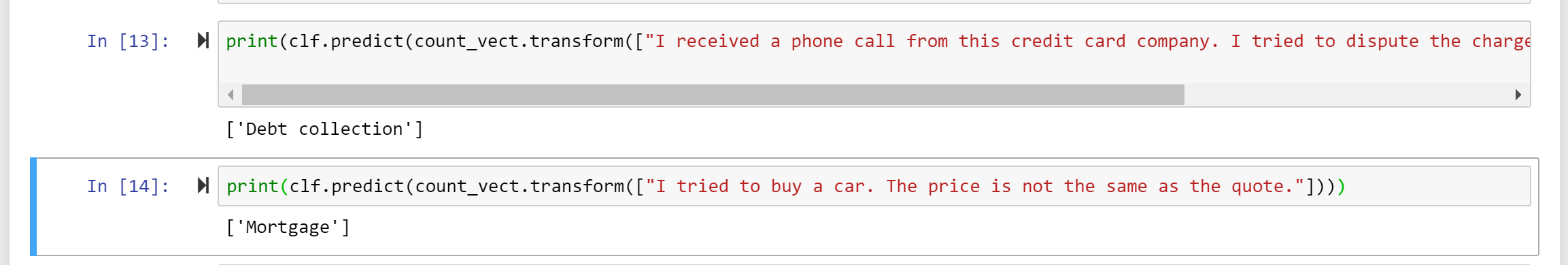


1. **Result**

It is seen that some test dataset records were mis-categorized in the original data source:



This tool can be used to classify any consumer complaints, such as:



1. **Conclusion**

This simple tool is developed using machine learning technology based on Python’s Scikit-Learn library, and it can be used to classify any free text similar to consumer complaints or comments. It will be done by the tool automatically, therefore, it will greatly reduce the work load of human interpretation and improve the accuracy.