In [247]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

In [248]:
```
data = pd.read_csv("shoe_reviews_1.txt", delimiter = ":")
```

In [249]:
```
data.head()
```

Out[249]:

	Review	Class
0	Not happy at all. Second time out the backs of...	Negative
1	These Brooks shoes are great! I run every few ...	Positive
2	Worst running shoe ever.	Negative
3	This is honestly one of the best shoes I have ...	Positive
4	Well I used the shoes for couple of weeks to g...	Negative

In [250]:
```
def basic_split(data, y, length, split_point = 0.5): #split data 50-50
    n = int(split_point * length)
    X_train = data[:n].copy() #All data points till the split point
    X_test = data[n:].copy() #all points after split
    y_train = y[:n].copy()
    y_test = y[n:].copy()
    return X_train, X_test, y_train, y_test
```

In [251]:
```
vectorizer = CountVectorizer()
```

In [252]:
```
X_train, X_test, y_train, y_test = basic_split(data.Review, data.Class, len(d
```

In [253]:
```
X_train = vectorizer.fit_transform(X_train) #fit and transform to training da
X_test = vectorizer.transform(X_test) #transform for test data set
```

In [254]:
```
features = vectorizer.get_feature_names()
print(len(features))
print(features)
```

```
98
['13', 'about', 'accommodated', 'actually', 'all', 'also', 'amount', 'and',
'ankles', 'are', 'as', 'at', 'awesome', 'backs', 'became', 'best', 'bigge
r', 'bleeding', 'bought', 'bounce', 'brooks', 'but', 'causing', 'changer',
'couple', 'currently', 'days', 'disappointed', 'discomfort', 'do', 'dug',
'ever', 'every', 'fast', 'feet', 'few', 'for', 'forget', 'game', 'get', 'gh
ost', 'go', 'got', 'great', 'half', 'happy', 'have', 'honestly', 'hurt', 'i
f', 'into', 'is', 'it', 'just', 'light', 'like', 'me', 'my', 'not', 'of',
'one', 'out', 'pain', 'progressively', 'regret', 'right', 'run', 'running',
'second', 'shin', 'shoe', 'shoes', 'size', 'slower', 'small', 'sneakers',
'so', 'style', 'super', 'support', 'the', 'them', 'these', 'they', 'think',
'this', 'time', 'to', 'try', 'used', 'very', 'want', 'weeks', 'well', 'wo
n', 'worse', 'worst', 'you']
```

In [255]:
```
bag_of_words_first_half = pd.DataFrame(X_train[:,:].todense(), columns = feat
bag_of_words_first_half #Need to add column for class here
```

Out[255]:

	13	about	accommodated	actually	all	also	amount	and	ankles	are	...	try	used	ve
0	0	0	0	0	1	0	0	1	1	0	...	0	0	
1	1	0	0	0	0	0	0	1	0	2	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	1	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	1	1	0	2	0	2	0	0	...	1	2	
5	0	0	0	0	0	0	1	1	0	0	...	0	0	

6 rows × 98 columns

In [256]:
```
#Extract class column for first/last 6 entries
class_column_first_half = data.Class[0:6]
class_column_second_half = data.Class[6:]
class_column_first_half
class_column_second_half
```

Out[256]:
```
6     Negative
7     Positive
8     Negative
9     Positive
10    Negative
11    Positive
Name: Class, dtype: object
```

In [257]:
```
bag_of_words_first_half['Class'] = class_column_first_half
```

In [258]:
```
bag_of_words_first_half
```

Out[258]:

	13	about	accommodated	actually	all	also	amount	and	ankles	are	...	used	very
0	0	0	0	0	1	0	0	1	1	0	...	0	1
1	1	0	0	0	0	0	0	1	0	2	...	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0
3	0	1	0	0	0	0	0	0	0	0	...	0	0
4	0	0	1	1	0	2	0	2	0	0	...	2	0
5	0	0	0	0	0	0	1	1	0	0	...	0	0

6 rows × 99 columns

In [259]:
```
is_positive = bag_of_words_first_half['Class'] == 'Positive'
positive_df = bag_of_words_first_half[is_positive]
is_negative = bag_of_words_first_half['Class'] == 'Negative'
negative_df = bag_of_words_first_half[is_negative]
negative_df
```

Out[259]:

	13	about	accommodated	actually	all	also	amount	and	ankles	are	...	used	very
0	0	0	0	0	1	0	0	1	1	0	...	0	1
2	0	0	0	0	0	0	0	0	0	0	...	0	0
4	0	0	1	1	0	2	0	2	0	0	...	2	0

3 rows × 99 columns

In [260]:
```
len(positive_df)
```

Out[260]: 3

In [261]:
```
positive_prior = 3/6
negative_prior = 3/6
positive_likelihood = []
negative_likelihood = []
for feature in features:
    count = 0
    for item in positive_df[feature]:
        if item != 0:
            count= count + 1 #count number of non zero entries for each word
        positive_likelihood.append(count/len(positive_df))
```

In [262]:
```
for feature in features:
    counter = 0
    for item in negative_df[feature]:
        if item != 0:
            counter = counter + 1
        negative_likelihood.append(counter/len(negative_df)) #build negative
```

In [263]:
```
evidence = []
for feature in features:
    counters = 0
    for item in bag_of_words_first_half[feature]:
        if item != 0:
            counters = counters + 1
    evidence.append(counters/len(bag_of_words_first_half))
```

In [264]:
```
positive_posterior = []
negative_posterior = []

for i in range(0, len(features)):
    positive_posterior_i = (positive_likelihood[i] * positive_prior)#/ (evide
    positive_posterior.append(positive_posterior_i)

for i in range(0, len(positive_posterior)) :
    if positive_posterior[i] < 0.00001:
        positive_posterior[i] = .000001
```

In [265]:
```
for i in range(0, len(features)):
    negative_posterior_i = (negative_likelihood[i] * negative_prior)#/ (evide
    negative_posterior.append(negative_posterior_i)
for i in range(0, len(negative_posterior)) :
    if negative_posterior[i] < 0.00001:
        negative_posterior[i] = .000001
```

In [266]:
```
bag_of_words_second_half = pd.DataFrame(X_test[:,:].todense(), columns = feat
```

In [267]:
```
bag_of_words_second_half
```

Out[267]:

	13	about	accommodated	actually	all	also	amount	and	ankles	are	...	try	used	ve
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	0	0	0	0	0	0	0	1	0	1	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	0	0	0	0	2	0	0	...	0	0	
4	0	0	0	0	0	0	0	2	0	1	...	0	0	
5	0	0	0	0	1	0	0	2	0	0	...	0	0	

6 rows × 98 columns

In [268]:
```
bag_of_words_second_half['Class'] = ['Negative', 'Positive', 'Negative', 'Pos
```

In [269]:
```
bag_of_words_second_half
```

Out[269]:

	13	about	accommodated	actually	all	also	amount	and	ankles	are	...	used	very	
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	0	0	0	0	0	0	0	1	0	1	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	0	0	0	0	2	0	0	...	0	0	
4	0	0	0	0	0	0	0	2	0	1	...	0	0	
5	0	0	0	0	1	0	0	2	0	0	...	0	0	

6 rows × 99 columns

In [270]:
```
predictions = []
prob_positive = positive_prior
prob_negative = negative_prior
for c in range(0,6):
    index_vector = []
    index = 0
    for item in bag_of_words_second_half.iloc[c]:
        if item != 0 and item != 'Positive' and item != 'Negative':
            index_vector.append(index) #any word that is present in the revie
        index = index + 1
    #At this point we have built our index vector for a particular row
    for elem in index_vector: #elem represents features index of every presen
        #word = features[elem]
        #first calculate probability that review is positive
        prob_positive = prob_positive * positive_posterior[elem]
        prob_negative = prob_negative * negative_posterior[elem]
    print(str(prob_positive) + "    " + str(prob_negative))
    if prob_positive >= prob_negative:
        predictions.append('Positive')
    else:
        predictions.append('Negative')
```

```
1.3888888888888886e-26    2.314814814814814e-21
3.8580246913580226e-70    3.858024691358022e-70
3.8580246913580226e-70    3.858024691358022e-70
1.78612254229538e-114    1.78612254229538e-114
2.2969682899889128e-184    3.828280483314854e-179
1.0634112453652372e-204    6.380467472191422e-210
```

In [271]:
```
predictions
```

Out[271]:
```
['Negative', 'Positive', 'Positive', 'Positive', 'Negative', 'Positive']
```

In [272]:
```
bag_of_words_second_half['Predicted_Class'] = predictions
bag_of_words_second_half
```

Out[272]:

	13	about	accommodated	actually	all	also	amount	and	ankles	are	...	very	want
0	0	0	0	0	0	0	0	0	0	0	...	0	0
1	0	0	0	0	0	0	0	1	0	1	...	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0
3	0	0	0	0	0	0	0	2	0	0	...	0	0
4	0	0	0	0	0	0	0	2	0	1	...	0	0
5	0	0	0	0	1	0	0	2	0	0	...	0	0

6 rows × 100 columns

Now 80-20 split

In [291]:
```
vectorizer = CountVectorizer()
```

In [292]:
```
X_train, X_test, y_train, y_test = basic_split(data.Review, data.Class, len(d
```

In [293]:
```
X_train
```

Out[293]:
```
0    Not happy at all. Second time out the backs of...
1    These Brooks shoes are great! I run every few ...
2                              Worst running shoe ever.
3    This is honestly one of the best shoes I have ...
4    Well I used the shoes for couple of weeks to g...
5    Just do it. you won't regret it. great support...
6    Was so excited to get these, only to get heal ...
7    Go up 1/2 size for the best fit. My son and da...
8                                            Fit tight
Name: Review, dtype: object
```

In [294]:
```
X_train = vectorizer.fit_transform(X_train) #fit and transform to training da
X_test = vectorizer.transform(X_test) #transform for test data set
```

In [295]:
```
features = vectorizer.get_feature_names()
print(len(features))
print(features)
```

```
122
['13', '30', 'about', 'accommodated', 'actually', 'after', 'all', 'also',
'amount', 'and', 'ankles', 'are', 'around', 'as', 'at', 'awesome', 'backs',
'became', 'best', 'bigger', 'bleeding', 'blisters', 'bought', 'bounce', 'br
ooks', 'but', 'causing', 'center', 'changer', 'community', 'couple', 'curre
ntly', 'daughter', 'days', 'disappointed', 'discomfort', 'do', 'dug', 'eve
r', 'every', 'excited', 'fast', 'feet', 'few', 'fit', 'for', 'forget', 'gam
e', 'get', 'ghost', 'go', 'got', 'great', 'had', 'half', 'happy', 'have',
'heal', 'honestly', 'hurt', 'if', 'into', 'is', 'it', 'just', 'light', 'lik
e', 'love', 'me', 'minute', 'my', 'not', 'of', 'one', 'only', 'our', 'out',
'pain', 'power', 'progressively', 'regret', 'return', 'right', 'run', 'runn
ers', 'running', 'second', 'shin', 'shoe', 'shoes', 'size', 'slower', 'smal
l', 'sneakers', 'so', 'son', 'style', 'super', 'support', 'the', 'them', 't
hese', 'they', 'think', 'this', 'tight', 'time', 'to', 'try', 'up', 'used',
'very', 'walk', 'want', 'was', 'weeks', 'well', 'who', 'won', 'worse', 'wor
st', 'you']
```

In [296]:
```
bag_of_words_train = pd.DataFrame(X_train[:,:].todense(), columns = features)
bag_of_words_train #Need to add column for class here
```

Out[296]:

	13	30	about	accommodated	actually	after	all	also	amount	and	...	walk	want	wa
0	0	0	0	0	0	0	1	0	0	1	...	0	0	
1	1	0	0	0	0	0	0	0	0	1	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	1	1	0	0	2	0	2	...	0	1	
5	0	0	0	0	0	0	0	0	1	1	...	0	0	
6	0	1	0	0	0	1	0	0	0	0	...	1	0	
7	0	0	0	0	0	0	0	0	0	1	...	0	0	
8	0	0	0	0	0	0	0	0	0	0	...	0	0	

9 rows × 122 columns

In [297]:
```
class_column_train = data.Class[0:9]
class_column_test = data.Class[9:]
class_column_train
class_column_test
```

Out[297]:
```
9     Positive
10    Negative
11    Positive
Name: Class, dtype: object
```

In [298]:
```
bag_of_words_train['Class'] = class_column_train
bag_of_words_train
```

Out[298]:

	13	30	about	accommodated	actually	after	all	also	amount	and	...	want	was	wee
0	0	0	0	0	0	0	1	0	0	1	...	0	0	
1	1	0	0	0	0	0	0	0	0	1	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	1	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	1	1	0	0	2	0	2	...	1	0	
5	0	0	0	0	0	0	0	0	1	1	...	0	0	
6	0	1	0	0	0	1	0	0	0	0	...	0	1	
7	0	0	0	0	0	0	0	0	0	1	...	0	0	
8	0	0	0	0	0	0	0	0	0	0	...	0	0	

9 rows × 123 columns

In [299]:
```
is_positive = bag_of_words_train['Class'] == 'Positive'
positive_df = bag_of_words_train[is_positive]
is_negative = bag_of_words_train['Class'] == 'Negative'
negative_df = bag_of_words_train[is_negative]
negative_df
```

Out[299]:

	13	30	about	accommodated	actually	after	all	also	amount	and	...	want	was	wee
0	0	0	0	0	0	0	1	0	0	1	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	1	1	0	0	2	0	2	...	1	0	
6	0	1	0	0	0	1	0	0	0	0	...	0	1	
8	0	0	0	0	0	0	0	0	0	0	...	0	0	

5 rows × 123 columns

In [300]:
```
positive_prior = 4/9
negative_prior = 5/9
positive_likelihood = []
negative_likelihood = []
for feature in features:
    count = 0
    for item in positive_df[feature]:
        if item != 0:
            count= count + 1 #count number of non zero entries for each word
        positive_likelihood.append(count/len(positive_df))
```

In [301]:
```
for feature in features:
    counter = 0
    for item in negative_df[feature]:
        if item != 0:
            counter = counter + 1
        negative_likelihood.append(counter/len(negative_df)) #build negative
```

In [302]:
```
positive_posterior = []
negative_posterior = []

for i in range(0, len(features)):
    positive_posterior_i = (positive_likelihood[i] * positive_prior)#/ (evide
    positive_posterior.append(positive_posterior_i)

for i in range(0, len(positive_posterior)) :
    if positive_posterior[i] < 0.00001:
        positive_posterior[i] = .000001
```

In [303]:
```
for i in range(0, len(features)):
    negative_posterior_i = (negative_likelihood[i] * negative_prior)#/ (evide
    negative_posterior.append(negative_posterior_i)
for i in range(0, len(negative_posterior)) :
    if negative_posterior[i] < 0.00001:
        negative_posterior[i] = .000001
```

In [304]:
```
bag_of_words_test = pd.DataFrame(X_test[:,:].todense(), columns = features)
```

In [305]:
```
bag_of_words_test['Class'] = ['Positive', 'Negative', 'Positive']
bag_of_words_test
```

Out[305]:

	13	30	about	accommodated	actually	after	all	also	amount	and	...	want	was	wee
0	0	0	0	0	0	0	0	0	0	2	...	0	0	
1	0	0	0	0	0	0	0	0	0	2	...	0	0	
2	0	0	0	0	0	0	1	0	0	2	...	0	0	

3 rows × 123 columns

In [306]:
```
predictions = []
prob_positive = positive_prior
prob_negative = negative_prior
for c in range(0,3):
    index_vector = []
    index = 0
    for item in bag_of_words_test.iloc[c]:
        if item != 0 and item != 'Positive' and item != 'Negative':
            index_vector.append(index) #any word that is present in the revie
        index = index + 1
    #At this point we have built our index vector for a particular row
    for elem in index_vector: #elem represents features index of every presen
        #word = features[elem]
        #first calculate probability that review is positive
        prob_positive = prob_positive * positive_posterior[elem]
        prob_negative = prob_negative * negative_posterior[elem]
    print(str(prob_positive) + "    " + str(prob_negative))
    if prob_positive >= prob_negative:
        predictions.append('Positive')
    else:
        predictions.append('Negative')
```

```
1.505341138527136e-35    1.0453757906438451e-30
6.993987479451151e-97    8.225263339969969e-67
8.634552443766852e-123    1.2536600121886863e-82
```

In [307]:
```
predictions
```

Out[307]:
```
['Negative', 'Negative', 'Negative']
```

In [309]:
```
bag_of_words_test['Predicted_Class'] = predictions
bag_of_words_test
```

Out[309]:

	13	30	about	accommodated	actually	after	all	also	amount	and	...	was	weeks	w
0	0	0	0	0	0	0	0	0	0	2	...	0	0	
1	0	0	0	0	0	0	0	0	0	2	...	0	0	
2	0	0	0	0	0	0	1	0	0	2	...	0	0	

3 rows × 124 columns

In [ ]:

In [ ]: