In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
```

In [2]:
```
data = pd.read_csv("cokezeroreviewsv3.csv")
```

In [3]:
```
data.shape
```

Out[3]:
```
(2138, 2)
```

In [4]:
```
data.head()
```

Out[4]:

	Text	Sentiment
0	This is absolutely the best soda ever in my op...	1
1	I am not really a fan of Coke Zero. It's just ...	2
2	This is the best diet soda out there. I tried ...	1
3	Let's start off by saying I am not a fan of ze...	2
4	Zero Sugar Coke isn't the tastiest soda around...	1

In [5]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2138 entries, 0 to 2137
Data columns (total 2 columns):
Text         2138 non-null object
Sentiment    2138 non-null int64
dtypes: int64(1), object(1)
memory usage: 33.5+ KB
```

In [6]:
```
data_class = data[(data['Sentiment'] == 1) | (data['Sentiment'] == 2)]
data_class.shape
```

Out[6]:
```
(2138, 2)
```

In [7]:
```
X = data_class['Text']
y = data_class['Sentiment']
```

In [8]:
```
import string
def text_process(text):
    '''
    Takes in a string of text, then performs the following:
    1. Remove all punctuation
    2. Remove all stopwords
    3. Return the cleaned text as a list of words
    '''
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    return [word for word in nopunc.split() if word.lower() not in stopwords.w
ords('english')]
```

In [9]:
```
sample_text = "Hey there! This is a sample review, which happens to contain pu
nctuations."
print(text_process(sample_text))
```

```
['Hey', 'sample', 'review', 'happens', 'contain', 'punctuations']
```

In [10]:
```
X[0]
```

Out[10]:
```
"This is absolutely the best soda ever in my opinion. If I'm not drinking wat
er, it's none other than Coke Zero. It's such a refreshing, crisp taste that
satisfies my taste buds. I'm totally in love with Coca Cola Zero Sugar. At le
ast I'm not getting all that sugar but there's really not much of a taste dif
ference than Coke Classic. And the best part is about Coca Cola Zero Sugar is
that there's NEVER an aftertaste. This is my sugar fix without all the calori
es My fave part. I have gotten several of my family members hooked on it as w
ell. I will have to give kudos to Coke for creating that amazing soda.\n#infl
uenster. #Cocacolazerosugar #heavydrinker #noaftertaste"
```

In [11]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [12]:
```
bow_transformer = CountVectorizer(analyzer=text_process).fit(X)
```

In [13]:
```
len(bow_transformer.vocabulary_)
```

Out[13]:
```
3355
```

In [14]:
```
review_25 = X[24]
review_25
```

Out[14]:
```
"I am totally disappointed in Coke for making this new soda. The soda could h
ave simply been renamed instead of the recipe being messed up. It's not awfu
l, but it's not great.I"
```

In [15]:
```
X = bow_transformer.transform(X)
```

In [32]:
```
print('Shape of Sparse Matrix: ', X.shape)
print('Amount of Non-Zero occurrences: ', X.nnz)
```

```
Shape of Sparse Matrix:  (2138, 3355)
Amount of Non-Zero occurrences:  30032
```

In [100]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, rando
m_state=101)
```

In [17]:
```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(X_train, y_train)
```

Out[17]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [18]:
```
preds = nb.predict(X_test)
```

In [19]:
```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.7990654205607477
```

In [20]:
```
from sklearn.metrics import confusion_matrix, classification_report
print(confusion_matrix(y_test, preds))
print('\n')
print(classification_report(y_test, preds))
```

```
[[501  24]
 [105  12]]


             precision    recall  f1-score   support

          1       0.83      0.95      0.89       525
          2       0.33      0.10      0.16       117

avg / total       0.74      0.80      0.75       642
```

In [21]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
tf=TfidfVectorizer()
text_tf= tf.fit_transform(data['Text'])
```

In [22]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    text_tf, data['Sentiment'], test_size=0.3, random_state=123)
```

In [24]:
```
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
# Model Generation Using Multinomial Naive Bayes
clf = MultinomialNB().fit(X_train, y_train)
predicted= clf.predict(X_test)
print("MultinomialNB Accuracy:",metrics.accuracy_score(y_test, predicted))
```

```
MultinomialNB Accuracy: 0.8161993769470405
```

In [106]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.model_selection import cross_val_score

pipe = Pipeline([("vectorizer", CountVectorizer(ngram_range=(1, 2), min_df=1
)),
                 ("clf", MultinomialNB())])
grid = GridSearchCV(pipe, param_grid={
    "clf":[MultinomialNB(), BernoulliNB()],
    "clf__alpha":[0.001,0.01,0.1,1,10]
}, cv=5)

cross_val_score(grid, X, y, cv=5)
```

Out[106]:
```
array([0.82009346, 0.77570093, 0.82009346, 0.82009346, 0.82159624])
```