ANISH ADNANI

D17B / 01

EXPERIMENT 9

Aim: Implement multi-class classifier for Text categorization for various news articles or websites to categorize based on news title

Theory:

Text classification also known as text tagging or text categorization is the process of categorizing text into organized groups by using natural language processing (NLP).
Text classifiers can automatically analyze text and then assign a set of predefined tags or categories based on its content

A free-text document is typically represented as a feature vector, where feature values typically encode the presence of words, word n-grams, syntactically or semantically tagged phrases, named entities, etc in the document.

It is useful to differentiate text classification problems by number of classes a document can belong to. If they are exactly 2 classes Ex: spam/no spam this is called a binary text classification problem. If there are more than 2 classes and each document falls into exactly one class, this is 'multi-class' problem.

In many cases, however a document may have more than one associated category in a classification scheme. For example, a journal article could belong to computational biology, machine learning and some sub-domains in other categories. This type of classification is called a multi-label categorization problem.

Multi-label and multi-class tasks are often handled by reducing them to k binary classification tasks, one for each category.
For each such binary classification tasks, members of the respective category are defined as positive examples, while all others are designated as negative examples.

Application and usecases:

1) Tagging content or products using categories as a way to improve browsing or to identify related content on website platforms such as e-commerce, news agencies, content creators blogs and likes can use automated technologies to classify and tag content and products.

2) Text classification can also be used to automate CRM tasks - The text classifier is highly customizable and can be trained accordingly.

3) Text classification of content on the website using tags helps google crawl the website easily which ultimately helps in SEO. Additionaly automating the content has on website and app can make user experience better and helps to standardize them.

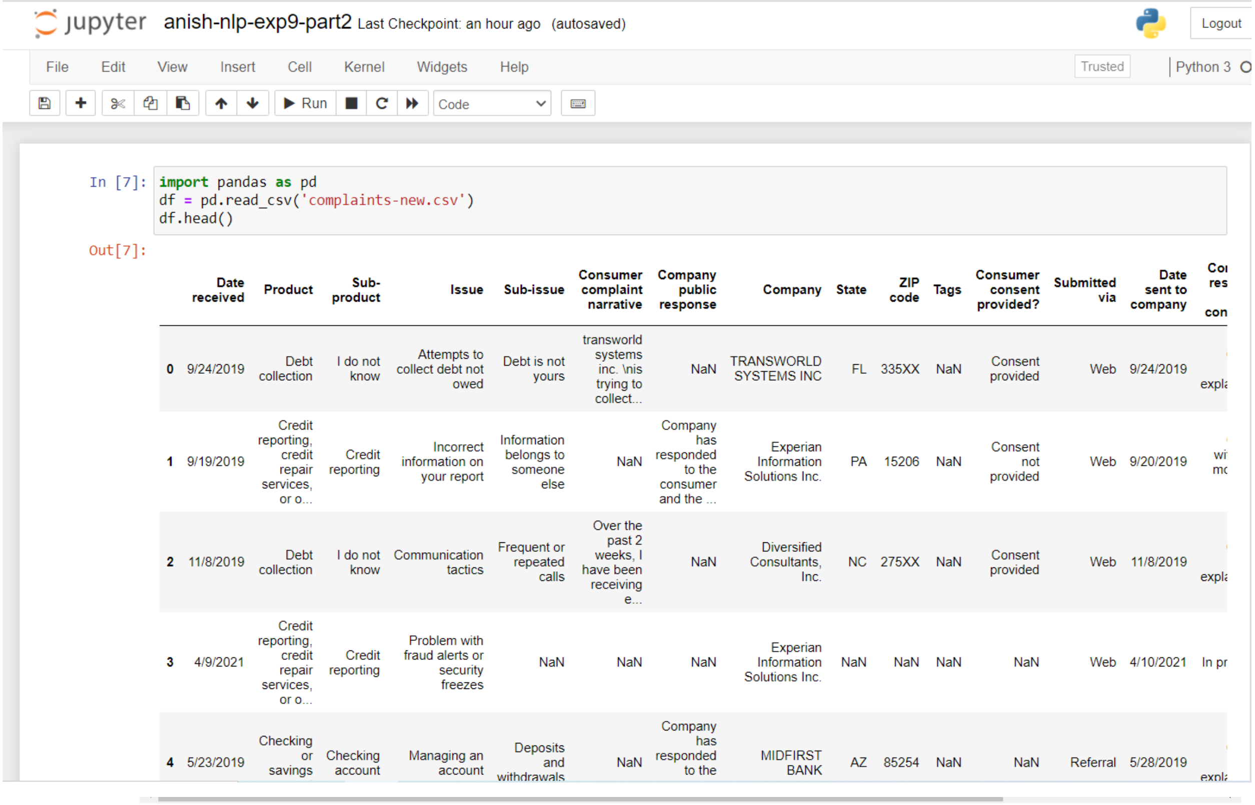
4) As marketing is becoming more targeted everyday, automated classification of users into cohorts can make marketer's life simple. marketers can monitor and classify users based on how they talk about a product or brand online.

5) Academia, law practioners, social researchers, government, and non-profit organisations can also make use of text classification technology. As these organisations deal with a lot of unstructured text, handling the data would be much easier if it is standardized by categories/ tags.

Conclusion:

We successfully implemented multiclass text classifier for text categorization

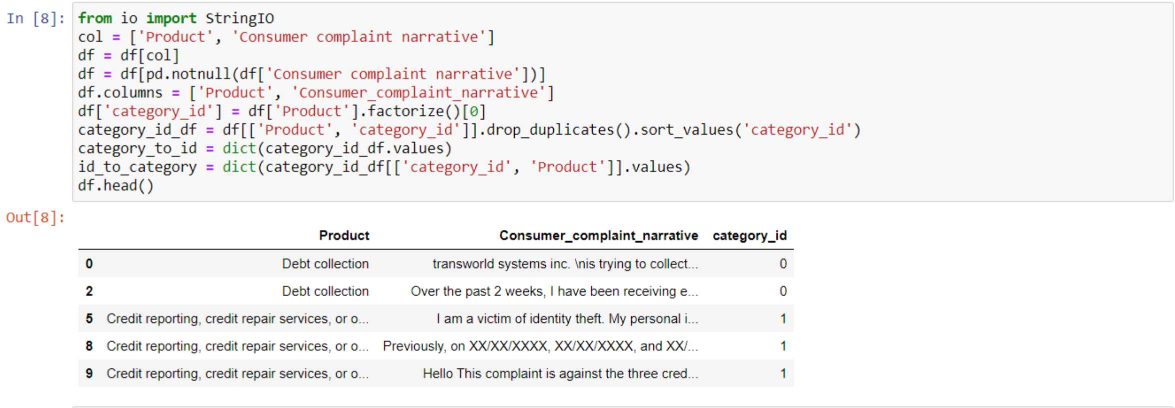
Multi-Class Text Classification with Scikit-Learn



In [7]:
```
import pandas as pd
df = pd.read_csv('complaints-new.csv')
df.head()
```

Out[7]:

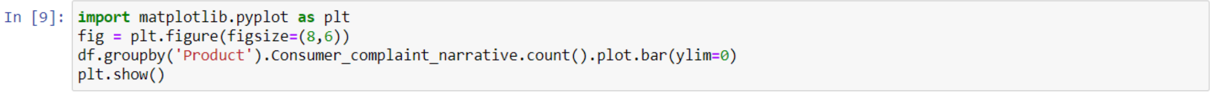
	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags	Consumer consent provided?	Submitted via	Date sent to company	Co res con
0	9/24/2019	Debt collection	I do not know	Attempts to collect debt not owed	Debt is not yours	transworld systems inc. \nis trying to collect...	NaN	TRANSWORLD SYSTEMS INC	FL	335XX	NaN	Consent provided	Web	9/24/2019	expl
1	9/19/2019	Credit reporting, credit repair services, or o...	Credit reporting	Incorrect information on your report	Information belongs to someone else	NaN	Company has responded to the consumer and the ...	Experian Information Solutions Inc.	PA	15206	NaN	Consent not provided	Web	9/20/2019	wi mc
2	11/8/2019	Debt collection	I do not know	Communication tactics	Frequent or repeated calls	Over the past 2 weeks, I have been receiving e...	NaN	Diversified Consultants, Inc.	NC	275XX	NaN	Consent provided	Web	11/8/2019	expl
3	4/9/2021	Credit reporting, credit repair services, or o...	Credit reporting	Problem with fraud alerts or security freezes	NaN	NaN	NaN	Experian Information Solutions Inc.	NaN	NaN	NaN	NaN	Web	4/10/2021	In pr
4	5/23/2019	Checking or savings	Checking account	Managing an account	Deposits and withdrawals	NaN	Company has responded to the	MIDFIRST BANK	AZ	85254	NaN	NaN	Referral	5/28/2019	expl



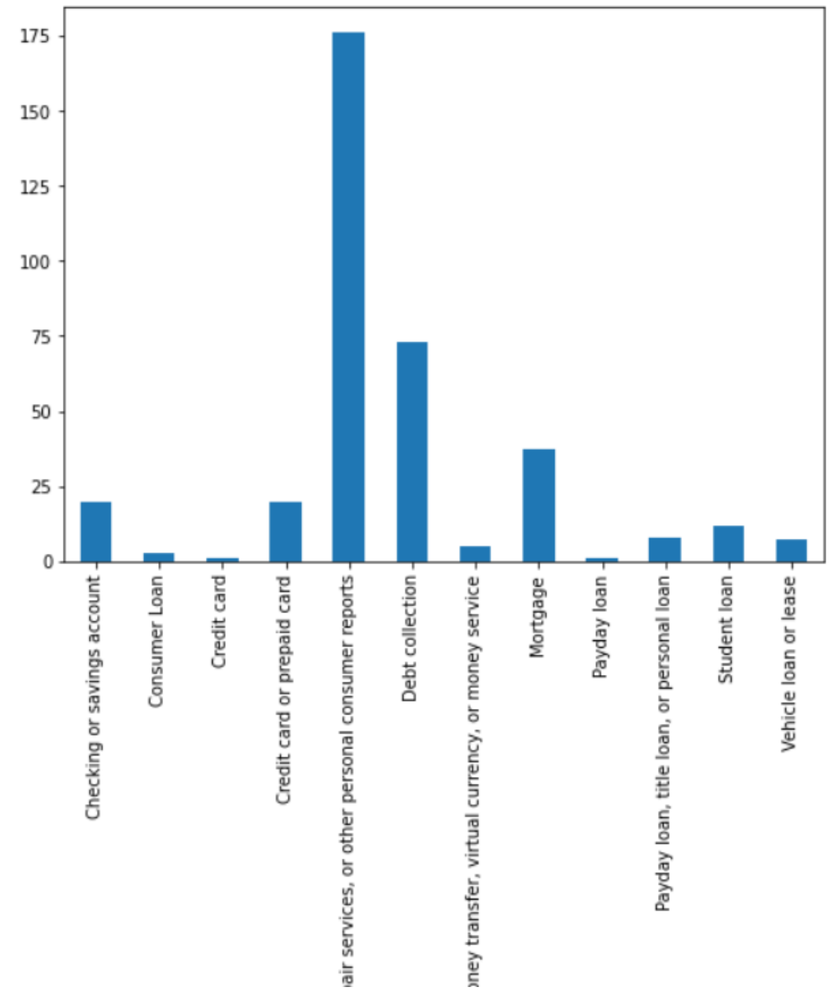
In [8]:
```
from io import StringIO
col = ['Product', 'Consumer complaint narrative']
df = df[col]
df = df[pd.notnull(df['Consumer complaint narrative'])]
df.columns = ['Product', 'Consumer_complaint_narrative']
df['category_id'] = df['Product'].factorize()[0]
category_id_df = df[['Product', 'category_id']].drop_duplicates().sort_values('category_id')
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'Product']].values)
df.head()
```

Out[8]:

	Product	Consumer_complaint_narrative	category_id
0	Debt collection	transworld systems inc. \nis trying to collect...	0
2	Debt collection	Over the past 2 weeks, I have been receiving e...	0
5	Credit reporting, credit repair services, or o...	I am a victim of identity theft. My personal i...	1
8	Credit reporting, credit repair services, or o...	Previously, on XX/XX/XXXX, XX/XX/XXXX, and XX/...	1
9	Credit reporting, credit repair services, or o...	Hello This complaint is against the three cred...	1



In [9]:
```
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,6))
df.groupby('Product').Consumer_complaint_narrative.count().plot.bar(ylim=0)
plt.show()
```



```
In [10]: from sklearn.feature_extraction.text import TfidfVectorizer
         tfidf = TfidfVectorizer(sublinear_tf=True, min_df=5, norm='l2', encoding='latin-1', ngram_range=(1, 2), stop_words='english')
         features = tfidf.fit_transform(df.Consumer_complaint_narrative).toarray()
         labels = df.category_id
         features.shape

Out[10]: (363, 1468)
```

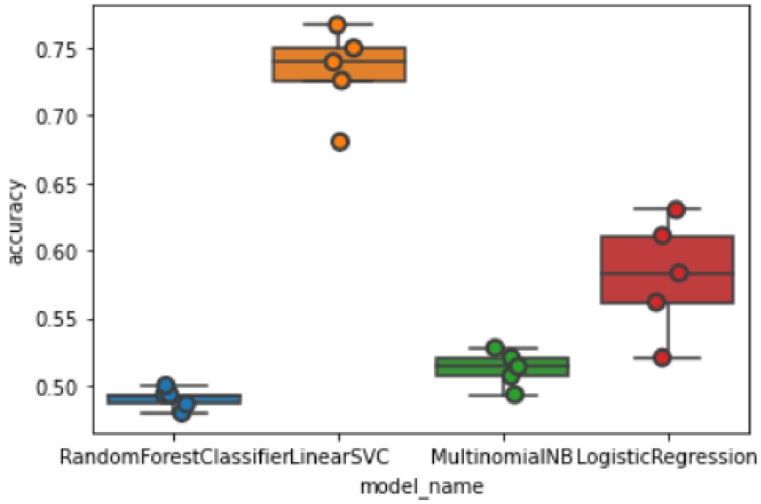
```
In [11]: from sklearn.model_selection import train_test_split
         from sklearn.feature_extraction.text import CountVectorizer
         from sklearn.feature_extraction.text import TfidfTransformer
         from sklearn.naive_bayes import MultinomialNB
         X_train, X_test, y_train, y_test = train_test_split(df['Consumer_complaint_narrative'], df['Product'], random_state = 0)
         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)
```

```
In [12]: refuses to provide me verification and validation of debt per my right under the FDCPA. I do not believe this debt is mine."]))

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

```
In [14]: ion now and they would n't have to research once again. I would like the reported information to be removed : XXXX XXXX XXXX"]))

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

In [16]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
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()
```



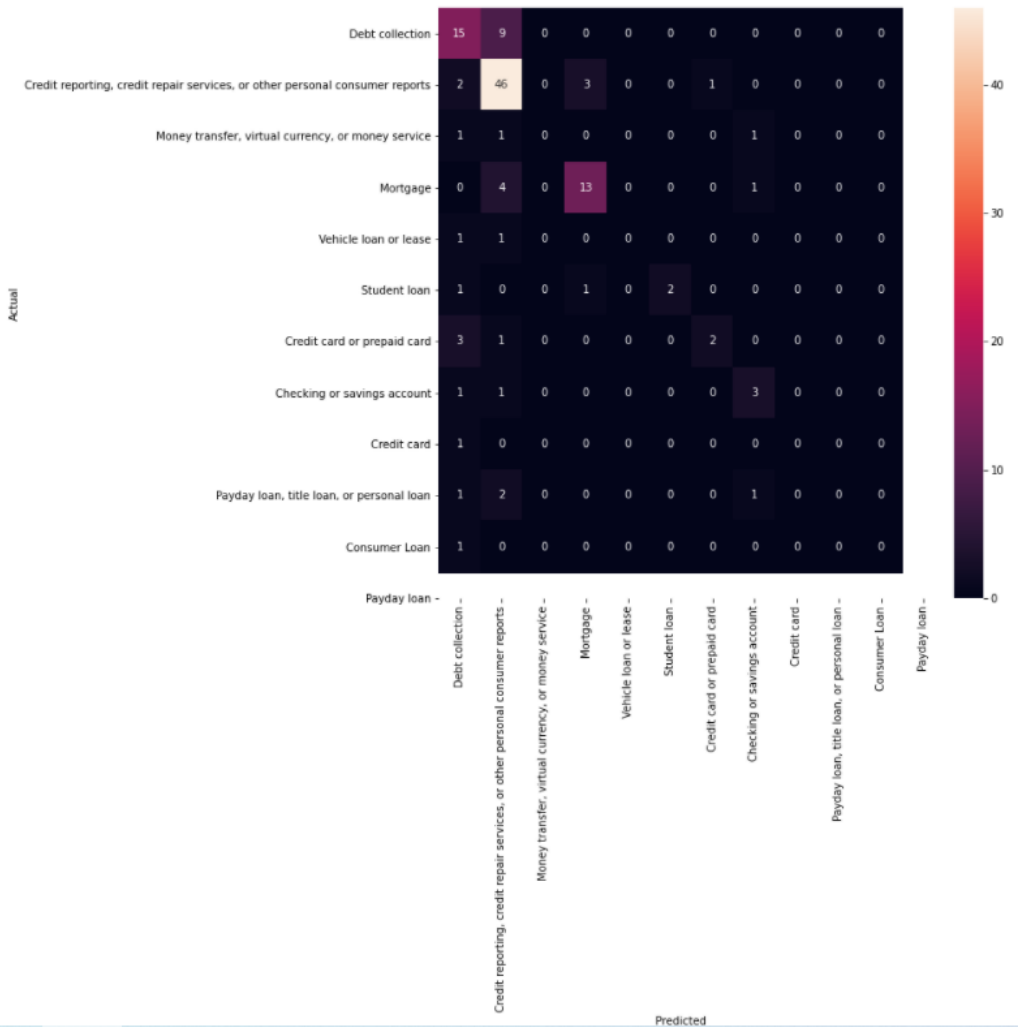
In [17]:

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

Out[17]:

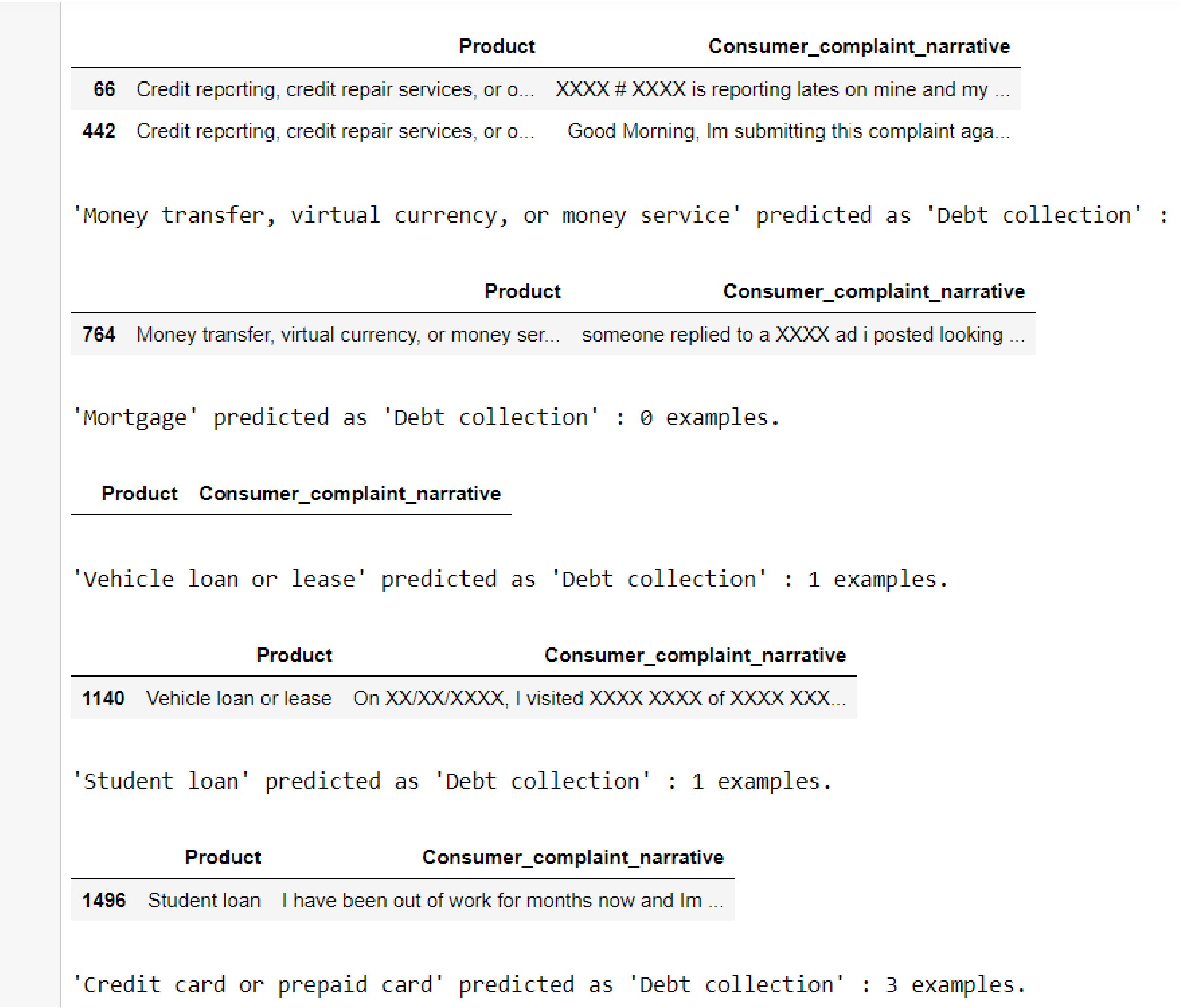
```
model_name
LinearSVC                 0.732686
LogisticRegression        0.581355
MultinomialNB             0.512443
RandomForestClassifier    0.490373
Name: accuracy, dtype: float64
```

In [18]:
```python
model = LinearSVC()
X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split(features, labels, df.index, test_size=0.33, ranc
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.values)
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



In [24]:
```python
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] >= 0:
      print("'{}' predicted as '{}' : {} examples.".format(id_to_category[actual], id_to_category[predicted], conf_mat[actual, pr
      display(df.loc[indices_test[(y_test == actual) & (y_pred == predicted)]][['Product', 'Consumer_complaint_narrative']])
      print('')
```

```
'Credit reporting, credit repair services, or other personal consumer reports' predicted as 'Debt collection' : 2 examples.
```



	Product	Consumer_complaint_narrative
66	Credit reporting, credit repair services, or o...	XXXX # XXXX is reporting lates on mine and my ...
442	Credit reporting, credit repair services, or o...	Good Morning, Im submitting this complaint aga...

```
'Money transfer, virtual currency, or money service' predicted as 'Debt collection' :
```

	Product	Consumer_complaint_narrative
764	Money transfer, virtual currency, or money ser...	someone replied to a XXXX ad i posted looking ...

```
'Mortgage' predicted as 'Debt collection' : 0 examples.
```

	Product	Consumer_complaint_narrative

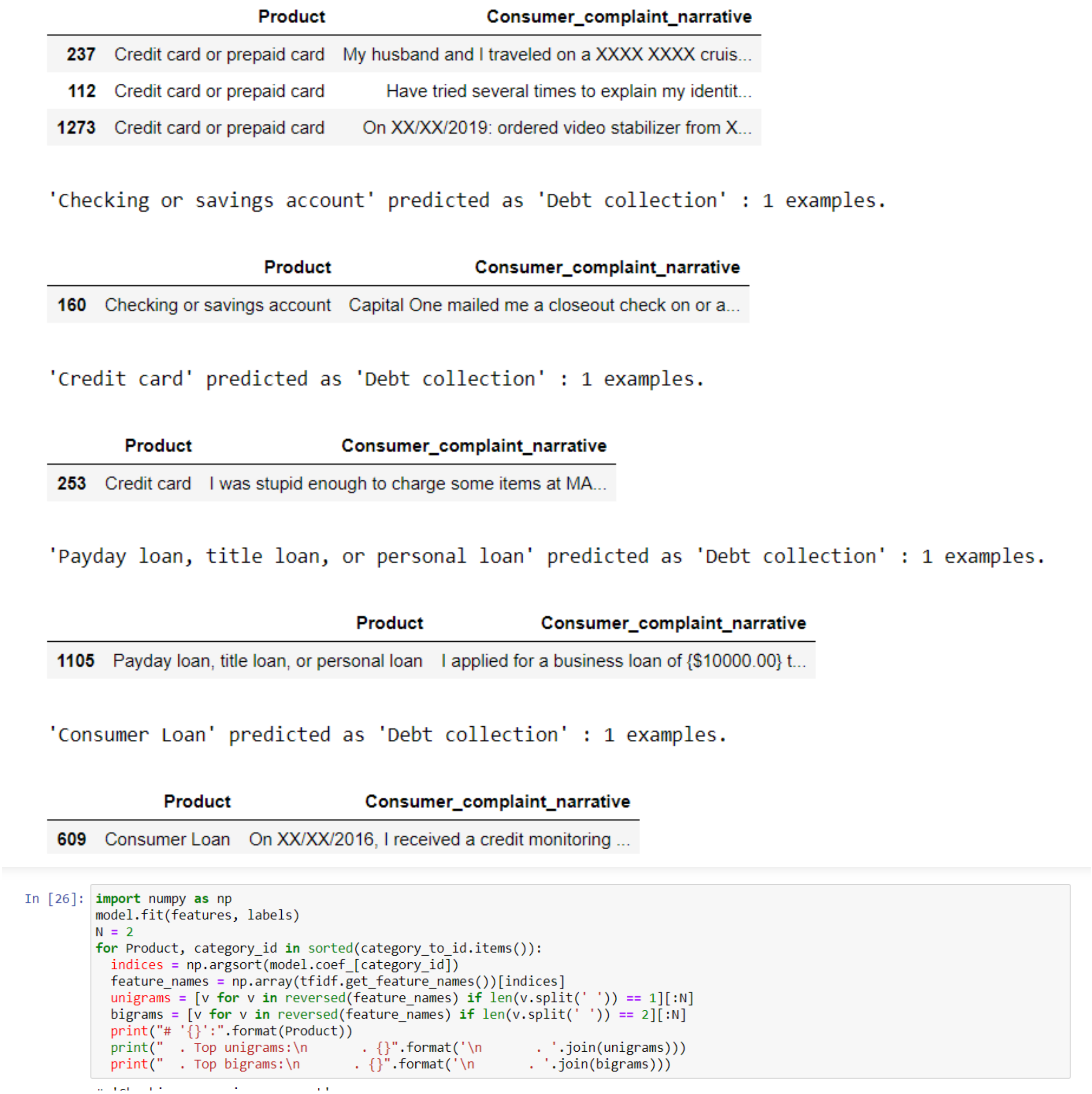
```
'Vehicle loan or lease' predicted as 'Debt collection' : 1 examples.
```

	Product	Consumer_complaint_narrative
1140	Vehicle loan or lease	On XX/XX/XXXX, I visited XXXX XXXX of XXXX XXX...

```
'Student loan' predicted as 'Debt collection' : 1 examples.
```

	Product	Consumer_complaint_narrative
1496	Student loan	I have been out of work for months now and Im ...

```
'Credit card or prepaid card' predicted as 'Debt collection' : 3 examples.
```



	Product	Consumer_complaint_narrative
237	Credit card or prepaid card	My husband and I traveled on a XXXX XXXX cruis...
112	Credit card or prepaid card	Have tried several times to explain my identit...
1273	Credit card or prepaid card	On XX/XX/2019: ordered video stabilizer from X...

```
'Checking or savings account' predicted as 'Debt collection' : 1 examples.
```

	Product	Consumer_complaint_narrative
160	Checking or savings account	Capital One mailed me a closeout check on or a...

```
'Credit card' predicted as 'Debt collection' : 1 examples.
```

	Product	Consumer_complaint_narrative
253	Credit card	I was stupid enough to charge some items at MA...

```
'Payday loan, title loan, or personal loan' predicted as 'Debt collection' : 1 examples.
```

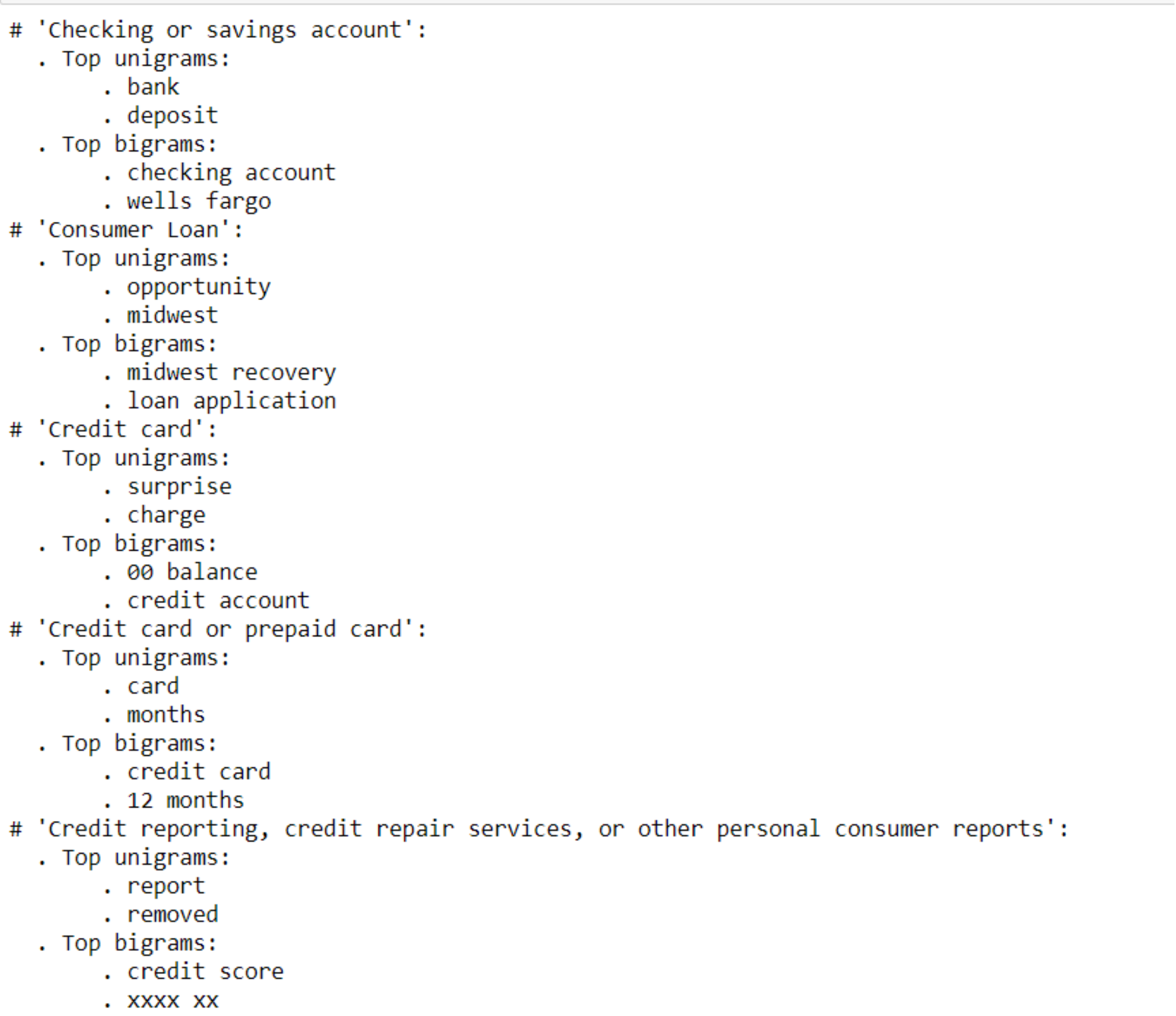
	Product	Consumer_complaint_narrative
1105	Payday loan, title loan, or personal loan	I applied for a business loan of {$10000.00} t...

```
'Consumer Loan' predicted as 'Debt collection' : 1 examples.
```

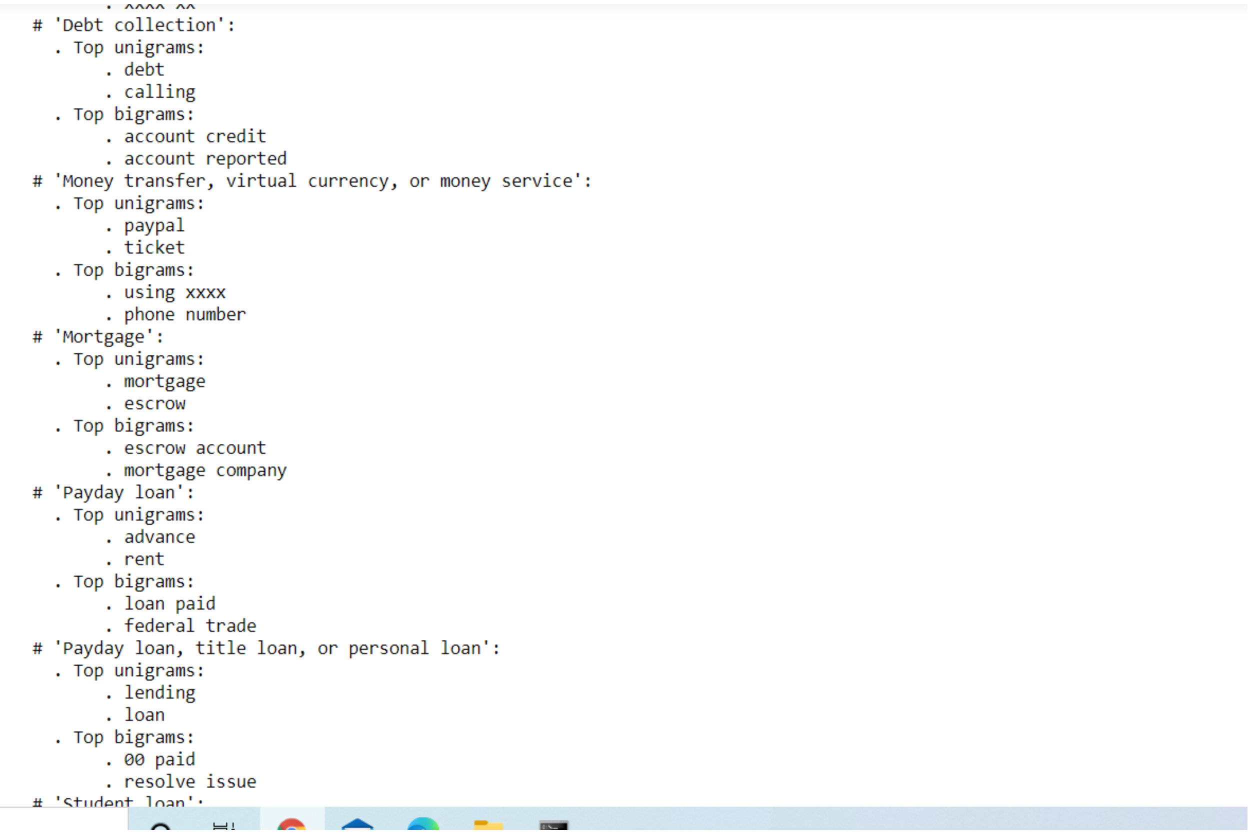
	Product	Consumer_complaint_narrative
609	Consumer Loan	On XX/XX/2016, I received a credit monitoring ...

In [26]:

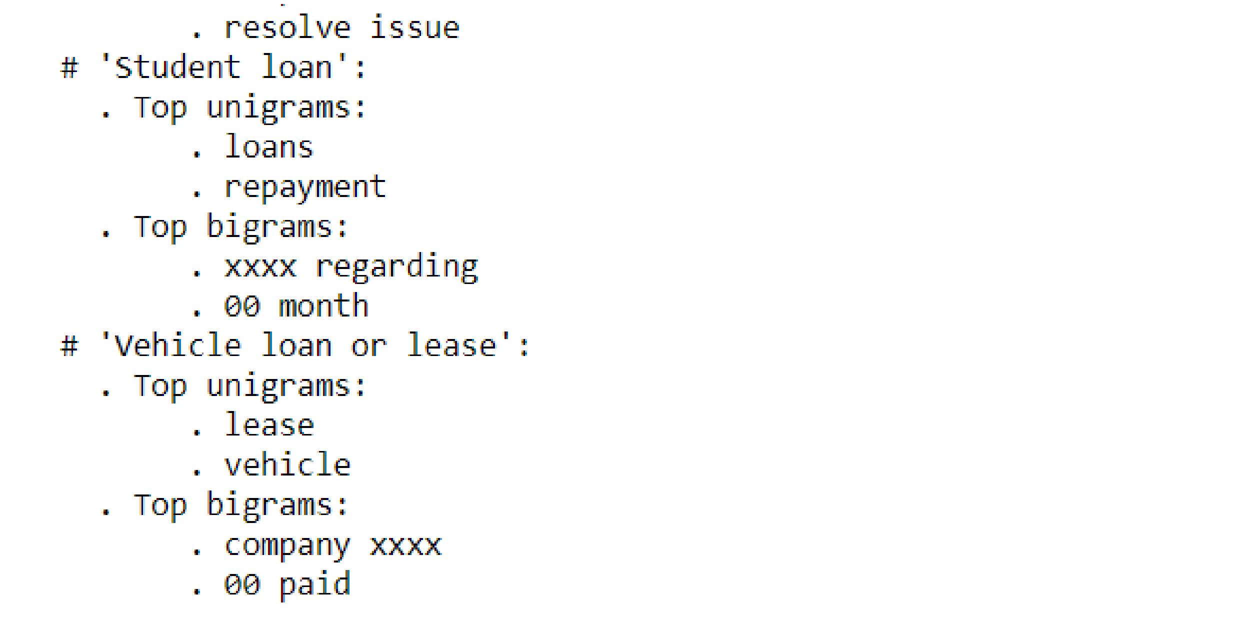
```
import numpy as np
model.fit(features, labels)
N = 2
for Product, category_id in sorted(category_to_id.items()):
  indices = np.argsort(model.coef_[category_id])
  feature_names = np.array(tfidf.get_feature_names())[indices]
  unigrams = [v for v in reversed(feature_names) if len(v.split(' ')) == 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)))
```



```
# 'Checking or savings account':
  . Top unigrams:
       . bank
       . deposit
  . Top bigrams:
       . checking account
       . wells fargo
# 'Consumer Loan':
  . Top unigrams:
       . opportunity
       . midwest
  . Top bigrams:
       . midwest recovery
       . loan application
# 'Credit card':
  . Top unigrams:
       . surprise
       . charge
  . Top bigrams:
       . 00 balance
       . credit account
# 'Credit card or prepaid card':
  . Top unigrams:
       . card
       . months
  . Top bigrams:
       . credit card
       . 12 months
# 'Credit reporting, credit repair services, or other personal consumer reports':
  . Top unigrams:
       . report
       . removed
  . Top bigrams:
       . credit score
       . xxxx xx
```

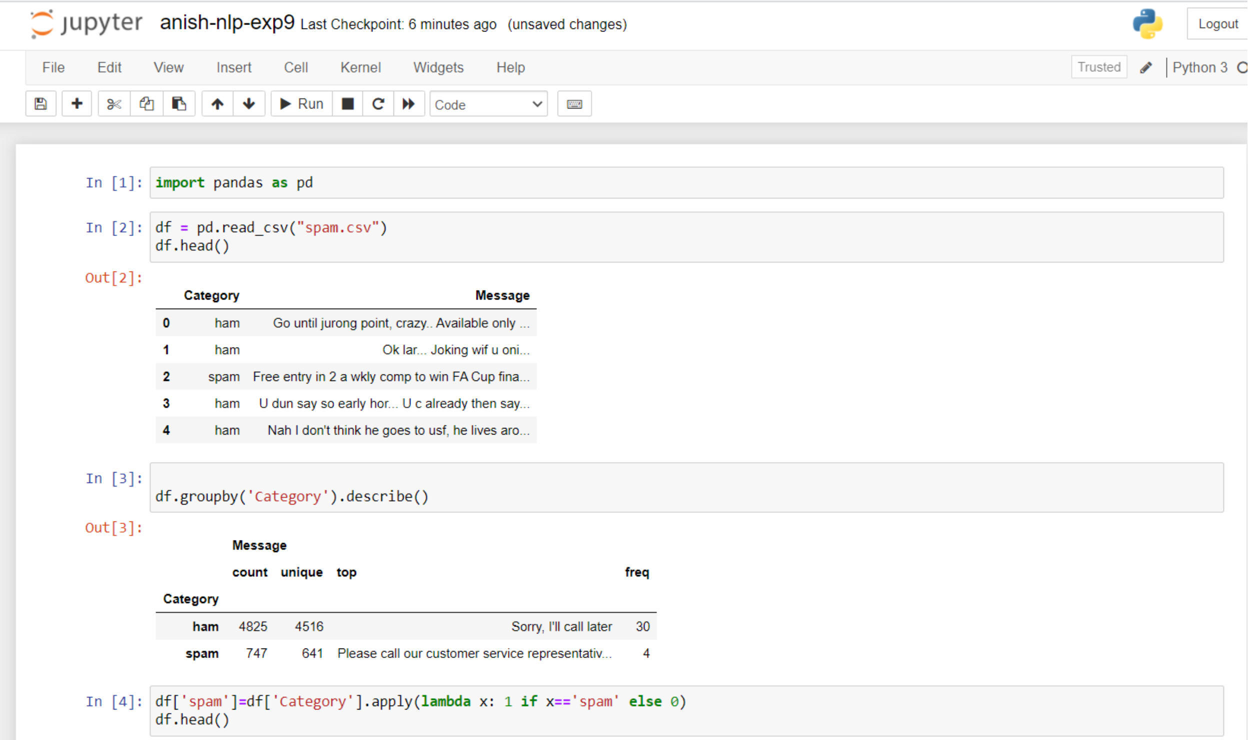


```
        . xxxx xx
# 'Debt collection':
  . Top unigrams:
       . debt
       . calling
  . Top bigrams:
       . account credit
       . account reported
# 'Money transfer, virtual currency, or money service':
  . Top unigrams:
       . paypal
       . ticket
  . Top bigrams:
       . using xxxx
       . phone number
# 'Mortgage':
  . Top unigrams:
       . mortgage
       . escrow
  . Top bigrams:
       . escrow account
       . mortgage company
# 'Payday loan':
  . Top unigrams:
       . advance
       . rent
  . Top bigrams:
       . loan paid
       . federal trade
# 'Payday loan, title loan, or personal loan':
  . Top unigrams:
       . lending
       . loan
  . Top bigrams:
       . 00 paid
       . resolve issue
# 'Student loan':
```



```
       . resolve issue
# 'Student loan':
  . Top unigrams:
       . loans
       . repayment
  . Top bigrams:
       . xxxx regarding
       . 00 month
# 'Vehicle loan or lease':
  . Top unigrams:
       . lease
       . vehicle
  . Top bigrams:
       . company xxxx
       . 00 paid
```

Naive Bayes Email Spam Filter



In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("spam.csv")
df.head()
```

Out[2]:

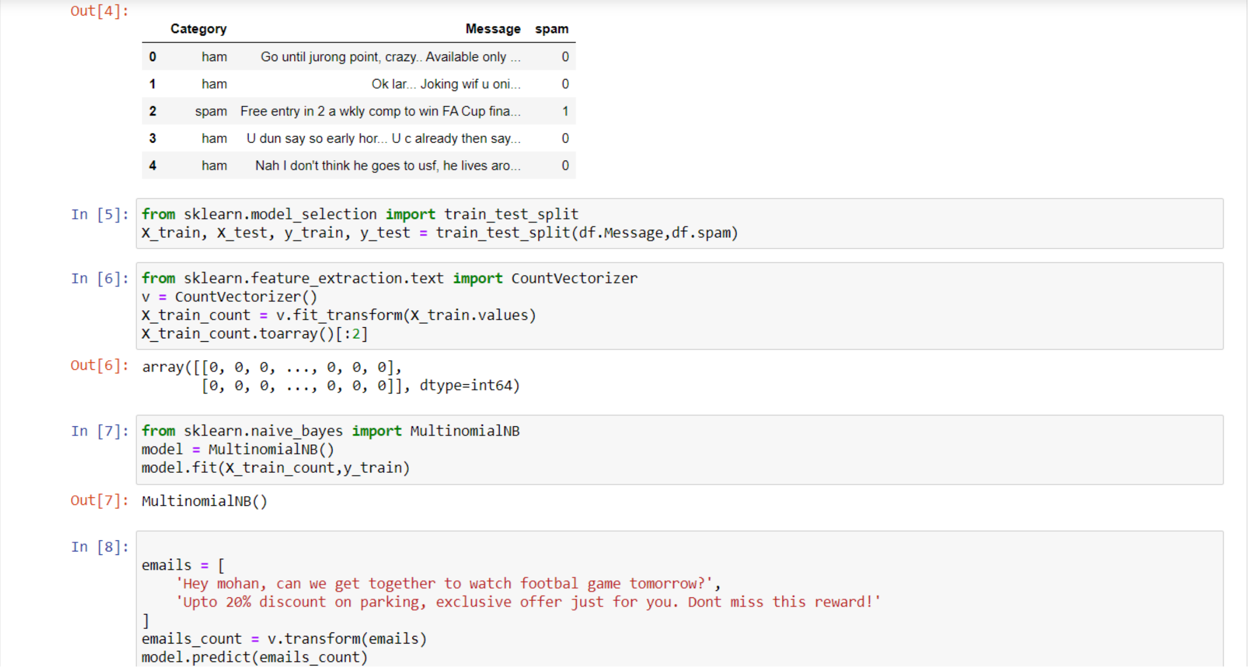
	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

In [3]:
```
df.groupby('Category').describe()
```

Out[3]:

	Message			
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	641	Please call our customer service representativ...	4

In [4]:
```
df['spam']=df['Category'].apply(lambda x: 1 if x=='spam' else 0)
df.head()
```



Out[4]:

	Category	Message	spam
0	ham	Go until jurong point, crazy.. Available only ...	0
1	ham	Ok lar... Joking wif u oni...	0
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	1
3	ham	U dun say so early hor... U c already then say...	0
4	ham	Nah I don't think he goes to usf, he lives aro...	0

In [5]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.Message,df.spam)
```

In [6]:
```
from sklearn.feature_extraction.text import CountVectorizer
v = CountVectorizer()
X_train_count = v.fit_transform(X_train.values)
X_train_count.toarray()[:2]
```

Out[6]:
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [7]:
```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB()
model.fit(X_train_count,y_train)
```

Out[7]:
```
MultinomialNB()
```

In [8]:
```

emails = [
    'Hey mohan, can we get together to watch footbal game tomorrow?',
    'Upto 20% discount on parking, exclusive offer just for you. Dont miss this reward!'
]
emails_count = v.transform(emails)
model.predict(emails_count)
```

```
 Out[8]: array([0, 1], dtype=int64)

  In [9]: X_test_count = v.transform(X_test)
          model.score(X_test_count, y_test)

 Out[9]: 0.9863603732950467

 In [10]: from sklearn.pipeline import Pipeline
          clf = Pipeline([
              ('vectorizer', CountVectorizer()),
              ('nb', MultinomialNB())
          ])

 In [11]: clf.fit(X_train, y_train)

Out[11]: Pipeline(steps=[('vectorizer', CountVectorizer()), ('nb', MultinomialNB())])

 In [12]: 
          clf.score(X_test,y_test)

Out[12]: 0.9863603732950467

 In [13]: clf.predict(emails)

Out[13]: array([0, 1], dtype=int64)

  In [ ]: 
```