In [1]:
```
import pandas as pd
data = pd.read_csv('Consumer_Complaints.csv')
data.groupby('Product').count()
columns = ['Virtual currency','Other financial service']
smallGroupDf = data[data['Product'].isin(columns)]
smallGroupDf.shape
sampleDf = data[~data['Product'].isin(columns)]
sampleDf = sampleDf.sample(frac=0.02)
df = pd.concat([smallGroupDf,sampleDf])
df.shape
del data, sampleDf, smallGroupDf
df.head()
```

```
c:\users\mglewis\appdata\local\programs\python\python36\lib\site-packages\IPyth
on\core\interactiveshell.py:3049: DtypeWarning: Columns (5,6,11,16) have mixed
types. Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[1]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company
489377	04/21/2017	Other financial service	Credit repair	Disclosures	NaN	NaN	NaN	TIAA, FSB
489513	04/21/2017	Other financial service	Debt settlement	Fraud or scam	NaN	NaN	Company believes it acted appropriately as aut...	Equitable Acceptance Corporation
489637	04/21/2017	Other financial service	Check cashing	Customer service/Customer relations	NaN	NaN	Company has responded to the consumer and the ...	Fidelity National Information Services Inc. (...
489773	04/20/2017	Other financial service	Debt settlement	Fraud or scam	NaN	NaN	Company believes it acted appropriately as aut...	Equitable Acceptance Corporation
489812	04/20/2017	Other financial service	Debt settlement	Fraud or scam	NaN	I attempted to contact Navient to discuss my s...	NaN	Navient Solutions LLC

In [2]:
```
df = df[pd.notnull(df['Consumer complaint narrative'])]
```

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7830 entries, 489812 to 835259
Data columns (total 18 columns):
Date received                   7830 non-null object
Product                         7830 non-null object
Sub-product                     6812 non-null object
Issue                           7830 non-null object
Sub-issue                       5337 non-null object
Consumer complaint narrative    7830 non-null object
Company public response         3697 non-null object
Company                         7830 non-null object
State                           7799 non-null object
ZIP code                        5998 non-null object
Tags                            1364 non-null object
Consumer consent provided?      7830 non-null object
Submitted via                   7830 non-null object
Date sent to company            7830 non-null object
Company response to consumer    7830 non-null object
Timely response?                7830 non-null object
Consumer disputed?              3490 non-null object
Complaint ID                    7830 non-null int64
dtypes: int64(1), object(17)
memory usage: 1.1+ MB
```

In [4]:
```
col = ['Product', 'Consumer complaint narrative']
df = df[col]
```

In [5]:
```

df.columns
```

Out[5]:
```
Index(['Product', 'Consumer complaint narrative'], dtype='object')
```

In [6]:
```

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

In [7]:
```
df['category_id'] = df['Product'].factorize()[0]
from io import StringIO
category_id_df = df[['Product', 'category_id']].drop_duplicates().sort_values('c
category_to_id = dict(category_id_df.values)
id_to_category = dict(category_id_df[['category_id', 'Product']].values)
```

In [8]:
```
df.head()
```

Out[8]:

	Product	Consumer_complaint_narrative	category_id
489812	Other financial service	I attempted to contact Navient to discuss my s...	0
491814	Other financial service	I was charged XXXX by The Student Loan Help ...	0
492528	Other financial service	I received a call from XXXX XXXX of Premie...	0
492632	Other financial service	Several checks were issued from XXXX for possi...	0
493590	Other financial service	I enrolled in National Budget Planners of Sout...	0

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()
```

```
<Figure size 800x600 with 1 Axes>
```

In [10]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

Out[10]:
```
(7830, 21848)
```

In [11]:
```
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(u
  print("  . Most correlated bigrams:\n       . {}".format('\n       . '.join(bi
```

```
# 'Student loan':
  . Most correlated unigrams:
       . loans
       . navient
  . Most correlated bigrams:
       . income based
       . student loans
# 'Vehicle loan or lease':
  . Most correlated unigrams:
       . car
       . vehicle
  . Most correlated bigrams:
       . return car
       . credit acceptance
# 'Virtual currency':
  . Most correlated unigrams:
       . coinbase
       . signup
  . Most correlated bigrams:
       . 00 completed
```

In [12]:
```
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_narra
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 [13]:
```
refuses to provide me verification and validation of debt per my right under the
```

```
['Debt collection']
```

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

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

In [15]:
```
any refuses to provide me verification and validation of debt per my right under t
```

Out[15]:

Product	Consumer_complaint_narrative	category_id

In [16]:
```
sition now and they would n't have to research once again. I would like the report
```

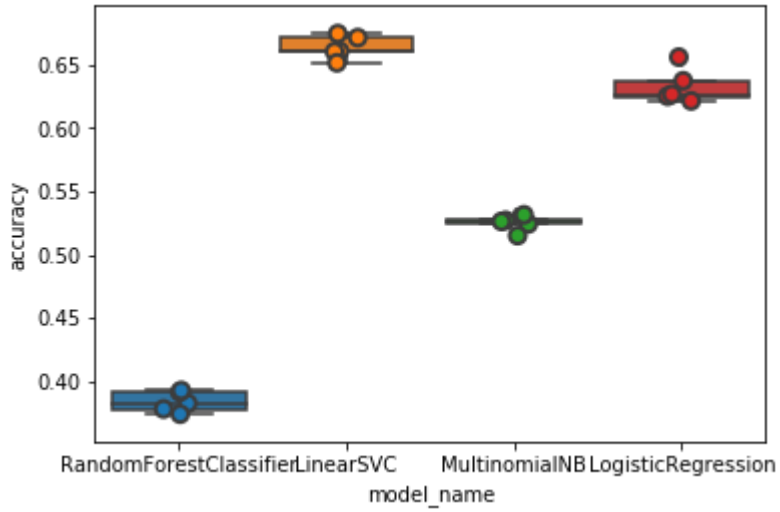
Out[16]:

Product	Consumer_complaint_narrative	category_id

In [17]:
```
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'])
```

```
c:\users\mglewis\appdata\local\programs\python\python36\lib\site-packages\sklea
rn\ensemble\weight_boosting.py:29: DeprecationWarning: numpy.core.umath_tests i
s an internal NumPy module and should not be imported. It will be removed in a
future NumPy release.
  from numpy.core.umath_tests import inner1d
```

In [18]:
```
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()
```

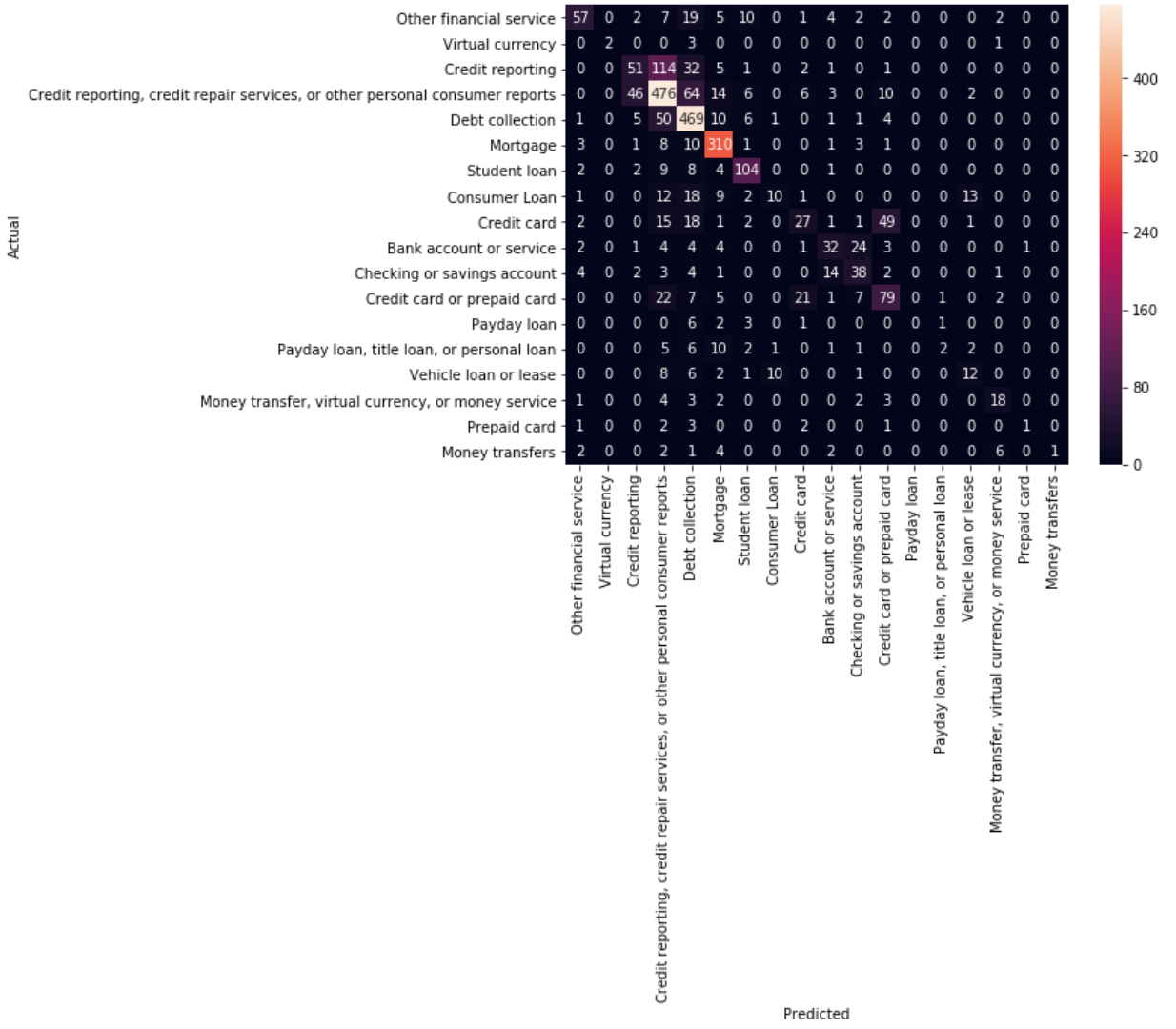


In [19]:
```
cv_df.groupby('model_name').accuracy.mean()
```

Out[19]:
```
model_name
LinearSVC                 0.663479
LogisticRegression        0.633097
MultinomialNB             0.524776
RandomForestClassifier    0.383906
Name: accuracy, dtype: float64
```

In [20]:
```
from sklearn.model_selection import train_test_split

model = LinearSVC()

X_train, X_test, y_train, y_test, indices_train, indices_test = train_test_split
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

In [21]:
```
from sklearn.metrics import confusion_matrix

conf_mat = confusion_matrix(y_test, y_pred)
fig, ax = plt.subplots(figsize=(8,6))
sns.heatmap(conf_mat, annot=True, fmt='d',
            xticklabels=category_id_df.Product.values, yticklabels=category_id_d
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



In [22]:
```
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
      display(df.loc[indices_test[(y_test == actual) & (y_pred == predicted)]][[
      print('')
```

```
'Credit reporting, credit repair services, or other personal consumer report
s' predicted as 'Credit reporting' : 46 examples.
```

In [23]:
```
model.fit(features, labels)
```

Out[23]:
```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
```

In [24]:
```
from sklearn.feature_selection import chi2

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
       . cd
  . Top bigrams:
       . debit card
       . 00 bank
# 'Checking or savings account':
  . Top unigrams:
       . bank
       . funds
  . Top bigrams:
       . account xx
       . deposit xx
# 'Consumer Loan':
  . Top unigrams:
       . vehicle
       . car
  . Top bigrams:
```

In [25]:
```
texts = ["I requested a home loan modification through Bank of America. Bank of A
         "It has been difficult for me to find my past due balance. I missed a re
         "I can't get the money out of the country.",
         "I have no money to pay my tuition",
         "Coinbase closed my account for no reason and furthermore refused to giv
text_features = tfidf.transform(texts)
predictions = model.predict(text_features)
for text, predicted in zip(texts, predictions):
  print('"{}"'.format(text))
  print("  - Predicted as: '{}'".format(id_to_category[predicted]))
  print("")
```

```
"I requested a home loan modification through Bank of America. Bank of America
never got back to me."
  - Predicted as: 'Mortgage'

"It has been difficult for me to find my past due balance. I missed a regular m
onthly payment"
  - Predicted as: 'Consumer Loan'

"I can't get the money out of the country."
  - Predicted as: 'Other financial service'

"I have no money to pay my tuition"
  - Predicted as: 'Debt collection'

"Coinbase closed my account for no reason and furthermore refused to give me a
reason despite dozens of request"
  - Predicted as: 'Money transfer, virtual currency, or money service'
```

In [26]:
```
from sklearn import metrics
print(metrics.classification_report(y_test, y_pred,
                                    target_names=df['Product'].unique()))
```

```
                                                                                p
recision    recall  f1-score   support

                                                          Other financial service
0.75      0.51      0.61       111
                                                                 Virtual currency
1.00      0.33      0.50         6
                                                                 Credit reporting
0.46      0.25      0.32       207
Credit reporting, credit repair services, or other personal consumer reports
0.64      0.76      0.70       627
                                                                  Debt collection
0.69      0.86      0.76       548
                                                                         Mortgage
0.80      0.92      0.85       338
                                                                     Student loan
0.75      0.80      0.78       130
                                                                    Consumer Loan
0.45      0.15      0.23        66
                                                                      Credit card
0.44      0.23      0.30       117
                                                          Bank account or service
0.52      0.42      0.46        76
                                                      Checking or savings account
0.47      0.55      0.51        69
                                                      Credit card or prepaid card
0.51      0.54      0.53       145
                                                                      Payday loan
0.00      0.00      0.00        13
                                          Payday loan, title loan, or personal loan
0.50      0.07      0.12        30
                                                           Vehicle loan or lease
0.40      0.30      0.34        40
                                   Money transfer, virtual currency, or money service
0.60      0.55      0.57        33
                                                                     Prepaid card
0.50      0.10      0.17        10
                                                                  Money transfers
1.00      0.06      0.11        18

                                                                      avg / total
0.63      0.65      0.63      2584

c:\users\mglewis\appdata\local\programs\python\python36\lib\site-packages\sklea
rn\metrics\classification.py:1135: UndefinedMetricWarning: Precision and F-scor
e are ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

In [ ]: