**Consumer-Complaints**

# coding: utf-8

# In[1]:

import pandas as pd

df = pd.read\_csv(r'C:\Users\akasriva2\Downloads\Consumer\_Complaints.csv')

df.head()

# In[2]:

df = df[pd.notnull(df['Consumer complaint narrative'])]

# In[3]:

df.info()

# In[4]:

col = ['Product', 'Consumer complaint narrative']

df = df[col]

# In[5]:

df.columns

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('category\_id')

category\_to\_id = dict(category\_id\_df.values)

id\_to\_category = dict(category\_id\_df[['category\_id', 'Product']].values)

# In[8]:

df.head()

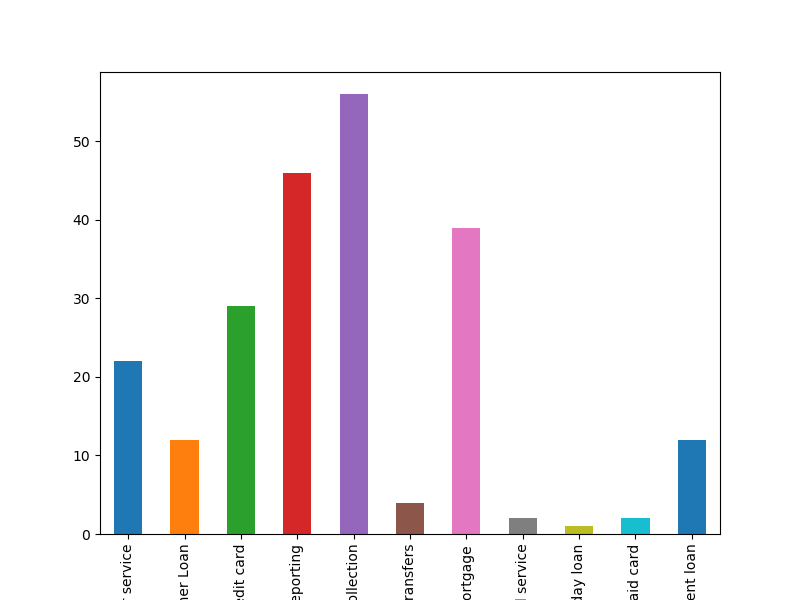
# In[9]:

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

# 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(unigrams[-N:])))

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

# 'Bank account or service':

. Most correlated unigrams:

. funds

. overdraft

. Most correlated bigrams:

. 20 00

. closed account

# 'Consumer Loan':

. Most correlated unigrams:

. ally

. vehicle

. Most correlated bigrams:

. 00 xxxx

. received xxxx

# 'Credit card':

. Most correlated unigrams:

. annual

. card

. Most correlated bigrams:

. months later

. credit card

# 'Credit reporting':

. Most correlated unigrams:

. equifax

. experian

. Most correlated bigrams:

. information credit

. credit report

# 'Debt collection':

. Most correlated unigrams:

. collection

. debt

. Most correlated bigrams:

. collection agency

. debt collector

# 'Money transfers':

. Most correlated unigrams:

. buyer

. attention

. Most correlated bigrams:

. sent xxxx

. mortgage company

# 'Mortgage':

. Most correlated unigrams:

. modification

. mortgage

. Most correlated bigrams:

. short sale

. mortgage loan

# 'Other financial service':

. Most correlated unigrams:

. worked

. support

. Most correlated bigrams:

. score xxxx

. bank america

# 'Payday loan':

. Most correlated unigrams:

. finance

. principal

. Most correlated bigrams:

. xxxx month

. xxxx loan

# 'Prepaid card':

. Most correlated unigrams:

. faxed

. ability

. Most correlated bigrams:

. credit card

. federal law

# 'Student loan':

. Most correlated unigrams:

. repayment

. navient

. Most correlated bigrams:

. make payments

. student loan

# 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\_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[13]:

print(clf.predict(count\_vect.transform(["This company refuses to provide me verification and validation of debt per my right under the FDCPA. I do not believe this debt is mine."])))

['Debt collection']

# In[14]:

df[df['Consumer\_complaint\_narrative'] == "This company refuses to provide me verification and validation of debt per my right under the FDCPA. I do not believe this debt is mine."]

|  | **Product** | **Consumer\_complaint\_narrative** | **category\_id** |
| --- | --- | --- | --- |
| **12** | Debt collection | This company refuses to provide me verificatio... | 2 |

# In[15]:

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'])

# In[16]:

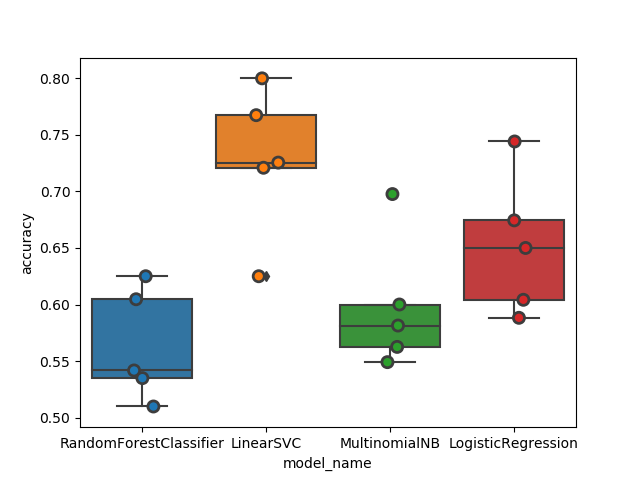
import seaborn as sns

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()

model\_name

LinearSVC 0.727772

LogisticRegression 0.652201

MultinomialNB 0.598118

RandomForestClassifier 0.563201

Name: accuracy, dtype: float64