# NLP For Drugs.com Data Set

## Packages Import

In [102]:

```
### Visualization
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud
sns.color_palette("Blues", as_cmap=True)

### Standard Packages
import numpy as np
import warnings
import nltk
import re
import pandas as pd
pd.set_option('display.max_colwidth', None)
warnings.filterwarnings("ignore")

### NLTK
from nltk.tokenize import TweetTokenizer
from nltk.corpus import stopwords, wordnet
from nltk import pos_tag
from nltk.stem import WordNetLemmatizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.tokenize import RegexpTokenizer
nltk.download('wordnet')
nltk.download('vader_lexicon')
import contractions

### Scikit-Learn
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer, TfidfVect
from sklearn.model_selection import GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.svm import SVC
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix,
                            accuracy_score, f1_score, recall_score, pr

### ImbLearn
from imblearn.pipeline import Pipeline as imbpipeline
from imblearn.over_sampling import SMOTE
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/albertcc/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package vader_lexicon to
```

```
[nltk_data]     /Users/albertcc/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

## Bringing in two .tsv files as test and train

In [103]:
```
# Load in the test and train datasets provided in the data file
data_test = pd.read_csv('data/drugsComTest_raw.tsv', sep='\t')
data_train = pd.read_csv('data/drugsComTrain_raw.tsv', sep='\t')
```

In [104]:
```
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 53766 entries, 0 to 53765
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Unnamed: 0   53766 non-null  int64  
 1   drugName     53766 non-null  object 
 2   condition    53471 non-null  object 
 3   review       53766 non-null  object 
 4   rating       53766 non-null  float64
 5   date         53766 non-null  object 
 6   usefulCount  53766 non-null  int64  
dtypes: float64(1), int64(2), object(4)
memory usage: 2.9+ MB
```

In [105]:
```
data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 161297 entries, 0 to 161296
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   Unnamed: 0   161297 non-null  int64  
 1   drugName     161297 non-null  object 
 2   condition    160398 non-null  object 
 3   review       161297 non-null  object 
 4   rating       161297 non-null  float64
 5   date         161297 non-null  object 
 6   usefulCount  161297 non-null  int64  
dtypes: float64(1), int64(2), object(4)
memory usage: 8.6+ MB
```

## Merge Test and Train dataframes

- The data provided is already split into test and train tsv files. I would like to combine these to not only have more data to work with, but any cleaning could be applied to the merged dataset before splitting into a training and testing set.

In [106]:
```
# Combining both tsv files
merged_df = pd.concat([data_test, data_train], axis=0)
```

In [107]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 215063 entries, 0 to 161296
Data columns (total 7 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   Unnamed: 0   215063 non-null  int64  
 1   drugName     215063 non-null  object 
 2   condition    213869 non-null  object 
 3   review       215063 non-null  object 
 4   rating       215063 non-null  float64
 5   date         215063 non-null  object 
 6   usefulCount  215063 non-null  int64  
dtypes: float64(1), int64(2), object(4)
memory usage: 13.1+ MB
```

In [108]:
```
# Drop first column since these appear to be entry numbers
merged_df = merged_df.drop(merged_df.columns[0],axis=1)
```

In [109]:
```
# Sanity check that the Unnamed column has been dropped
merged_df.head()
```

Out[109]:

| | drugName | condition | review | rating | date | usefulCount |
|---|---|---|---|---|---|---|
| **0** | Mirtazapine | Depression | "I've tried a few antidepressants over the years (citalopram, fluoxetine, amitriptyline), but none of those helped with my depression, insomnia & anxiety. My doctor suggested and changed me onto 45mg mirtazapine and this medicine has saved my life. Thankfully I have had no side effects especially the most common - weight gain, I've actually lost alot of weight. I still have suicidal thoughts but mirtazapine has saved me." | 10.0 | February 28, 2012 | 22 |

| | | | | | | |
|---|---|---|---|---|---|---|
| **1** | Mesalamine | Crohn's Disease, Maintenance | "My son has Crohn's disease and has done very well on the Asacol. He has no complaints and shows no side effects. He has taken as many as nine tablets per day at one time. I've been very happy with the results, reducing his bouts of diarrhea drastically." | 8.0 | May 17, 2009 | 17 |
| **2** | Bactrim | Urinary Tract Infection | "Quick reduction of symptoms" | 9.0 | September 29, 2017 | 3 |
| **3** | Contrave | Weight Loss | "Contrave combines drugs that were used for alcohol, smoking, and opioid cessation. People lose weight on it because it also helps control over-eating. I have no doubt that most obesity is caused from sugar/carb addiction, which is just as powerful as any drug. I have been taking it for five days, and the good news is, it seems to go to work immediately. I feel hungry before I want food now. I really don't care to eat; it's just to fill my stomach. Since I have only been on it a few days, I don't know if I've lost weight (I don't have a scale), but my clothes do feel a little looser, so maybe a pound or two. I'm hoping that after a few months on this medication, I will develop healthier habits that I can continue without the aid of Contrave." | 9.0 | March 5, 2017 | 35 |
| **4** | Cyclafem 1 / 35 | Birth Control | "I have been on this birth control for one cycle. After reading some of the reviews on this type and similar birth controls I was a bit apprehensive to start. Im giving this birth control a 9 out of 10 as I have not been on it long enough for a 10. So far I love this birth control! My side effects have been so minimal its like Im not even on birth control! I have experienced mild headaches here and there and some nausea but other than that ive been feeling great! I got my period on cue on the third day of the inactive pills and I had no idea it was coming because I had zero pms! My period was very light and I barely had any cramping! I had unprotected sex the first month and obviously didn't get pregnant so I'm very pleased! Highly recommend" | 9.0 | October 22, 2015 | 4 |

- Noticed how 'condition' has some missing values, but other columns are fine

In [110]:
```
# Drop null values that are in 'condition'

merged_df = merged_df.dropna(subset=['condition'])
```

In [111]:
```
merged_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 213869 entries, 0 to 161296
Data columns (total 6 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   drugName     213869 non-null  object 
 1   condition    213869 non-null  object 
 2   review       213869 non-null  object 
 3   rating       213869 non-null  float64
 4   date         213869 non-null  object 
 5   usefulCount  213869 non-null  int64  
dtypes: float64(1), int64(1), object(4)
memory usage: 11.4+ MB
```

- Reduces the dataset to 213,869 values

In [112]:
```
# Want to see which conditions are the most frequent

merged_df['condition'].value_counts()
```

Out[112]:
```
Birth Control                                        38436
Depression                                           12164
Pain                                                  8245
Anxiety                                               7812
Acne                                                  7435
                                                     ...  
Steroid Responsive Inflammatory Conditions               1
48</span> users found this comment helpful.              1
Lyme Disease, Erythema Chronicum Migrans                 1
79</span> users found this comment helpful.              1
Rat-bite Feve                                            1
Name: condition, Length: 916, dtype: int64
```

## Looking at unique drugs under 'Birth Control' condition

In [113]:
```
# Focus will be seeing which birth control drugs there are

merged_df['drugName'][merged_df['condition'] =='Birth Control'].nuniqu
```

Out[113]:
```
181
```

In [114]:
```
# Want to find which kinds of birth control are the most common

merged_df['drugName'][merged_df['condition'] =='Birth Control'].value_
```

Out[114]:
```
Etonogestrel                          4394
Ethinyl estradiol / norethindrone     3081
Levonorgestrel                        2884
Nexplanon                             2883
Ethinyl estradiol / levonorgestrel    2107
                                      ...
Loestrin 21 1.5 / 30                     1
Pirmella 1 / 35                          1
Norlyda                                  1
Emoquette                                1
Larin 24 Fe                              1
Name: drugName, Length: 181, dtype: int64
```

In [115]:
```
# Select the the top 7 birth controls since we want to focus on these
bc_drugs = ['Etonogestrel', 'Ethinyl estradiol / norethindrone', 'Nexp
            'Ethinyl estradiol / norgestimate', 'Implanon']

bc_data = merged_df[merged_df['drugName'].isin(bc_drugs)]
```

In [116]:
```
bc_data.head()
```

| | | | | | | |
|---|---|---|---|---|---|---|
| **42** | Levonorgestrel | Emergency Contraception | approximately 4 days before ovulation. This means I was fertile. \r\nwhen we noticed that the condom broke, we immediately bought Plan B One Step and was taken about 45 minutes or an hour later. \r\nThe sooner you take the greater the chances it will work.\r\nSo I waited anxiously, praying I get my period. I felt side effects such as fatigue, bloated, nauseous, cramps, and moody. I was that these might be this pregnancy symptoms. \r\nI waited exactly two weeks and I got my period. I was the happiest person ever. My period was here early by approximately 3-4 days. Good luck."<br><br>"I've had this birth | 10.0 | December 28, 2013 | 20 |

In [117]:
```
bc_data['drugName'].value_counts()
```

Out[117]:
```
Levonorgestrel                      4896
Etonogestrel                        4402
Ethinyl estradiol / norethindrone   3619
Nexplanon                           2892
Ethinyl estradiol / norgestimate    2682
Ethinyl estradiol / levonorgestrel  2400
Implanon                            1506
Name: drugName, dtype: int64
```

- Wanted to only include in our dataset the top 7 drugs with condition = birth control, however when we filtered for these drugs we see additional conditions were selected

In [118]:
```
bc_data['condition'].value_counts()
```

Out[118]:
```
Birth Control                                    18942
Emergency Contraception                           1651
Abnormal Uterine Bleeding                          812
Acne                                               439
Endometriosis                                      178
Menstrual Disorders                                124
Ovarian Cysts                                      106
Polycystic Ovary Syndrome                           89
Not Listed / Othe                                   19
Premenstrual Syndrome                               11
0</span> users found this comment helpful.           8
Postmenopausal Symptoms                             6
2</span> users found this comment helpful.           4
8</span> users found this comment helpful.           2
1</span> users found this comment helpful.           2
4</span> users found this comment helpful.           1
Gonadotropin Inhibition                             1
3</span> users found this comment helpful.           1
9</span> users found this comment helpful.           1
Name: condition, dtype: int64
```

In [119]:
```
# Let's try to get rid of these conditions that took in the 'Useful' r
bc_data = bc_data[~bc_data['condition'].str.contains('comment')]
```

In [120]:
```
bc_data['condition'].value_counts()
```

Out[120]:
```
Birth Control                18942
Emergency Contraception       1651
Abnormal Uterine Bleeding      812
Acne                           439
Endometriosis                  178
Menstrual Disorders            124
Ovarian Cysts                  106
Polycystic Ovary Syndrome       89
Not Listed / Othe               19
Premenstrual Syndrome           11
Postmenopausal Symptoms          6
Gonadotropin Inhibition          1
Name: condition, dtype: int64
```

In [121]:
```
bc_data['drugName'].value_counts()
```

Out[121]:
```
Levonorgestrel                      4896
Etonogestrel                        4402
Ethinyl estradiol / norethindrone   3619
Nexplanon                           2883
Ethinyl estradiol / norgestimate    2682
Ethinyl estradiol / levonorgestrel  2400
Implanon                            1496
Name: drugName, dtype: int64
```

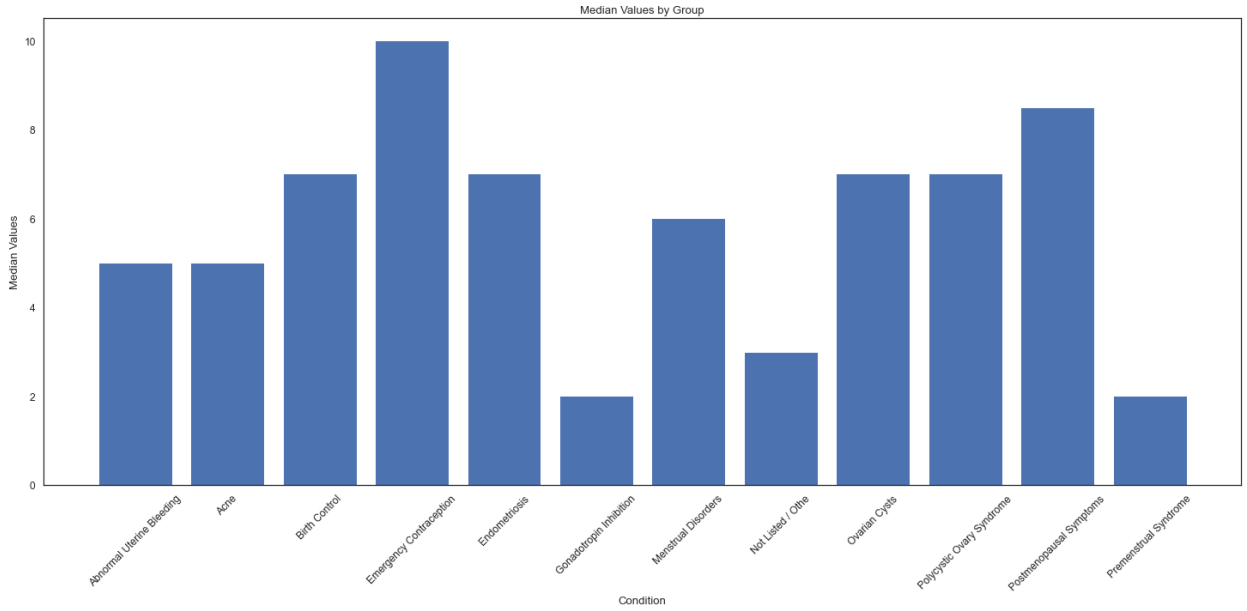
In [122]:
```
bc_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22378 entries, 42 to 161273
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   drugName     22378 non-null  object 
 1   condition    22378 non-null  object 
 2   review       22378 non-null  object 
 3   rating       22378 non-null  float64
 4   date         22378 non-null  object 
 5   usefulCount  22378 non-null  int64  
dtypes: float64(1), int64(1), object(4)
memory usage: 1.2+ MB
```

In [123]:
```
# Create new column called 'sentiment' that will have the target varia
bc_data['sentiment'] = ['Positive' if x > 7.0 else 'Negative' for x in
```

## Analyze the median rating based on condition

In [124]:
```
median_values = bc_data.groupby('condition')['rating'].median()

# Create bar chart
fig, ax = plt.subplots(figsize = (25, 10))
ax.bar(median_values.index, median_values.values)
plt.xticks(rotation=45)

# Set axis labels and title
ax.set_xlabel('Condition')
ax.set_ylabel('Median Values')
ax.set_title('Median Values by Group')

# Show plot
plt.show()
```




In [125]:
```
bc_data.loc[bc_data['condition'] == 'Emergency Contraception'].rating.
```

Out[125]:
```
10.0    1025
9.0      216
1.0      155
8.0      106
7.0       43
5.0       43
6.0       27
4.0       14
3.0       13
2.0        9
Name: rating, dtype: int64
```

## Create columns that count emphasis and capital letters in text, as this could express sentiment

In [126]:
```
# Creating a 'punc_emphasis' column that scores how many exclamation p

bc_data['punc_emphasis'] = bc_data['review'].apply(lambda x: sum([1 fo

# Creating a 'capt_emphasis' column that scores how many capitalized w

bc_data['capt_emphasis'] = bc_data['review'].apply(lambda x: sum([1 fo
```

In [127]:
```
bc_data.head()
```

Out[127]:

| | drugName | condition | review | rating | date | usefulCount | sentiment | pu |
|---|---|---|---|---|---|---|---|---|
| | | | "I want to share my experience to possibly ease some of the ladies out there.\r\nThe incident occur in Nov 18 (the condom broke) and it was 4 days after my period ended and approximately 4 days before ovulation. | | | | | |

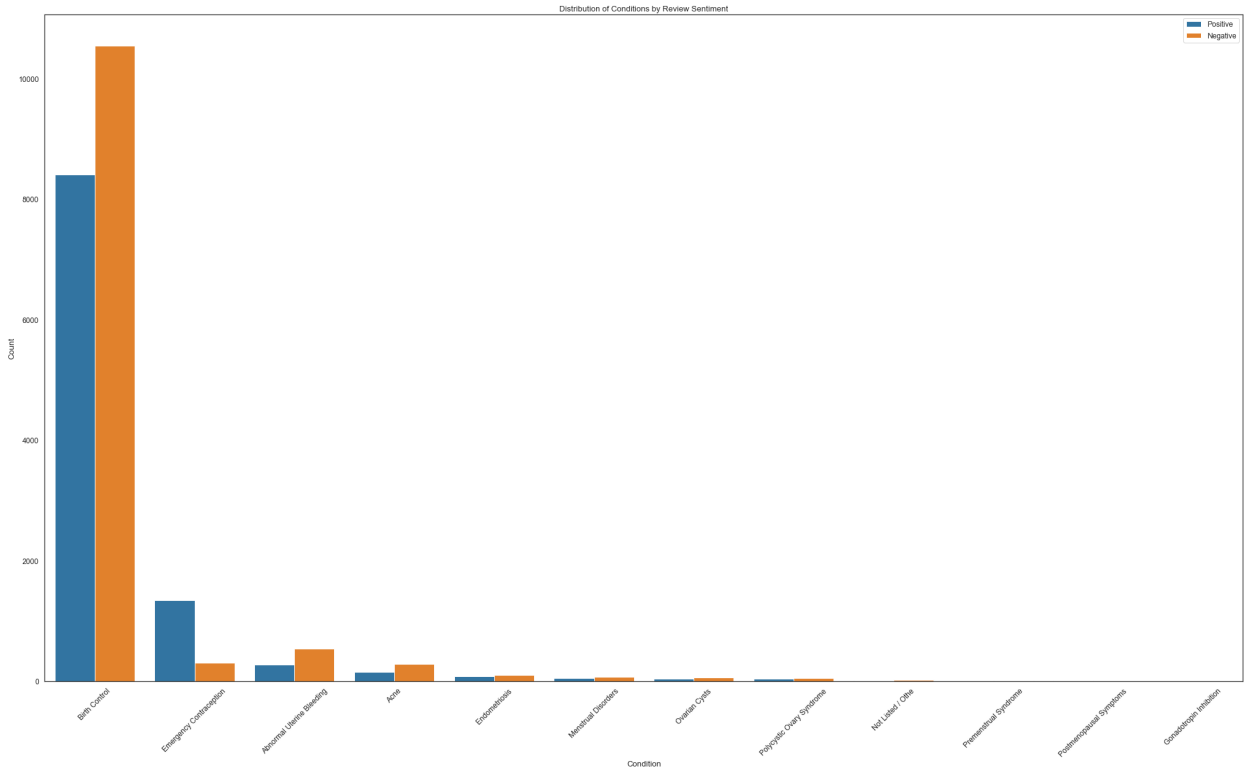
In [128]:
```
bc_data['sentiment'].value_counts(normalize=True)
```

Out[128]:
```
Negative    0.535526
Positive    0.464474
Name: sentiment, dtype: float64
```

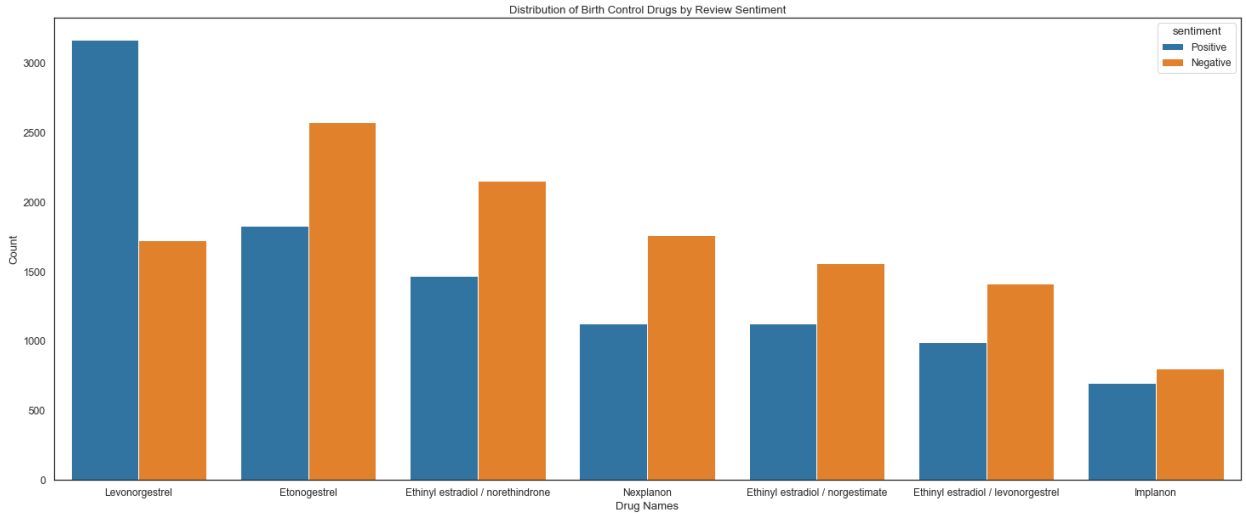
## Sentiment Analysis Against Condition

- Within the conditions we have selected, how do the reviews look pertaining to each condition?

In [129]:
```
# Let's try plotting sentiment against groups

fig = plt.figure(figsize = (35, 20))

hue_order = ['Positive', 'Negative']
sns.countplot(x='condition', hue='sentiment', data=bc_data, hue_order=

plt.xticks(rotation=45)

sns.set(style='white', font_scale=1.1)

plt.legend(loc='upper right')
plt.xlabel('Condition')
plt.ylabel('Count')
plt.title('Distribution of Conditions by Review Sentiment');
```



In [130]:
```
# Let's try plotting sentiment against groups

fig = plt.figure(figsize = (25, 10))


sns.countplot(x='drugName', hue='sentiment', palette='tab10', data=bc_

# plt.xticks(rotation=45)

sns.set(style='white', font_scale=1.1)

plt.xlabel('Drug Names')
plt.ylabel('Count')
plt.title('Distribution of Birth Control Drugs by Review Sentiment');

fig.savefig('images/distribution_bc_review_sentiment.png', dpi=300)
```




## Can we do anything with 'UsefulCount'?

In [131]:
```
bc_data['usefulCount'].value_counts(bins=3)
```

Out[131]:
```
(-1.248, 415.667]     22371
(415.667, 831.333]        6
(831.333, 1247.0]         1
Name: usefulCount, dtype: int64
```

In [132]:
```
bc_data['usefulCount'].min()
```

Out[132]:
```
0
```

In [133]:
```
bc_data['usefulCount'].describe()
```

Out[133]:
```
count    22378.000000
mean         8.008535
std         18.501084
min          0.000000
25%          2.000000
50%          4.000000
75%          9.000000
max       1247.000000
Name: usefulCount, dtype: float64
```

- Not sure if this is too useful of a feature, maybe we could filter the reviews that were found useful above a certain threshold to take in user input.

## After research on birth controls, wanted to read what reviews are saying

In [134]:
```
# Look at the reviews for Levonogestrel
bc_data[bc_data['drugName'] == 'Levonorgestrel']
```

Out[134]:

| | drugName | condition | review | rating | date | usefulCount | sentim |
|---|---|---|---|---|---|---|---|
| | | | "I want to share my experience to possibly ease some of the ladies out there.\r\nThe incident occur in Nov 18 (the condom broke) and it was 4 days after my period ended and approximately 4 days before ovulation. This means I was fertile. \r\nwhen we noticed that | | | | |

## Binning the years these reviews were written into two groups to see if there's a difference over time

In [135]:
```
# Convert date to date time
bc_data['date'] = pd.to_datetime(bc_data['date'])
```

In [136]:
```
# Look at the minimum and maximum dates
bc_data['date'].describe()
```

Out[136]:
```
count                   22378
unique                   3014
top       2015-10-21 00:00:00
freq                       38
first     2008-02-27 00:00:00
last      2017-12-12 00:00:00
Name: date, dtype: object
```

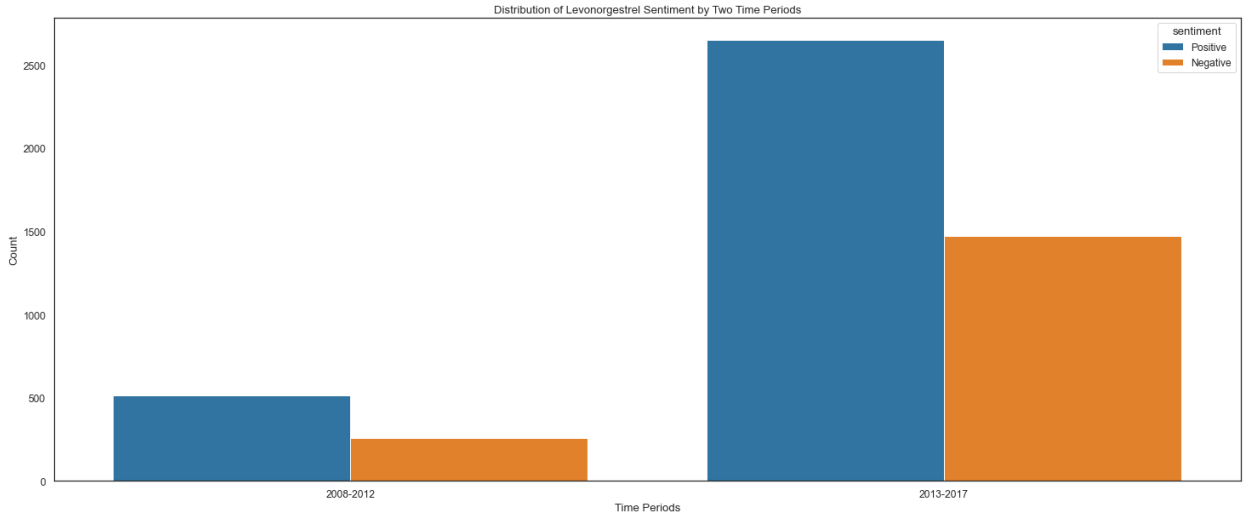
In [137]:
```
bc_data['date'].value_counts(bins=2)
```

Out[137]:
```
(2013-01-19, 2017-12-12]                          18297
(2008-02-23 10:10:33.599999999, 2013-01-19]        4081
Name: date, dtype: int64
```

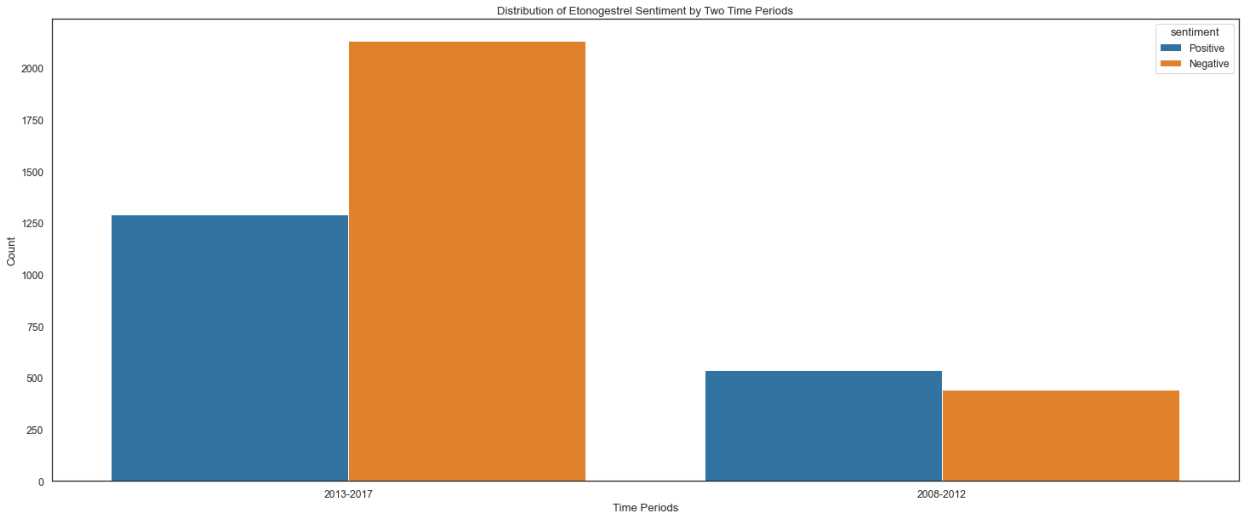
In [138]:
```
# Create new column called 'date_column' that will have grouped time r
bc_data['date_column'] = ['2013-2017' if x.year > 2013 else '2008-2012
```

## Can we see the difference in reviews of these drugs over time?

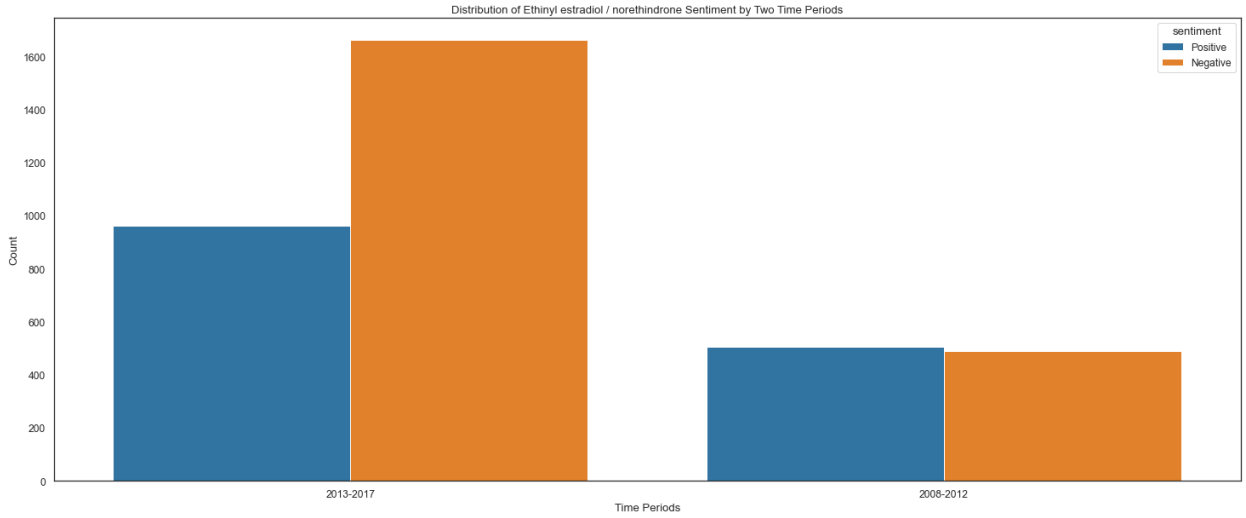
In [139]:
```
# Let's try plotting sentiment of Levonorgestrel against the two assig

fig = plt.figure(figsize = (25, 10))

sns.countplot(x='date_column', hue='sentiment', data=bc_data[bc_data['

# plt.xticks(rotation=45)

plt.xlabel('Time Periods')
plt.ylabel('Count')
plt.title('Distribution of Levonorgestrel Sentiment by Two Time Period
```



In [140]:
```
# Let's try plotting sentiment of Etonogestrel against the two assigne

fig = plt.figure(figsize = (25, 10))

sns.countplot(x='date_column', hue='sentiment', data=bc_data[bc_data['

# plt.xticks(rotation=45)
plt.xlabel('Time Periods')
plt.ylabel('Count')
plt.title('Distribution of Etonogestrel Sentiment by Two Time Periods'

fig.savefig('images/etonogestrel_sentiment_time.png', dpi=300)
```



In [141]:
```
# Let's try plotting sentiment of Ethinyl estradiol / norethindrone ag

fig = plt.figure(figsize = (25, 10))

sns.countplot(x='date_column', hue='sentiment', data=bc_data[bc_data['

# plt.xticks(rotation=45)
plt.xlabel('Time Periods')
plt.ylabel('Count')
plt.title('Distribution of Ethinyl estradiol / norethindrone Sentiment
```




## Cleaning Text Reviews

In [142]:
```
# Create function that will lowercase the text

def lower_case(text):
    text = text.lower()
    return text

# Create function to remove the html apostrophes in the text

def apostrophe(text):
    text = text.replace('&#039;', '\'')
    return text

# Want to expand the contractions so we can see if these words have im

def fixcontractions(text):
    text = contractions.fix(text)
    return text

# Create a function that uses a regex tokenizer to remove punctuation
```

```
def remove_punctuation(text):
    tokenizer = RegexpTokenizer(r'\w+\'?\w+')
    text = tokenizer.tokenize(text)
    text = ' '.join(text)
    return text

# Remove stopwords from the reviews

def remove_stopwords(text, stop_words_list = set(stopwords.words('engl
    text = text.split()
    text = [word for word in text if word not in stop_words_list]
    text = ' '.join(text)
    return text

# Create a function that lemmatizes words

def lemmatize(text):
    lemmatizer = WordNetLemmatizer()
    text = text.split()
    text = [lemmatizer.lemmatize(word) for word in text]
    text = ' '.join(text)
    return text

def clean_text(text):
    text = lower_case(text)
    text = apostrophe(text)
    text = fixcontractions(text)
    text = remove_punctuation(text)
    text = remove_stopwords(text)
    text = lemmatize(text)
    return text
```

In [143]:
```
# Manually testing the contractions.fix function
contractions.fix("I've aren't Tim's got a lovely bunch of coconuts")
```

Out[143]: "I have are not Tim's got a lovely bunch of coconuts"

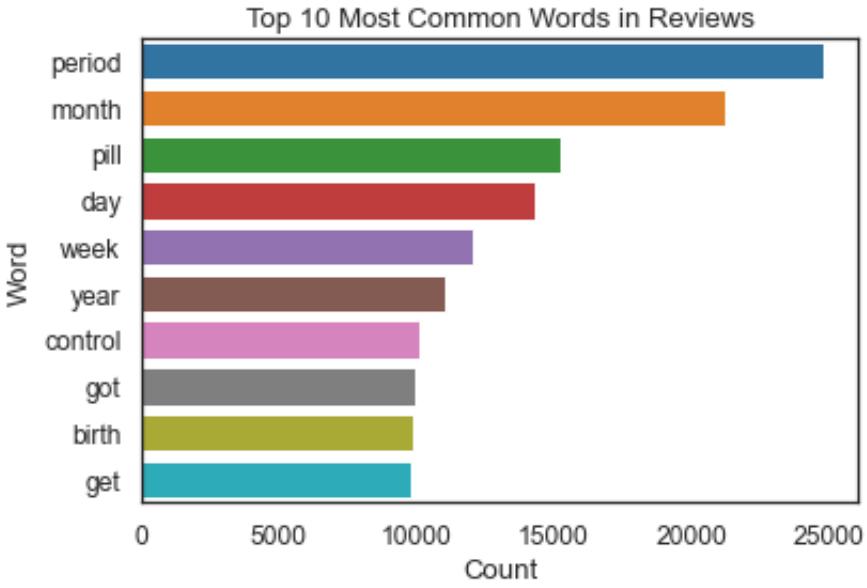
In [144]:
```
# Original review text location 6
bc_data['review'][14]
```

Out[144]:
```
'"Started Nexplanon 2 months ago because I have a minimal amount of c
ontraception&#039;s I can take due to my inability to take the hormon
e that is used in most birth controls. I&#039;m trying to give it tim
e because it is one of my only options right now. But honestly if I h
ad options I&#039;d get it removed.\r\nI&#039;ve never had acne probl
ems in my life, and immediately broke out after getting it implanted.
Sex drive is completely gone, and I used to have sex with my boyfrien
d a few days a week, now its completely forced and not even fun for m
e anymore. I mean I&#039;m on birth control because I like having sex
but don&#039;t want to get pregnant, why take a birth control that ta
kes away sex? Very unhappy and hope that I get it back with time or I
&#039;m getting it removed."'
```

In [145]:
```
# Testing one of the reviews to see what it is doing to the text, as a
clean_text(bc_data['review'][14])
```

Out[145]:
```
"started nexplanon month ago minimal amount contraception's take due
inability take hormone used birth control trying give time one option
right honestly option would get removed never acne problem life immed
iately broke getting implanted sex drive completely gone used sex boy
friend day week completely forced even fun anymore mean birth control
like sex want get pregnant take birth control take away sex unhappy h
ope get back time getting removed"
```

In [146]:
```
### Plotting the top 10 most common words in the 'text' column in an s

text = ' '.join(bc_data['review'])
text = clean_text(text)
text = text.split()

freq = pd.Series(text).value_counts()[:10]
freq = freq.to_frame()
freq = freq.reset_index()
freq.columns = ['word', 'count']
freq = freq.sort_values(by='count', ascending=False)

fig = plt.figure(figsize=(6,4))
sns.barplot(x='count', y='word', data=freq, palette='tab10')
plt.xlabel('Count')
plt.ylabel('Word')
plt.title('Top 10 Most Common Words in Reviews')
plt.show()
```




# Modeling

In [147]:
```
bc_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22378 entries, 42 to 161273
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   drugName       22378 non-null  object        
 1   condition      22378 non-null  object        
 2   review         22378 non-null  object        
 3   rating         22378 non-null  float64       
 4   date           22378 non-null  datetime64[ns]
 5   usefulCount    22378 non-null  int64         
 6   sentiment      22378 non-null  object        
 7   punc_emphasis  22378 non-null  int64         
 8   capt_emphasis  22378 non-null  int64         
 9   date_column    22378 non-null  object        
dtypes: datetime64[ns](1), float64(1), int64(3), object(5)
memory usage: 2.5+ MB
```

In [148]:
```
# How is the distribution of sentiment amongst the Birth Control
bc_data['sentiment'].value_counts(normalize=True)
```

Out[148]:
```
Negative    0.535526
Positive    0.464474
Name: sentiment, dtype: float64
```

## First Simple Model - Count Vectorizer / Logistic Regression / No Features

In [149]:
```
# Using review text against sentiment analysis

X1 = bc_data['review']
y1 = bc_data['sentiment']

X_train_1, X_test_1, y_train_1, y_test_1 = train_test_split(X1, y1, te

# For Train Set, apply clean_text function

X_train_1 = X_train_1.apply(clean_text)

### Train - Tokenize the training data with a simple split of words, a

X_train_1 = X_train_1.apply(lambda x: x.split())
X_train_1 = X_train_1.map(' '.join)

### Train - Vectorize the training data using CountVectorizer

cv = CountVectorizer()
X_train_1 = cv.fit_transform(X_train_1)

### Train - Fit training data to Logistic Regression Model

logit = LogisticRegression()
logit.fit(X_train_1, y_train_1)

### VALIDATION - Perform a cross validation on the logistic regression

scores = cross_val_score(logit, X_train_1, y_train_1, cv=5)
print('Cross Validation Scores: ', scores)
print('Mean Cross Validation Score: ', scores.mean())
```

```
Cross Validation Scores:  [0.82658475 0.83105278 0.82513966 0.8184357
5 0.82234637]
Mean Cross Validation Score:  0.8247118638250607
```

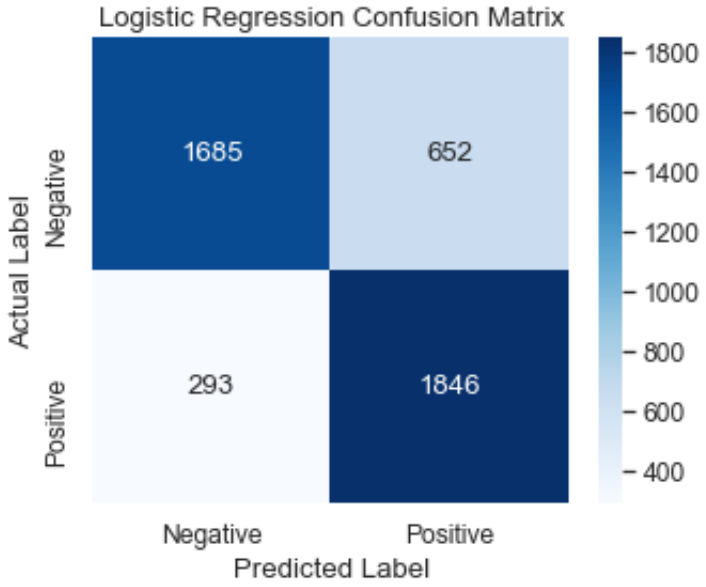
In [150]:
```
# Logistic Regression Test Set Preprocessing

X_test_1_logit = X_test_1.apply(clean_text)
X_test_1_logit = X_test_1.apply(lambda x: x.split())
X_test_1_logit = X_test_1.map(' '.join)
X_test_1_logit = cv.transform(X_test_1)
```

In [151]:
```
logit_pred = logit.predict(X_test_1_logit)
```

In [152]:
```
print('Logistic Regression Accuracy: ', accuracy_score(y_test_1, logit
print('Logistic Regression F1 Score: ', f1_score(y_test_1, logit_pred,
print('Logistic Regression Precision Score: ', precision_score(y_test_
print('Logistic Regression Recall Score: ', recall_score(y_test_1, log
```

```
Logistic Regression Accuracy:  0.7888739946380697
Logistic Regression F1 Score:  0.7882641149643028
Logistic Regression Precision Score:  0.7979275466120083
Logistic Regression Recall Score:  0.7888739946380697
```

In [153]:
```
# Get scores for the sentiments individually to see

# Quick look at decision matrix for our first model:

# Logistic Regression Confusion Matrix

cm = confusion_matrix(y_test_1, logit_pred)
cm_df = pd.DataFrame(cm, index=['Negative', 'Positive'], columns=['Neg

fig_cm1 = plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()
fig_cm1.savefig('images/logistic_regression_confusion_matrix.png', dpi
```



In [154]:
```
# Quick math check to see if these add up to the train split. It does!
1685 + 1846 + 293 + 652
```

Out[154]: 4476

## Second Model - TFIDF Vectorizer / Logistic Regression / No Features

- Want to test if TFIDF Vectorizer makes a difference compared to Count Vectorizer

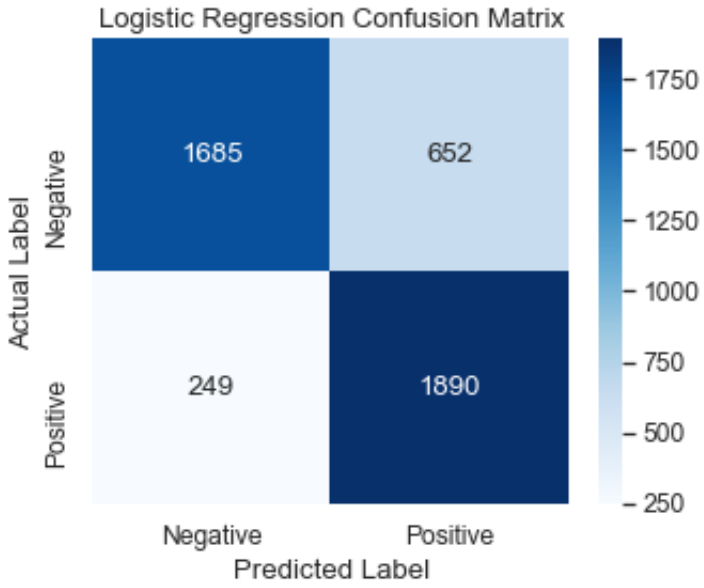
In [155]:
```
X1_tfidf = bc_data['review']
y1_tfidf = bc_data['sentiment']

X_train_1_tfidf, X_test_1_tfidf, y_train_1_tfidf, y_test_1_tfidf = tra

# For Train Set, apply clean_text function

X_train_1_tfidf = X_train_1_tfidf.apply(clean_text)

### Train - Tokenize the training data with a simple split of words, a

X_train_1_tfidf = X_train_1_tfidf.apply(lambda x: x.split())
X_train_1_tfidf = X_train_1_tfidf.map(' '.join)

### Train - Vectorize the training data using TFIDFVectorizer

tfidf = TfidfVectorizer()
X_train_1_tfidf = tfidf.fit_transform(X_train_1_tfidf)

### Train - Fit training data to Logistic Regression Model

logit_tfidf = LogisticRegression()
logit_tfidf.fit(X_train_1_tfidf, y_train_1_tfidf)

### VALIDATION - Perform a cross validation on the decision tree class

scores = cross_val_score(logit_tfidf, X_train_1_tfidf, y_train_1_tfidf
print('Cross Validation Scores: ', scores)
print('Mean Cross Validation Score: ', scores.mean())
```

```
Cross Validation Scores:  [0.83216978 0.83524155 0.83463687 0.8220670
4 0.8273743 ]
Mean Cross Validation Score:  0.8302979099811388
```

In [156]:
```
# Logistic Regression Test (TFIDF) Set Preprocessing

X_test_1_tfidf = X_test_1_tfidf.apply(clean_text)
X_test_1_tfidf = X_test_1_tfidf.apply(lambda x: x.split())
X_test_1_tfidf = X_test_1_tfidf.map(' '.join)
X_test_1_tfidf = cv.transform(X_test_1_tfidf)
```

In [157]:
```
logit_pred_tfidf = logit_tfidf.predict(X_test_1_tfidf)
```

In [158]:
```
### Logistic Regression Confusion Matrix w/ TFIDF

cm = confusion_matrix(y_test_1_tfidf, logit_pred_tfidf)
cm_df = pd.DataFrame(cm, index=['Negative', 'Positive'], columns=['Neg

fig_cm1b = plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
plt.title('Logistic Regression Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()
fig_cm1b.savefig('images/logistic_regression_confusion_matrix_tfidf.pr
```



In [159]:
```
print('Logistic Regression (TFIDF) Accuracy: ', accuracy_score(y_test_
print('Logistic Regression (TFIDF) F1 Score: ', f1_score(y_test_1_tfid
print('Logistic Regression (TFIDF) Precision Score: ', precision_score
print('Logistic Regression (TFIDF) Recall Score: ', recall_score(y_tes
```

```
Logistic Regression (TFIDF) Accuracy:  0.798704200178731
Logistic Regression (TFIDF) F1 Score:  0.7978723877450785
Logistic Regression (TFIDF) Precision Score:  0.8102055684985682
Logistic Regression (TFIDF) Recall Score:  0.798704200178731
```

- We see here that our Logistic regression model using TFIDF has a higher accuracy than the model with Count Vector.
- Originally this was a lower accuracy due to the fact that we tried using three classes to predict sentiment that was imbalanced between the classes.
- In this version we made only a positive and negative sentiment for Birth Control and it was more balanced.

**Grid Search for best Logistic Regression parameters**

In [160]:
```
param_grid = {'C': [1, 10, 100, 1000, 10000],'penalty': ['none', 'l1',
grid = GridSearchCV(logit, param_grid, cv=5, scoring='accuracy', n_job
grid.fit(X_train_1, y_train_1)
print('Best Parameters: ', grid.best_params_)
```

```
Best Parameters:  {'C': 1, 'penalty': 'l2'}
```

- Looks that the default parameters are the best parameters for this model

In [161]:
```
param_grid = {'C': [1, 10, 100, 1000, 10000],'penalty': ['none', 'l1',
grid = GridSearchCV(logit_tfidf, param_grid, cv=5, scoring='accuracy',
grid.fit(X_train_1_tfidf, y_train_1_tfidf)
print('Best Parameters: ', grid.best_params_)
```

```
Best Parameters:  {'C': 1, 'penalty': 'l2'}
```

- Same as above

## Third Model: Count Vectorizer / Multinomial Bayes / No Features

In [162]:
```
# Using the same X_train and y_train from our first models but using a

X1_nb = bc_data['review']
y1_nb = bc_data['sentiment']

X_train_1_nb, X_test_1_nb, y_train_1_nb, y_test_1_nb = train_test_spli

# For Train Set, apply clean_text function

X_train_1_nb = X_train_1_nb.apply(clean_text)

### Train - Tokenize the training data with a simple split of words, a

X_train_1_nb = X_train_1_nb.apply(lambda x: x.split())
X_train_1_nb = X_train_1_nb.map(' '.join)

### Train - Vectorize the training data using CountVectorizer

cv = CountVectorizer()
X_train_1_nb = cv.fit_transform(X_train_1_nb)

# Train - fitting the training data to a Naive Bayes Classifier

nb = MultinomialNB()
nb.fit(X_train_1_nb, y_train_1_nb)

# Validation - Performing cross validation on the Naive Bayes Classifi

scores = cross_val_score(nb, X_train_1_nb, y_train_1_nb, cv=5)
print('Cross Validation Scores: ', scores)
print('Mean Cross Validation Score: ', scores.mean())
```

```
Cross Validation Scores:  [0.82016197 0.82435074 0.81564246 0.8047486
0.81061453]
Mean Cross Validation Score:  0.8151036585080476
```

### Grid Search for best conditions with NB

In [163]:
```
param_grid = {'alpha': [0.1, 0.5, 1, 2, 5, 20, 50]}
grid = GridSearchCV(nb, param_grid, cv=5, scoring='accuracy', n_jobs=-
grid.fit(X_train_1_nb, y_train_1_nb)
print('Best Parameters: ', grid.best_params_)
```

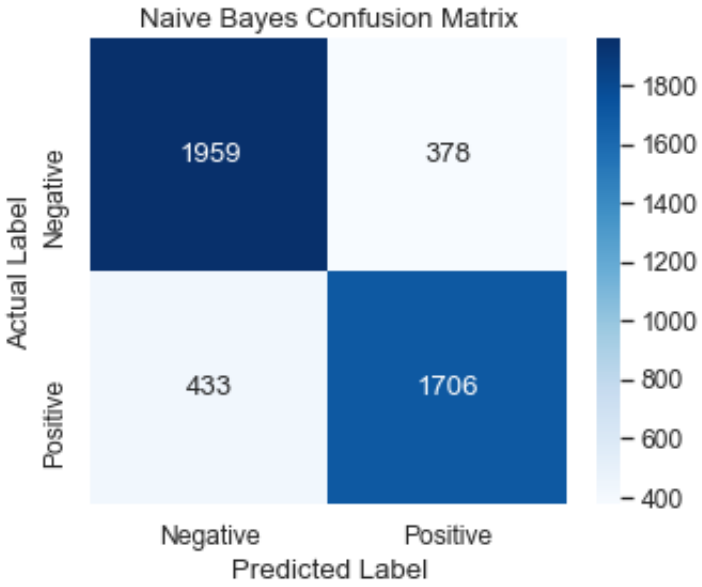
```
Best Parameters:  {'alpha': 0.5}
```

- Default parameter is 1.0 for alpha, we can try seeing what the results are with 0.5

In [164]:
```
# Naive Bayes Test Set Preprocessing

X_test_1_nb = X_test_1_nb.apply(clean_text)
X_test_1_nb = X_test_1_nb.apply(lambda x: x.split())
X_test_1_nb = X_test_1_nb.map(' '.join)
X_test_1_nb = cv.transform(X_test_1_nb)
```

In [165]:
```
nb_pred_cv = nb.predict(X_test_1_nb)
```

In [166]:
```
### Naive Bayes Confusion Matrix w/ CountVectorizer

cm = confusion_matrix(y_test_1_nb, nb_pred_cv)
cm_df = pd.DataFrame(cm, index=['Negative', 'Positive'], columns=['Neg

fig_cm3 = plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
plt.title('Naive Bayes Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()
fig_cm3.savefig('images/mnb_confusion_matrix.png', dpi=300)
```




In [167]:
```
print('Naive Bayes Accuracy: ', accuracy_score(y_test_1_nb, nb_pred_cv
print('Naive Bayes F1 Score: ', f1_score(y_test_1_nb, nb_pred_cv, aver
print('Naive Bayes Precision Score: ', precision_score(y_test_1_nb, nb
print('Naive Bayes Recall Score: ', recall_score(y_test_1_nb, nb_pred_
```

```
Naive Bayes Accuracy:  0.8188114387846291
Naive Bayes F1 Score:  0.8186851910506858
Naive Bayes Precision Score:  0.8188069919574142
Naive Bayes Recall Score:  0.8188114387846291
```

## 4th Model: Count Vectorizer / Logistic Regression / All Features

In [168]:
```
# Additional features to see if it helps improve the model
X2 = bc_data[['review', 'drugName', 'condition', 'punc_emphasis', 'cap
y2 = bc_data['sentiment']

X_train_2, X_test_2, y_train_2, y_test_2 = train_test_split(X2, y2, te

# For Train Set, apply clean_text function

X_train_2['review'] = X_train_2['review'].apply(clean_text)

### Train - Tokenize the training data with a simple split of words, a

X_train_2['review'] = X_train_2['review'].apply(lambda x: x.split())
X_train_2['review'] = X_train_2['review'].map(' '.join)

### Train - Vectorize the training data using CountVectorizer

cv = CountVectorizer()
X_train_2 = cv.fit_transform(X_train_2['review'])

### Train - Fit training data to Logistic Regression Model

logit = LogisticRegression()
logit.fit(X_train_2, y_train_2)

### VALIDATION - Perform a cross validation on the logistic regression

scores = cross_val_score(logit, X_train_2, y_train_2, cv=5)
print('Cross Validation Scores: ', scores)
print('Mean Cross Validation Score: ', scores.mean())
```

```
Cross Validation Scores:  [0.82658475 0.83105278 0.82513966 0.8184357
5 0.82234637]
Mean Cross Validation Score:  0.8247118638250607
```

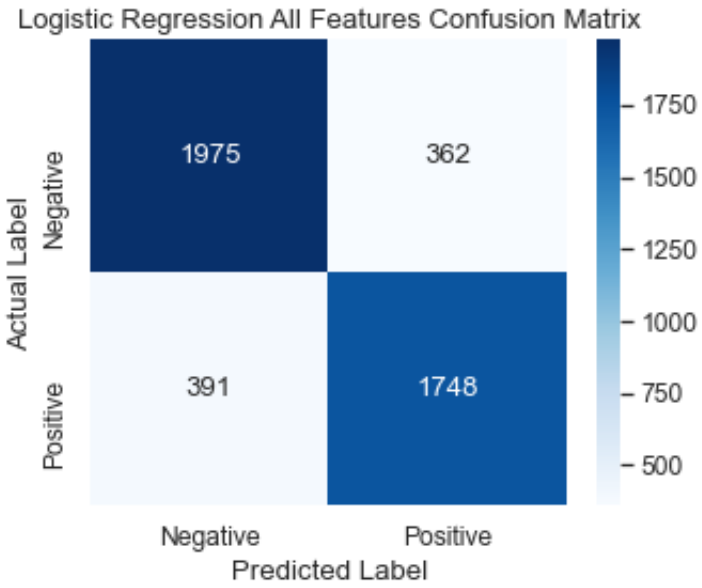
In [169]:
```
logit.coef_
```

Out[169]:
```
array([[ 0.43082765, -0.53249771,  0.15085222, ..., -0.28975559,
        -0.15145837, -0.39983239]])
```

In [170]:
```
# Logistic Regression Test (All Features) Preprocessing

X_test_2['review'] = X_test_2['review'].apply(clean_text)
X_test_2['review'] = X_test_2['review'].apply(lambda x: x.split())
X_test_2['review'] = X_test_2['review'].map(' '.join)
X_test_2 = cv.transform(X_test_2['review'])
```

In [171]:
```
logit_pred_all = logit.predict(X_test_2)
```

In [172]:
```
### Logistic Regression All Features Confusion Matrix w/ CountVectoriz

cm = confusion_matrix(y_test_2, logit_pred_all)
cm_df = pd.DataFrame(cm, index=['Negative', 'Positive'], columns=['Neg

fig_cm4 = plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
plt.title('Logistic Regression All Features Confusion Matrix')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()
fig_cm4.savefig('images/logit_all_confusion_matrix.png', dpi=300)
```




In [173]:
```
print('Logistic Regression All Features Accuracy: ', accuracy_score(y_
print('Logistic Regression All Features F1 Score: ', f1_score(y_test_2
print('Logistic Regression All Features Precision Score: ', precision_
print('Logistic Regression All Features Recall Score: ', recall_score(
```

```
Logistic Regression All Features Accuracy:  0.8317694369973191
Logistic Regression All Features F1 Score:  0.8317140169306872
Logistic Regression All Features Precision Score:  0.8317285793797539
Logistic Regression All Features Recall Score:  0.8317694369973191
```

## 5th Model: Want to see training on Levonogestrel on its own

- The previous models are evaluated on all Birth Control types, but want to see how the model performs on just one of the drugs

In [174]:
```
data_lev = bc_data[bc_data['drugName'] == 'Levonorgestrel']
data_lev.head()
```

Out[174]:

| | drugName | condition | review | rating | date | usefulCount | sentiment | pur |
|---|---|---|---|---|---|---|---|---|
| | | | "I want to share my experience to possibly ease some of the ladies out there.\r\nThe incident occur in Nov 18 (the condom broke) and it was 4 days after my period ended and approximately 4 days before ovulation. This means I was fertile. \r\nwhen we noticed that | | | | | |

In [175]:
```
data_lev['sentiment'].value_counts()
```

Out[175]:
```
Positive    3168
Negative    1728
Name: sentiment, dtype: int64
```

In [176]:
```
data_lev.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4896 entries, 42 to 161257
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype         
---  ------         --------------  -----         
 0   drugName       4896 non-null   object        
 1   condition      4896 non-null   object        
 2   review         4896 non-null   object        
 3   rating         4896 non-null   float64       
 4   date           4896 non-null   datetime64[ns]
 5   usefulCount    4896 non-null   int64         
 6   sentiment      4896 non-null   object        
 7   punc_emphasis  4896 non-null   int64         
 8   capt_emphasis  4896 non-null   int64         
 9   date_column    4896 non-null   object        
dtypes: datetime64[ns](1), float64(1), int64(3), object(5)
memory usage: 420.8+ KB
```

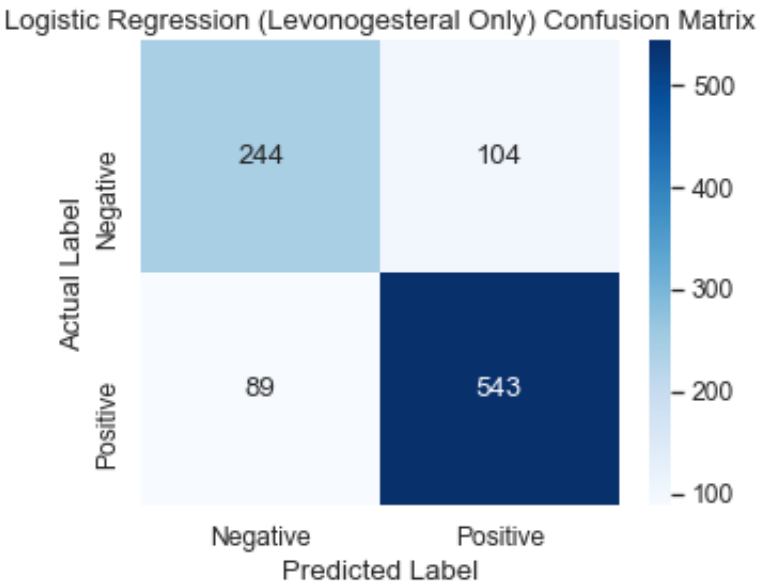
In [177]:
```
# Loading in review vs sentiment for only the Levonogestrel drug

X3 = data_lev['review']
y3 = data_lev['sentiment']

X_train_3, X_test_3, y_train_3, y_test_3 = train_test_split(X3, y3, te

# For Train Set, apply clean_text function

X_train_3 = X_train_3.apply(clean_text)

### Train - Tokenize the training data with a simple split of words, a

X_train_3 = X_train_3.apply(lambda x: x.split())
X_train_3 = X_train_3.map(' '.join)

### Train - Vectorize the training data using CountVectorizer

cv = CountVectorizer()
X_train_3 = cv.fit_transform(X_train_3)

### Train - Fit training data to Logistic Regression Model

logit_lev = LogisticRegression()
logit_lev.fit(X_train_3, y_train_3)

### VALIDATION - Perform a cross validation on the logistic regression

scores = cross_val_score(logit_lev, X_train_3, y_train_3, cv=5)
print('Cross Validation Scores: ', scores)
print('Mean Cross Validation Score: ', scores.mean())
```

```
Cross Validation Scores:  [0.78061224 0.79182631 0.79438059 0.7879948
9 0.81481481]
Mean Cross Validation Score:  0.7939257695415332
```

In [178]:
```
# Logistic Regression Test Set Preprocessing

X_test_3_logit = X_test_3.apply(clean_text)
X_test_3_logit = X_test_3.apply(lambda x: x.split())
X_test_3_logit = X_test_3.map(' '.join)
X_test_3_logit = cv.transform(X_test_3)
```

In [179]:
```
logit_pred_3 = logit_lev.predict(X_test_3_logit)
```

In [180]:
```
### Get scores for the sentiments individually to see

# Quick look at decision matrix for our fifth model:

### Logistic Regression Confusion Matrix

cm = confusion_matrix(y_test_3, logit_pred_3)
cm_df = pd.DataFrame(cm, index=['Negative', 'Positive'], columns=['Neg

fig_cm3 = plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True, fmt='g', cmap='Blues')
plt.title('Logistic Regression (Levonogesteral Only) Confusion Matrix'
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
plt.show()
```




In [181]:
```
print('Logistic Regression (Levonogesteral Only) Accuracy: ', accuracy
print('Logistic Regression (Levonogesteral Only) F1 Score: ', f1_score
print('Logistic Regression (Levonogesteral Only) Precision Score: ', p
print('Logistic Regression (Levonogesteral Only) Recall Score: ', reca
```

```
Logistic Regression (Levonogesteral Only) Accuracy:  0.80306122448979
59
Logistic Regression (Levonogesteral Only) F1 Score:  0.80204713560087
27
Logistic Regression (Levonogesteral Only) Precision Score:  0.8014307
3395447
Logistic Regression (Levonogesteral Only) Recall Score:  0.8030612244
897959
```

## 6th Model - TFIDF / Random Forest Classifier / All Features

In [182]:
```
# Additional features to see if it helps improve the model

X2 = bc_data[['review', 'drugName', 'condition', 'punc_emphasis', 'cap
y2 = bc_data['sentiment']

X_train_4, X_test_4, y_train_4, y_test_4 = train_test_split(X2, y2, te

# For Train Set, apply clean_text function

X_train_4['review'] = X_train_4['review'].apply(clean_text)

### Train - Tokenize the training data with a simple split of words, a

X_train_4['review'] = X_train_4['review'].apply(lambda x: x.split())
X_train_4['review'] = X_train_4['review'].map(' '.join)

### Train - Vectorize the training data using CountVectorizer

tfidf = TfidfVectorizer()
X_train_4 = tfidf.fit_transform(X_train_4['review'])

### Train - Fit training data to Logistic Regression Model

rfc = RandomForestClassifier()
rfc.fit(X_train_4, y_train_4)

### VALIDATION - Perform a cross validation on the logistic regressio

scores = cross_val_score(rfc, X_train_4, y_train_4, cv=5)
print('Cross Validation Scores: ', scores)
print('Mean Cross Validation Score: ', scores.mean())
```

```
Cross Validation Scores:  [0.84222284 0.84333985 0.84329609 0.8469273
7 0.84134078]
Mean Cross Validation Score:  0.843425387559107
```

### Grid Search for Best Conditions for RFC

In [*]:
```
param_grid = {'n_estimators': [200, 300, 400, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']}
grid = GridSearchCV(rfc, param_grid, cv=5, scoring='accuracy', n_jobs=
grid.fit(X_train_4, y_train_4)
print('Best Parameters: ', grid.best_params_)
```

In [*]:
```
X_test_4['review'] = X_test_4['review'].apply(clean_text)
X_test_4['review'] = X_test_4['review'].apply(lambda x: x.split())
X_test_4['review'] = X_test_4['review'].map(' '.join)
X_test_4 = tfidf.transform(X_test_4['review'])
```

## Results of Grid Search Tuning of Hyper Parameters

Logit Results = {'C': 1, 'penalty': 'l2'}

NB Results = {'alpha': 0.5}

RFC results = {'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 300}

## Apply Hyper Parameters Tuning to Models and Evaluate Results

In [*]:
```
# Applying the hyperparameters from the grid search to the Logistic Re

logit = LogisticRegression()
logit.fit(X_train_1_tfidf, y_train_1_tfidf)

### Applying the hyperparameters from the grid search to the Naive Bay

nb = MultinomialNB(alpha=0.5)
nb.fit(X_train_1_nb, y_train_1_nb)

### Applying the hyperparameters from the grid search to the Random Fo

rfc = RandomForestClassifier(criterion='gini', max_depth=8, max_featur
rfc.fit(X_train_4, y_train_4)
```

## Running Models on Test Data and Evaluating Results

In [*]:
```
# Logistic Regression Test Set Predictions

logit_pred = logit.predict(X_test_1_tfidf)

# Multinomial Bayes Test Set Predictions

nb_pred = nb.predict(X_test_1_nb)

# Random Forest Classifier Test Set Predictions

rfc_pred = rfc.predict(X_test_4)
```

In [*]:
```
X_train_1_nb_, X_test_1_nb_, y_train_1_nb_, y_test_1_nb_ = train_test_
```

In [*]:
```
type(X_test_1_nb_)
```

In [*]:
```
X_test_1_nb_.reset_index().drop('index', axis=1)['review'][1]
```

In [*]:
```
nb.predict(X_test_1_nb[1])
```

In [*]:
```
### Logistic Regression Test Set Evaluation

print('Logistic Regression Accuracy: ', accuracy_score(y_test_1_tfidf,
print('Logistic Regression F1 Score: ', f1_score(y_test_1_tfidf, logit
print('Logistic Regression Precision Score: ', precision_score(y_test_
print('Logistic Regression Recall Score: ', recall_score(y_test_1_tfid
```

In [*]:
```
### Naive Bayes Test Set Evaluation

print('Naive Bayes Accuracy: ', accuracy_score(y_test_2, nb_pred))
print('Naive Bayes F1 Score: ', f1_score(y_test_2, nb_pred, average='w
print('Naive Bayes Precision Score: ', precision_score(y_test_2, nb_pr
print('Naive Bayes Recall Score: ', recall_score(y_test_2, nb_pred, av
```

In [*]:
```
### Random Forest Test Set Evaluation

print('Random Forest Accuracy: ', accuracy_score(y_test_4, rfc_pred))
print('Random Forest F1 Score: ', f1_score(y_test_4, rfc_pred, average
print('Random Forest Precision Score: ', precision_score(y_test_4, rfc
print('Random Forest Recall Score: ', recall_score(y_test_4, rfc_pred,
```

## Deployment of Model: Making Word Clouds and Visualizations of Predicted Sentiment

In [*]:
```
# Create function that takes in our bc_data that contains the review a

def create_word_clouds(bc_data):
    bc_data['review'] = bc_data['review'].apply(clean_text)
    negative = bc_data.loc[bc_data['sentiment'] == 'Negative']
    positive = bc_data.loc[bc_data['sentiment'] == 'Positive']
    negative_text = ' '.join(negative['review'])
    positive_text = ' '.join(positive['review'])
    negative_wordcloud = WordCloud(width=600, height=400, collocations
                                   background_color='white').generate(n
    positive_wordcloud = WordCloud(width=600, height=400, collocations
                                   background_color='white').generate(p
    fig, ax = plt.subplots(1, 2, figsize=(20, 10))
    fig.tight_layout(pad=-1.0)
    ax[0].imshow(negative_wordcloud, interpolation='bilinear')
    ax[0].set_title('Negative Sentiment \n', fontsize=28)
    ax[0].axis('off')
    ax[1].imshow(positive_wordcloud, interpolation='bilinear')
    ax[1].set_title('Positive Sentiment \n', fontsize=28)
    ax[1].axis('off')
    fig.subplots_adjust(wspace=0.5)
    plt.show()

    fig.savefig('images/word_clouds', dpi=300)
```

In [*]:
```
create_word_clouds(bc_data)
```

## Deployment of Model: Create Function for Testing Data with Model

In [*]:
```
def multinomial_bayes(data):

    # Load in data, if it is the same format we have the tab seperatio
```

```
df = pd.read_csv(data, sep='\t')

# Data cleaning

df = df.dropna(subset=['condition'])

# Function for picking the top 7 birth controls

bc_drugs_nb = ['Etonogestrel', 'Ethinyl estradiol / norethindrone'
       'Ethinyl estradiol / norgestimate', 'Implanon']

bc_data_nb = df[df['drugName'].isin(bc_drugs_nb)]

# Further cleaning of conditions that were not correct

bc_data_nb = bc_data_nb[~bc_data_nb['condition'].str.contains('com

# Creating sentiment column to categorize positive or negative rev

bc_data_nb['sentiment'] = ['Positive' if x > 7.0 else 'Negative' f

# Creating a 'punc_emphasis' column that scores how many exclamati

bc_data_nb['punc_emphasis'] = bc_data_nb['review'].apply(lambda x:

# Creating a 'capt_emphasis' column that scores how many capitaliz

bc_data_nb['capt_emphasis'] = bc_data_nb['review'].apply(lambda x:

# Loading in the features for model analysis

X5 = bc_data_nb[['review', 'drugName', 'condition', 'punc_emphasis
y5 = bc_data_nb['sentiment']

# Train test split

X_train_5_nb, X_test_5_nb, y_train_5_nb, y_test_5_nb = train_test_

# Apply clean_text function for Train Set

X_train_5_nb['review'] = X_train_5_nb['review'].apply(clean_text)

# Tokenize the training data with a simple split of words

X_train_5_nb['review'] = X_train_5_nb['review'].apply(lambda x: x.
X_train_5_nb['review'] = X_train_5_nb['review'].map(' '.join)

# Instantiate CountVectorizer()

cv = CountVectorizer()
```

```
    cv = CountVectorizer()
    X_train_5_nb = cv.fit_transform(X_train_5_nb['review'])

    # Train - fitting the training data to a Naive Bayes Classifier

    nb = MultinomialNB(alpha=0.5)
    nb.fit(X_train_5_nb, y_train_5_nb)

    # Validation - Cross Validation on the Naive Bayes Classifier Mode

    scores = cross_val_score(nb, X_train_5_nb, y_train_5_nb, cv=5)
    print('Cross Validation Scores: ', scores)
    print('Mean Cross Validation Score: ', scores.mean())

    # Naive Bayes Classifier Test Preprocessing

    X_test_5_nb['review'] = X_test_5_nb['review'].apply(clean_text)
    X_test_5_nb['review'] = X_test_5_nb['review'].apply(lambda x: x.sp
    X_test_5_nb['review'] = X_test_5_nb['review'].map(' '.join)
    X_test_5_nb = cv.transform(X_test_5_nb['review'])

    # Naive Bayes Classifier Predictions

    nb_pred_final = nb.predict(X_test_5_nb)

    ### Naive Bayes Test Set Evaluation

    print('Naive Bayes Accuracy: ', accuracy_score(y_test_5_nb, nb_pre
    print('Naive Bayes F1 Score: ', f1_score(y_test_5_nb, nb_pred_fina
    print('Naive Bayes Precision Score: ', precision_score(y_test_5_nb
    print('Naive Bayes Recall Score: ', recall_score(y_test_5_nb, nb_p

    return nb
```

In [*]:
```
multinomial_bayes('data/drugsComTrain_raw.tsv')
```

In [*]:
```
model = multinomial_bayes('data/drugsComTrain_raw.tsv')
```

## Pickle the model we have created

In [*]:
```
import joblib

# use the built-in open() function to open a file
output_file = open("naivebayes_classifier.pkl", "wb")

# dump the variable's contents into the file
joblib.dump(model, output_file)

# close the file, ensuring nothing stays in the buffer
output_file.close()
```