Flatiron Phase 5 Project

Aaron Galbraith

https://www.linkedin.com/in/aarongalbraith (https://www.linkedin.com/in/aarongalbraith)
https://github.com/aarongalbraith (https://github.com/aarongalbraith)

Submitted: November 21, 2023

Contents

Business Understanding

In 2022 the US Supreme Court ruled in Dobbs v. Jackson Women's Health Organization (https://www.supremecourt.gov/opinions/21pdf/19-1392_6j37.pdf) that the United States Constitution would no longer confer a right to abortion, leaving the legality of abortion timelines and procedures to the discretion of the individual states and territories. This brought an end to constitutionally enshrined abortion access after the 1973 ruling in Roe v. Wade (https://tile.loc.gov/storage-services/service/ll/usrep/usrep410/usrep410113/usrep410113.pdf) had codified that right into law for roughly two generations. Both immediately and in the short time since the Dobbs decision, many states have reduced access to abortion, and 14 states have banned abortion entirely (https://reproductiverights.org/maps/abortion-laws-by-state/). Advocates for birth control access fear that new government controls could move beyond abortion and attempt to restrict birth control access as well (https://www.npr.org/2022/08/16/1117615628/abortion-birth-control-title-x-supreme-court-family-planning-planned-parenthood).

In the new reproductive environment created by this ruling, Americans who are concerned with family planning are showing greater interest in birth control options and are more likely to consume and practice the birth control methods (that remain legal) in greater numbers than before. Pfizer can capitalize on this trend by understanding public perceptions of the various methods and responding to these perceptions in their marketing.

In 2018, researchers Surya Kallumadi and Felix Gräßer at UC Irvine created the UCI ML Drug Review Dataset (https://www.kaggle.com/datasets/jessicali9530/kuc-hackathon-winter-2018/) after collecting reviews from Drugs.com (https://www.drugs.com/) that users had written about various drugs between 2008 and 2017. A substantial portion of these reviews addressed birth control and emergency contraception drugs.

People often share similar sentiments with each other in online spaces such as Reddit (https://www.reddit.com/r/birthcontrol/) and Quora (https://www.quora.com/search?q=birth%20control). Our project analyzes the Drug Review Dataset in order to 1) learn what the Dataset can tell Pfizer about sentiments toward the various methods of birth control and 2) train a model that can be applied in other online spaces to determine what birth control methods users are discussing and how they feel about them. With this tool, Pfizer can more effectively market their products to the increased demand created by the Dobbs ruling.

Data Understanding

Import files

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from datetime import datetime

import nltk
from nltk.stem import WordNetLemmatizer
from nltk import FreqDist
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import cmudict

nltk.download('wordnet')
nltk.download('punkt')

from nltk.corpus import stopwords

import string
from wordcloud import WordCloud

import html

! pip install contractions
import contractions

import re

from IPython.display import display

import seaborn as sns

from xgboost import XGBClassifier

from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, recall_score, confusion_matrix, roc_curve, auc
from sklearn.metrics import precision_score, f1_score, roc_auc_score, log_loss
from sklearn.dummy import DummyClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.preprocessing import LabelEncoder


from pathlib import Path

SEED = 1979
```

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

```
Requirement already satisfied: contractions in /Users/stubbletrouble/opt/anaconda3/envs/lear
n-env/lib/python3.8/site-packages (0.1.73)
Requirement already satisfied: textsearch>=0.0.21 in /Users/stubbletrouble/opt/anaconda3/env
s/learn-env/lib/python3.8/site-packages (from contractions) (0.0.24)
Requirement already satisfied: anyascii in /Users/stubbletrouble/opt/anaconda3/envs/learn-en
v/lib/python3.8/site-packages (from textsearch>=0.0.21->contractions) (0.3.2)
Requirement already satisfied: pyahocorasick in /Users/stubbletrouble/opt/anaconda3/envs/lea
rn-env/lib/python3.8/site-packages (from textsearch>=0.0.21->contractions) (2.0.0)
```

Load and briefly explore data set

In [2]:
```
url1 = 'https://raw.githubusercontent.com/aarongalbraith/flatiron-phase5-project/main/data/dr
url2 = 'https://raw.githubusercontent.com/aarongalbraith/flatiron-phase5-project/main/data/dr
```

In [3]:
```
d1 = pd.read_csv(url1, delimiter='\t', encoding='latin-1')
d2 = pd.read_csv(url2, delimiter='\t', encoding='latin-1')
df = pd.concat([d1,d2]).reset_index().drop(columns=['Unnamed: 0', 'index'])
```

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

Out[4]:

	drugName	condition	review	rating	date	usefulCount
0	Valsartan	Left Ventricular Dysfunction	"It has no side effect, I take it in combinati...	9.0	May 20, 2012	27
1	Guanfacine	ADHD	"My son is halfway through his fourth week of ...	8.0	April 27, 2010	192
2	Lybrel	Birth Control	"I used to take another oral contraceptive, wh...	5.0	December 14, 2009	17
3	Ortho Evra	Birth Control	"This is my first time using any form of birth...	8.0	November 3, 2015	10
4	Buprenorphine / naloxone	Opiate Dependence	"Suboxone has completely turned my life around...	9.0	November 27, 2016	37

In [5]:
```
df.shape
```

Out[5]:
```
(215063, 6)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 215063 entries, 0 to 215062
Data columns (total 6 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   drugName     215063 non-null  object 
 1   condition    213869 non-null  object 
 2   review       215063 non-null  object 
 3   rating       215063 non-null  float64
 4   date         215063 non-null  object 
 5   usefulCount  215063 non-null  int64  
dtypes: float64(1), int64(1), object(4)
memory usage: 9.8+ MB
```

There are some missing `condition` labels.

drugName feature

In [7]:
```
df.drugName.value_counts()
```

Out[7]:
```
drugName
Levonorgestrel                       4930
Etonogestrel                         4421
Ethinyl estradiol / norethindrone    3753
Nexplanon                            2892
Ethinyl estradiol / norgestimate     2790
                                     ...
Melpaque HP                             1
Cortisone                               1
Reyataz                                 1
Striant                                 1
Allergy DN PE                           1
Name: count, Length: 3671, dtype: int64
```

In [8]:
```
df.drugName.value_counts().quantile(.90)
```

Out[8]:
```
120.0
```

There are 3,671 unique drug names, and 10% of the drug names have more than 120 reviews.

In [9]:
```
pd.set_option("display.max_rows", None)
print(df.drugName.value_counts())
pd.set_option("display.max_rows", 60)
```

```
Voltaren
113
Acetaminophen / aspirin / caffeine
113
Atorvastatin
112
Prepopik
111
Citric acid / magnesium oxide / sodium picosulfate
111
Dexilant
111
Aubra
110
MiraLax
110
Intuniv
109
Mobic
```

A casual overview of the entire list of drug names indicates that they all seem valid. Some seem to specify drug combinations and/or dosage amounts.

condition feature

In [10]:
```
df.condition.value_counts()
```

Out[10]:
```
condition
Birth Control                                      38436
Depression                                         12164
Pain                                                8245
Anxiety                                             7812
Acne                                                7435
                                                   ...
Systemic Candidiasis                                   1
Wilson's Disease                                       1
unctional Gastric Disorde                              1
Sepsis                                                 1
105</span> users found this comment helpful.           1
Name: count, Length: 916, dtype: int64
```

In [11]:
```
df.condition.value_counts().quantile(.90)
```

Out[11]:
```
332.5
```

There are 916 unique conditions, and 10% of the conditions have more than 332 reviews.

In [12]:
```
pd.set_option("display.max_rows", None)
print(df.condition.value_counts())
pd.set_option("display.max_rows", 60)
```

```
Ventricular Tachycardia                                                34
Nocturnal Leg Cramps                                                   33
Herbal Supplementation                                                 33
Cyclic Vomiting Syndrome                                               33
Dysautonomia                                                           33
Tinea Versicol                                                         33
Hiccups                                                                33
Tinea Pedis                                                            33
Inflammatory Bowel Disease                                             33
Postoperative Ocular Inflammation                                      32
Night Terrors                                                          32
Precocious Puberty                                                     31
Tinea Corporis                                                         31
Prevention of Osteoporosis                                             31
Insomnia, Stimulant-Associated                                         30
Ischemic Stroke, Prophylaxis                                           30
Hypoestrogenism                                                        30
Vitamin/Mineral Supplementation and Deficiency                         30
Insulin Resistance Syndrome                                            29
Soft Tissue Sarcoma                                                    29
Labor Pain                                                             29
```

Oddly, the condition labels often (always?) omit initial 'F' and terminal 'r'. We can isolate instances of the former by searching for conditions that start with a lower case letter.

We will eventually trim our records to just birth control and emergency contraception (and perhaps birth control exclusively), but we will need all the records to help us determine missing condition labels. After we have restored (or discarded) all missing condition labels, we can drop the conditions outside the scope of this review.

drugName × condition features

In [13]:
```
df.groupby('drugName').condition.nunique().value_counts()[:10]
```

Out[13]:
```
condition
1     1869
2      782
3      334
4      195
5      122
6       83
7       52
8       51
9       38
11      20
Name: count, dtype: int64
```

This means that, for example, 1869 drugs treat just 1 condition, etc.

In [14]:
```
df.groupby('condition').drugName.nunique().value_counts()[:10]
```

Out[14]:
```
drugName
2     188
1     166
4      78
3      72
5      44
7      41
9      34
6      34
10     17
11     16
Name: count, dtype: int64
```

This means that 188 conditions are treatable by two drugs, etc.

There is a one-to-many relationship in both directions between drug names and conditions.

review feature

In [15]:
```
df.review.nunique()
```

Out[15]:
```
128478
```

This suggests that just over half of the review values are unique. Almost certainly there will be some duplication issues to deal with.

Let's look at several reviews.

In [16]:
```
for i in range(5):
    print(df.review[i], '\n-----')
```

```
"It has no side effect, I take it in combination of Bystolic 5 Mg and Fish Oil"
-----
"My son is halfway through his fourth week of Intuniv. We became concerned when he began thi
s last week, when he started taking the highest dose he will be on. For two days, he could h
ardly get out of bed, was very cranky, and slept for nearly 8 hours on a drive home from sch
ool vacation (very unusual for him.) I called his doctor on Monday morning and she said to s
tick it out a few days. See how he did at school, and with getting up in the morning. The la
st two days have been problem free. He is MUCH more agreeable than ever. He is less emotiona
l (a good thing), less cranky. He is remembering all the things he should. Overall his behav
ior is better.
We have tried many different medications and so far this is the most effective."
-----
"I used to take another oral contraceptive, which had 21 pill cycle, and was very happy- ver
y light periods, max 5 days, no other side effects. But it contained hormone gestodene, whic
h is not available in US, so I switched to Lybrel, because the ingredients are similar. When
my other pills ended, I started Lybrel immediately, on my first day of period, as the instru
ctions said. And the period lasted for two weeks. When taking the second pack- same two week
s. And now, with third pack things got even worse- my third period lasted for two weeks and
now it&#039;s the end of the third week- I still have daily brown discharge.
The positive side is that I didn&#039;t have any other side effects. The idea of being perio
d free was so tempting... Alas."
-----
"This is my first time using any form of birth control. I&#039;m glad I went with the patch,
I have been on it for 8 months. At first It decreased my libido but that subsided. The only
downside is that it made my periods longer (5-6 days to be exact) I used to only have period
s for 3-4 days max also made my cramps intense for the first two days of my period, I never
had cramps before using birth control. Other than that in happy with the patch"
-----
"Suboxone has completely turned my life around.  I feel healthier, I&#039;m excelling at my
job and I always have money in my pocket and my savings account.  I had none of those before
Suboxone and spent years abusing oxycontin.  My paycheck was already spent by the time I got
it and I started resorting to scheming and stealing to fund my addiction.  All that is histo
ry.  If you&#039;re ready to stop, there&#039;s a good chance that suboxone will put you on
the path of great life again.  I have found the side-effects to be minimal compared to oxyco
ntin.  I&#039;m actually sleeping better.  Slight constipation is about it for me.  It trul
y is amazing. The cost pales in comparison to what I spent on oxycontin."
-----
```

There appear to be escaped characters (e.g. `&#039;` , indicating an apostrophe) and contractions. We can address this now without affecting our analysis.

We'll reset the review texts to unescape these characters and expand all contractions.

Note: This will replace all instances of `ain't` with `are not` , resulting in some subject-verb agreement issues (e.g. `I are not`). This difference will be negligible in our analysis.

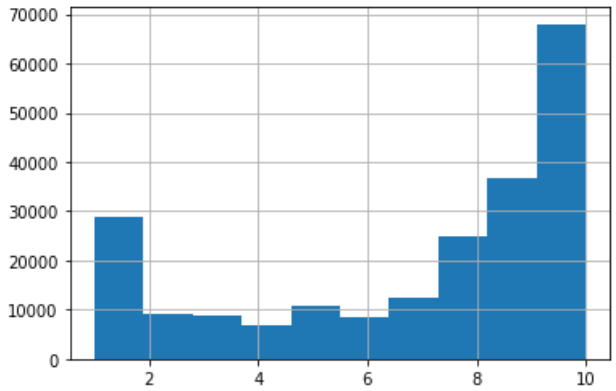
In [17]:
```
df.review = df.review.apply(lambda x: contractions.fix(html.unescape(x)))
```

`rating` feature

In [18]:
```
df.rating.value_counts()
```

Out[18]:
```
rating
10.0    68005
9.0     36708
1.0     28918
8.0     25046
7.0     12547
5.0     10723
2.0      9265
3.0      8718
6.0      8462
4.0      6671
Name: count, dtype: int64
```

In [19]:
```
df.rating.hist(bins=df.rating.nunique());
```



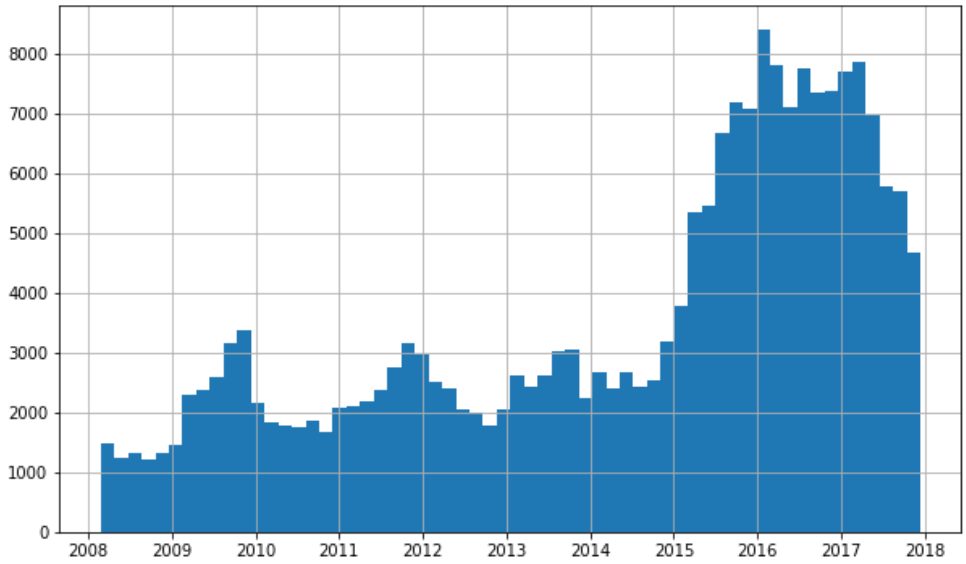
Most of the conditions lie at the extremes, and more of them appear to be at the positive extreme.

`date` feature

In order to get a better understanding of the `date` feature, we'll convert it to a `datetime` object.

In [20]:
```
df['datetime'] = df.date.apply(lambda x: datetime.strptime(x, '%B %d, %Y'))
```

In [21]:
```
df.datetime.hist(bins=60, figsize=(10,6));
```



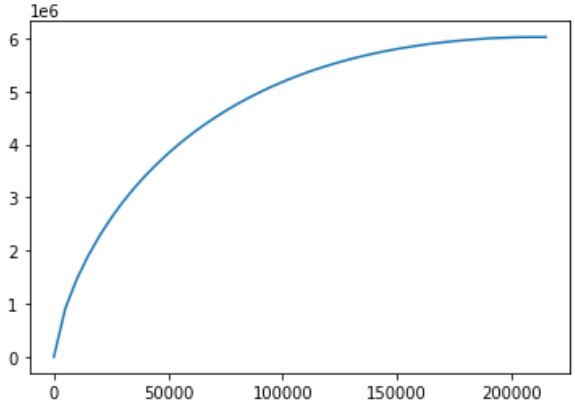
In [22]:
```
start_date = df.datetime.min().strftime('%B %d, %Y')
end_date = df.datetime.max().strftime('%B %d, %Y')

print('The reviews span specifically from', start_date+',', 'to', end_date+'.')
```

```
The reviews span specifically from February 24, 2008, to December 12, 2017.
```

The reviews began to surge in early 2015.

`usefulCount` feature

In [23]:
```
usefulCountCumulative = list(df.usefulCount.values)
usefulCountCumulative.sort(reverse=True)

X = range(0,len(df),5000)
Y = []
for x in X:
    Y.append(sum(usefulCountCumulative[:x]))
fig, ax = plt.subplots()
ax.plot(X,Y);
```



(If all reviews were equally useful, this graph would be a straight diagonal line from SW to NE)

In [24]:
```
df.usefulCount.value_counts()[:10]
```

Out[24]:
```
usefulCount
2    8543
3    8525
0    8402
4    8301
1    8237
5    7844
6    7213
7    6736
8    6271
9    5940
Name: count, dtype: int64
```

Data Preparation

An outline for getting the information we need

In addition to the standard goals of data cleaning, the main goals of our data preparation are

1. Drop any records that don't pertain to birth control
2. Identify which method of birth control each review discusses

The records indicate a drug name and what condition (e.g. birth control) it treats, but problems with both of these features abound. There are many missing condition labels, and we can only restore them (and find out which ones should also have the "birth control" label) by understanding how they associate with the drug name labels. Some drug labels are for brand names, and others are for the generic or chemical name of the drug. Many records are actually instances of the review being entered twice, once with the brand name and once with the generic name. That's not the end of the complications, but it will suffice for an overview.

Once the first main goal is achieved and we are able to trim the records to only those that concern the condition of birth control, we can use the edited and matched brand/generic names to identify the birth control method each drug is an

Missing and erroneous condition labels

In this section we will identify all `condition` labels that are either missing or in need of editing.

In [25]:
```
len(df[df.condition.isna()])
```

Out[25]: 1194

In [26]:
```
df.condition.fillna('missing', inplace=True)
```

In [27]:
```
len(df[df.condition == 'missing'])
```

Out[27]: 1194

We noticed another `condition` label that was meant to indicate missing and should be accordingly changed.

In [28]:
```
df.condition = df.condition.apply(lambda x: 'missing' if 'Not Listed' in x else x)
```

In [29]:
```
len(df[df.condition == 'missing'])
```

Out[29]: 1786

We've identified some actual missing `condition` labels, but we noticed there are more `condition` labels that seem suspicious, particularly ones that start with something other than an upper case character. Let's look at all such `condition` labels.

In [30]:
```
set(df[(~df.condition.str[0].isin(list(string.ascii_uppercase))) &
   (df.condition != 'missing')
  ].condition)
```

Out[30]:
```
{'0</span> users found this comment helpful.',
 '100</span> users found this comment helpful.',
 '105</span> users found this comment helpful.',
 '10</span> users found this comment helpful.',
 '110</span> users found this comment helpful.',
 '11</span> users found this comment helpful.',
 '121</span> users found this comment helpful.',
 '123</span> users found this comment helpful.',
 '12</span> users found this comment helpful.',
 '135</span> users found this comment helpful.',
 '13</span> users found this comment helpful.',
 '142</span> users found this comment helpful.',
 '145</span> users found this comment helpful.',
 '146</span> users found this comment helpful.',
 '14</span> users found this comment helpful.',
 '15</span> users found this comment helpful.',
 '16</span> users found this comment helpful.',
 '17</span> users found this comment helpful.',
 '18</span> users found this comment helpful.',
```

These fall into three categories:

1. "X users found this comment helpful" should be regarded as an erroneous label and retagged as "missing.

In [31]:
```
df.condition = df.condition.apply(lambda x: 'missing' if 'users found' in x else x)
```

In [32]:
```
len(df[df.condition == 'missing'])
```

Out[32]: 2957

2. Labels that show a clipped copy of the `drugName` label and end with a parenthesis should also be regarded as missing. These erroneous labels merely repeat information already available in the `drugName` feature.

In [33]:
```
df.condition = df.condition.apply(lambda x: 'missing' \
                                  if x[0] not in list(string.ascii_uppercase) and \
                                  x[-1] in ['(', ')'] \
                                  else x)
```

In [34]:
```
len(df[df.condition == 'missing'])
```

Out[34]: 3286

3. Other `condition` labels appear to omit the first and/or last several characters. We can infer certain corrections here to restore many of the conditions.

In [35]:
```
def condition_restore(condition):
    if condition.split()[-1] in ['Disorde', 'eve', 'Shoulde', 'Cance']:
        condition = condition+'r'
    if condition.split()[0] in ['acial', 'ibrocystic', 'ungal', 'amilial', 'ailure', 'ever', '
                                'emale', 'unctional', 'actor', 'ibromyalgia', 'atigue']:
        condition = 'F'+condition
    if condition.split()[0] in ['llicular', 'llicle', 'lic', 'cal']:
        condition = 'Fo'+condition
    if condition.split()[0] in ['mance']:
        condition = 'Perfor'+condition
    if condition.split()[0] in ['zen']:
        condition = 'Fro'+condition
    if condition.split()[0] in ['mis']:
        condition = 'Dermatitis Herpetifor'+condition
    return condition

df.condition = df.condition.apply(lambda x: condition_restore(x))
```

Let's look at what we have left.

In [36]:
```
set(df[(~df.condition.str[0].isin(list(string.ascii_uppercase))) &
   (df.condition != 'missing')
  ].condition)
```

Out[36]: {'m Pain Disorder', 'me', 't Care', "von Willebrand's Disease"}

"von Willebrand's Disease" appears to be a naturally uncapitalized condition. The others have been impossible to restore and will also be regarded as missing.

In [37]:
```
df.condition = df.condition.apply(lambda x: 'missing' \
                                  if x[0] not in list(string.ascii_uppercase) and \
                                  x.split()[0] != 'von' \
                                  else x)
```

In [38]:
```
len(df[df.condition == 'missing'])
```

Out[38]: 3293

We will be able to restore more of these missing condition labels after we do some work with duplicates.

Duplicates

In [39]:
```
df.duplicated().value_counts()
```

Out[39]:
```
False    215061
True          2
Name: count, dtype: int64
```

In [40]:
```
df[df.duplicated()]
```

Out[40]:

	drugName	condition	review	rating	date	usefulCount	datetime
178703	Levonorgestrel	Emergency Contraception	"I had a quickie n he decided to finish it off...	1.0	September 23, 2016	10	2016-09-23
191001	Plan B	Emergency Contraception	"I had a quickie n he decided to finish it off...	1.0	September 23, 2016	10	2016-09-23

At this point we'll create a function that will allow us to examine every instance of a certain review. For now, it will help us sort out duplication issues. Later, it will be a convenient way to display the content of certain reviews.

In [41]:
```
show_duplicates = True #useful until duplicates are sorted, then turned off
show_method = False #will become useful when birth control methods are assigned to all records

def show_review(index):
    if show_duplicates:
        display(df[df.review == df.loc[index].review][['drugName', 'condition', 'rating', 'da
    if show_method:
        print('\nReview #'+str(index),'| Method:',df.loc[index].method,'| Rating:',
              df.loc[index].rating,'| Upvotes:',df.loc[index].usefulCount, '\n\n'+df.review.l
    else:
        print('\nReview #'+str(index),'| Rating:',
              df.loc[index].rating,'| Upvotes:',df.loc[index].usefulCount, '\n\n'+df.review.l
```

In [42]:
```
show_review(178703)
```

	drugName	condition	rating	date	usefulCount
131531	Levonorgestrel	Emergency Contraception	1.0	September 23, 2016	10
143768	Plan B	Emergency Contraception	1.0	September 23, 2016	10
178703	Levonorgestrel	Emergency Contraception	1.0	September 23, 2016	10
191001	Plan B	Emergency Contraception	1.0	September 23, 2016	10

```
Review #178703 | Rating: 1.0 | Upvotes: 10

I had a quickie n he decided to finish it off in me... Well IMMEDIATELY we went 2 our local
pharmacy n bought this plan b 1 step pill.I took it immediately.2 weeks later,took a pregnan
cy test n got the world's BIGGEST POSITIVE. The small pill was $50.That was the 1st time in
a year n a half that I had intercourse n the last after I had my first son. I honestly belie
ve this pill is ineffective because they just want you to think it works when n reality, it
would never work. Alot of women do not know their bodies when they ovulate so if your not fe
rtile and he ejaculates n you and you take the pill n do not get preg., The pill is supposed
to make you think it worked. DO NOT buy. Was NEVER effective. Thank you!
```

For reasons we will explore later, we believe this review was submitted twice by the same person, that each instance of it happened to receive 10 upvotes, and that it should correctly be associated with a grand total of 20 upvotes. Because this is one special instance where the review happened to receive 10 upvotes both times, making it a true duplicate of the data set, we will fix the values here, lest it interfere with operations later on.

In [43]:
```
df.at[178703, 'usefulCount'] = 20
df.at[191001, 'usefulCount'] = 20
df.drop([131531, 143768], inplace=True)
```

Duplicates due to brand / generic pairs

The main type of duplicate we should look out for is records with duplicate reviews, as those likely indicate some kind of actual erroneous duplication. Let's see how many of those there are.

In [44]:
```
df.duplicated(subset=['review']).value_counts()
```

Out[44]:
```
False    128467
True      86594
Name: count, dtype: int64
```

That's a lot!

Let's explore some facets of these duplicates.

In [45]:
```
len(df[df.duplicated(subset=df.columns.difference(['drugName']))])
```

Out[45]:
```
85876
```

The vast majority of duplicate reviews are accounted for by different drug names. Let's explore some examples.

In [46]:
```
for ind in df[df.duplicated(subset=df.columns.difference(['drugName']))].index[:5]:
    show_review(ind)
```

	drugName	condition	rating	date	usefulCount
374	Etonogestrel	Birth Control	9.0	April 21, 2017	5
524	Nexplanon	Birth Control	9.0	April 21, 2017	5

```
Review #524 | Rating: 9.0 | Upvotes: 5

First had implanon then got Nexplanon, had a period first month and I have not had one sinc
e. I am due to remove it next year.  I do notice spotting  sometimes for a day but it honest
ly  usually coincides with when I am stressed.
Had some weight gain also.

So far the best BECAUSE I have  had in all my years.  I plan on trying for a baby next year
then I will be back on it.
```

	drugName	condition	rating	date	usefulCount
321	Duloxetine	Anxiety	1.0	September 5, 2010	27
574	Cymbalta	Anxiety	1.0	September 5, 2010	27

```
Review #574 | Rating: 1.0 | Upvotes: 27

Prescribed via a Psychiatrist for severe Panic attacks for 2 years.
If I take dosage late or forget to take it the withdrawal symptoms kick in.
Gnawing physical pain, breathlessness, disorientation to time, difficulties in word finding
while speaking, severe muscle pain and stiffness, nausea, labile emotions and panic.
```

	drugName	condition	rating	date	usefulCount
378	Ethinyl estradiol / levonorgestrel	Birth Control	2.0	October 8, 2015	7
726	Orsythia	Birth Control	2.0	October 8, 2015	7

```
Review #726 | Rating: 2.0 | Upvotes: 7

I have only been on orsythia for about 1 month and I just started my second week of my secon
d month.  I guess I did not notice earlier but I started to get slight headaches and I did n
ot feel very well physically and mentally.  It do help with my cramps and my period, not so
much my acne but it is better.  The worst part of orsythia has to be the mood swings and the
sweating!  I sweat a lot even if it is cold I will start a light sweat, it is gross.  But th
e mood swings are the worst I just started feeling this and it happened while I was talking
to one of my friends I just blew up on him, for no reason.  Then later on I started to feel
really bad(mentally) and I cried for a while and I could not figure out why I was crying! I
do not recommend!
```

	drugName	condition	rating	date	usefulCount
855	Pristiq	Depression	8.0	November 1, 2013	81
1070	Desvenlafaxine	Depression	8.0	November 1, 2013	81

```
Review #1070 | Rating: 8.0 | Upvotes: 81

I have suffered from severe anxiety (GAD) and was taking more and more Klonopin as time went
on.  I am very sensitive to medication and have tried many different SSRI/SNRI's through the
year with horrible side effects.  Finally, I had DNA testing to see what I would respond to
and the result was Pristiq.  I started it several months ago in a small dose (I split the pi
lls even though they say do not do this) and within a few days my anxiety literally went awa
y.  I was able to cut my Klonopin in 1/2 over a two month period.  The first week or two I w
as extremely tired but that passed.  The only side effect I get from time to time is migrain
es.  It still amazes me that my anxiety has disappeared.  I no longer keep Klonopin in my  p
ocket!
```

	drugName	condition	rating	date	usefulCount
609	Lo Loestrin Fe	Birth Control	8.0	February 1, 2012	7
1375	Ethinyl estradiol / norethindrone	Birth Control	8.0	February 1, 2012	7

```
Review #1375 | Rating: 8.0 | Upvotes: 7

I have been taking my first pack of Lo Loestrin Fe and I must say it really works for me. I
was a little nervous at first because this is my first time taking birth control and I have
heard all the negative side effects of taking birth control. I have had spotting [brown-ish
color] for three weeks after my period, but that is normal for the first month. I have breas
t tenderness and mood swings every now and then, then again it is expected for the first few
months. I have not yet experienced any weight gain. So far I am satisfied, but I wish it was
not so expensive.
```

These five examples make clear that the vast majority of duplicates are due to double-entry; (nearly) every review is entered once with its generic name and once with its brand name.

We can use this phenomenon to restore some of the missing condition labels. If a missing condition label is part of such a unique pair, then we can confidently assign it the condition of its pair-mate.

Let's broaden our search to records that duplicate every feature other than drug name and condition.

In [47]:
```
len(df[df.duplicated(subset=df.columns.difference(['drugName', 'condition']))])
```

Out[47]: 86221

This is how many records are duplicates of other records in all values EXCEPT (POSSIBLY) drug name and condition. If a record is duplicated in this manner, the second (and third, fourth, etc.) instance will be captured in this bucket of dupes.

If we check only this bucket for dupes, we can see whether there are any triplets, etc.

In [48]:
```
df_dupes = df[df.duplicated(subset=df.columns.difference(['drugName', 'condition']))]
```

In [49]:
```
len(df_dupes[df_dupes.duplicated(subset=df_dupes.columns.difference(['drugName', 'condition']
```

Out[49]: 1

There is only one.

In [50]:
```
df_dupes[df_dupes.duplicated(subset=df_dupes.columns.difference(['drugName', 'condition']))]
```

Out[50]:

	drugName	condition	review	rating	date	usefulCount	datetime
140144	Octreotide	Diabetes, Type 1	"Great medicine. No side effects."	9.0	October 31, 2011	2	2011-10-31

In [51]:
```
show_review(140144)
```

	drugName	condition	rating	date	usefulCount
39512	Reprexain	Pain	7.0	October 5, 2012	10
60998	Insulin regular	Diabetes, Type 1	9.0	October 31, 2011	2
119972	Insulin glulisine	Diabetes, Type 1	9.0	October 31, 2011	3
133212	Sandostatin	Diabetes, Type 1	9.0	October 31, 2011	2
140144	Octreotide	Diabetes, Type 1	9.0	October 31, 2011	2
141100	Insulin isophane / insulin regular	Diabetes, Type 1	9.0	October 31, 2011	10
148049	ReliOn / Novolin 70 / 30	Diabetes, Type 1	9.0	October 31, 2011	10
184262	Hydrocodone / ibuprofen	Pain	7.0	October 5, 2012	10

```
Review #140144 | Rating: 9.0 | Upvotes: 2

Great medicine. No side effects.
```

There are 6 records with the same review, date, rating, and condition. (The reviews on October 5, 2012, appear to be just a coincidence of the same review wording for a different drug and condition.) Because they're on the *same day*, it seems likely that these reviews were possibly entered repeatedly by the same person. The two with a useful count of 10 are likely a brand/generic pair.

As for the other 4, it's not clear what is going on. We will (would) later discover that there is also some discrepancy as to which of these is a brand or generic name. Since the review text isn't very descriptive, and the useful count is so low, (and it doesn't pertain to the main conditions relevant to our project), let's just drop all 4.

In [52]:
```
df.drop([60998, 119972, 133212, 140144], inplace=True)
```

Now we should be able to create a list of pairs of indices of records that match in all features except possibly drug name and condition. To make this run faster, we'll first create a way to sort them by date.

In [53]:
```
%%time
# ⏰ record the time for this cell -- usually 10-15 seconds

# create stripped down dataframe that does not have drug names or conditions
# we don't need these features for this operation because we're checking for matches on all o
df_pairs = df.drop(columns=['drugName', 'condition']).copy()

# create a list of indices of records that duplicate everything other than drug name and cond
df_dupes = df_pairs[df_pairs.duplicated()].index.tolist().copy()

# create and populate a dictionary whose keys are dates and whose values are indices
dates_dict = {}
# populate dictionary with keys that are dates belonging to the duplicates
for date_ in list(set(df[df.index.isin(df_dupes)].date.tolist())):
    dates_dict[date_] = []
# populate dictionary with values that are indices that are NOT from the duplicate list but D
for i in df[~df.index.isin(df_dupes)].index:
    dates_dict[df.loc[i].date].append(i)
```

```
CPU times: user 10.5 s, sys: 46.4 ms, total: 10.6 s
Wall time: 10.6 s
```

Now we can use this dates dictionary to sort and identify the pairs.

In [54]:
```
%%time
# ⏰ record the time for this cell -- usually 2-4 mins

# create a list of record pairs where each entry is a list of two indices
pairs = []

# iterate over the indices from the dupes list
for i in df_dupes:
    # set the date to the date from index i
    date_i = df.loc[i].date
    # iterate over OTHER indices who share that date
    for j in dates_dict[date_i]:
        # check for a match
        if df_pairs.loc[i].equals(df_pairs.loc[j]) and df.drugName.loc[i] != df.drugName.loc[
            # remove this index from the dates dictionary so we have fewer to search through
            dates_dict[date_i].remove(j)
            # add this pair to the pairs list
            pairs.append([i,j])
            break
```

```
CPU times: user 2min 52s, sys: 358 ms, total: 2min 53s
Wall time: 2min 53s
```

Let's take a look at several of the pairs we've collected.

In [55]:
```
pairs[:10]
```

Out[55]:
```
[[524, 374],
 [574, 321],
 [726, 378],
 [1070, 855],
 [1375, 609],
 [1397, 1281],
 [1735, 1043],
 [1965, 299],
 [2014, 1844],
 [2091, 1609]]
```

Here we'll create a dictionary that matches the index of one pair member to the other member of the pair.

In [56]:
```
pairs_dict = {}

for pair in pairs:
    for i in range(2):
        pairs_dict[pair[i]] = pair[1-i]
```

Restore missing labels

We will restore missing `condition` labels in two ways, in order of certainty:

1. For missing values that possess a pair match, we will assign it the condition of its match.
2. For the remaining missing values, we will assign it the condition that is most commonly associated with its drug name.

In [57]:
```
len(df[df.condition == 'missing'])
```

Out[57]:
```
3293
```

In [58]:
```
%%time
# ⏰ record the time for this cell -- usually 10-15 seconds

# iterate over each record pair
for pair in pairs:
    # iterate over each member of the pair
    for i in range(2):
        # identify a pair member whose condition is missing
        if df.loc[pair[i]].condition == 'missing':
            # assign to the pair member the condition of its pair-mate
            df.at[pair[i], 'condition'] = df.loc[pairs_dict[pair[i]]].condition
```

```
CPU times: user 13.6 s, sys: 24.2 ms, total: 13.7 s
Wall time: 13.7 s
```

In [59]:
```
len(df[df.condition == 'missing'])
```

Out[59]:
```
2968
```

Because it will be useful later, we'll make a feature that names the indicated drug and, if applicable, the paired drug.

This is not a *final* replacement for the drug name feature, but it will allow us to better recognize the relationship between the generic and brand drug names.

In [60]:
```
%%time
# ⏰ record the time for this cell -- usually 15-30 seconds

df['ind'] = df.index

def drugList_fix(index, drugName_):
    drugList = [drugName_]
    if index in pairs_dict:
        drugList.append(df.loc[pairs_dict[index]].drugName)
        # alphabetize each drug pair so that we will not mistakenly duplicate e.g. [A,B] & [B
        drugList.sort()
    return drugList

df['drugList'] = df.apply(lambda x: drugList_fix(x.ind, x.drugName), axis=1)

df.drop(columns='ind', inplace=True)
```

```
CPU times: user 18.1 s, sys: 108 ms, total: 18.3 s
Wall time: 18.3 s
```

Now we can create a feature that tells us if a record is associated with a paired drug name or not.

In [61]:
```
df['isPaired'] = df.drugList.apply(lambda x: True if len(x) > 1 else False)
```

In [62]:
```
len(df[df.isPaired])
```

Out[62]:
```
172438
```

In [63]:
```
len(df[~df.isPaired])
```

Out[63]:
```
42619
```

Because lists confuse certain operations, we'll make a similar feature that records this information as a string.

In [64]:
```
df['drugSetString'] = df.drugList.apply(lambda x: x[0] + ' ' + x[1] if len(x) == 2 else x[0])
```

In [65]:
```
len(df[df.duplicated(subset=df.columns.difference(['drugName', 'drugSet', 'drugList']))])
```

Out[65]:
```
86199
```

With this new feature in place, we can drop one record from each of the brand/generic pairs. The drug name feature will retain only one member of the pair -- unpredictably either the brand or the generic -- which will make this feature more or less useless for the moment.

Before we drop these records, we'll create a bookmark copy of the dataframe.

In [66]:
```
# DO NOT re-run this cell out of sequence
# to use the dataframe as it was at this stage, un-comment, run, and re-comment the cell that
df_bookmark_1 = df.copy()
```

In [67]:
```
# df = df_bookmark_1.copy()
```

In [68]:
```
df.drop_duplicates(subset=df.columns.difference(['drugName', 'drugSet', 'drugList']), inplace
```

Now that all the duplicates are sorted out, we no longer need to show those details in the `show_review` function, so we'll toggle this value off.

In [69]:
```
show_duplicates = False
```

For every remaining record with a missing condition, we will assign it the condition that is most common for the drug indicated by that record. (This will not be biased by duplicates from brand/generic pairs, because we have dropped those duplicates.)

This will be the last use we have for conditions *not* relevant to our project.

In [70]:
```
drugs_w_missing_condition = list(set(df[df.condition == 'missing'].drugSetString))
```

In [71]:
```
len(drugs_w_missing_condition)
```

Out[71]:
```
678
```

In [72]:
```
df.drugSetString.nunique()
```

Out[72]:
```
3320
```

This applies to some 20% of the drugs. We'll create a dictionary that reports the most common condition for these drugs.

In [73]:
```
%%time
# record the time for this cell -- usually 10-20 seconds

most_common_condition = {}

for drug in drugs_w_missing_condition:
    condition = df[df.drugSetString == drug].condition.value_counts().idxmax()
    if condition == 'missing' and len(set(df[df.drugSetString == drug].condition)) > 1:
        condition = df[(df.drugSetString == drug) &
                       (df.condition != 'missing')
                      ].condition.value_counts().idxmax()
    proportion = round(df[df.drugSetString == drug].condition.value_counts(normalize=True)[0]
    most_common_condition[drug] = [condition, proportion]
```

```
CPU times: user 11.4 s, sys: 29.8 ms, total: 11.4 s
Wall time: 11.5 s
```

In [74]:
```
most_common_condition['Sildenafil Viagra']
```

Out[74]:
```
['Erectile Dysfunction', 0.87]
```

For example, if a review with an unlisted condition is about Viagra, we will assume the condition is Erectile Dysfunction.

In [75]:
```
len(df[df.condition == 'missing'])
```

Out[75]:
```
1799
```

In [76]:
```
df['condition'] = df.apply(lambda x: most_common_condition[x.drugSetString][0] \
                           if x.condition == 'missing' \
                           else x.condition, axis = 1)
```

In [77]:
```
len(df[df.condition == 'missing'])
```

Out[77]:
```
27
```

This is how many records there are that still have no condition label. This means the drugs indicated in these records are *only* indicated in references without an indicated condition. As such, there's not really anything we can do with these records, and we may as well drop them.

In [78]:
```
df.drop(df[df.condition == 'missing'].index, inplace=True)
```

In [79]:
```
len(df[df.condition == 'missing'])
```

Out[79]:
```
0
```

Drop records by condition

At this point, we still have more cleaning to do, but we have identified all the conditions that we can, and we won't have any further need for records with certain condition values, so we'll drop them.

In [80]:
```
# DO NOT re-run this cell out of sequence
# to use the dataframe as it was at this stage, un-comment, run, and re-comment the cell that
df_bookmark_2 = df.copy()
```

In [81]:
```
# df = df_bookmark_2.copy()
```

Let's take another look at the complete list of conditions and choose which ones to keep.

In [82]:
```
df.condition.nunique()
```

Out[82]:
```
817
```

Since there are so many conditions to consider, let's limit this to just conditions with at least 25 reviews.

In [83]:
```
pd.set_option("display.max_rows", None)
display(df['condition'].value_counts().loc[lambda x: x >= 25])
pd.set_option("display.max_rows", 60)
```

```
Myasthenia Gravis                                                  34
Crohn's Disease, Acute                                             33
Malaria Prevention                                                 33
Facial Wrinkles                                                    32
Tinnitus                                                           32
Opioid-Induced Constipation                                        32
Iron Deficiency Anemia                                             32
Postoperative Pain                                                 32
Ulcerative Colitis, Maintenance                                    31
Vulvodynia                                                         30
Gastroparesis                                                      30
Temporomandibular Joint Disorder                                   29
Cyclic Vomiting Syndrome                                           29
Condylomata Acuminata                                              28
Oral Thrush                                                        28
Insulin Resistance Syndrome                                        28
Vitamin/Mineral Supplementation and Deficiency                     28
Breakthrough Pain                                                  28
Insomnia, Stimulant-Associated                                     28
Mitral Valve Prolapse                                              27
```

In [84]:
```
df.drop(df[~df.condition.isin(['Birth Control', 'Emergency Contraception'])].index, inplace=T
```

In [85]:
```
df.condition.value_counts()
```

Out[85]:
```
condition
Birth Control              20054
Emergency Contraception     1735
Name: count, dtype: int64
```

Pairing generic and brand names

Now that we have a smaller number of records to deal with, we can sort out generic and brand names.

First we'll create a list of all values from the drug name feature. (Some of these have been dropped from the drug name feature itself when we dropped one record from each brand/generic pair, but all of them were included in the drug list feature.)

We'll create two lists: paired drugs (which we will attempt to sort into brand and generic) and single drugs (each of which we will then try to identify as either brand or generic).

In [86]:
```
len(df)
```

Out[86]:
```
21789
```

In [87]:
```
all_drug_lists = df.drugList.tolist()
all_drug_lists.sort()
```

In [88]:
```
len(all_drug_lists)
```

Out[88]:
```
21789
```

In [89]:
```
all_drug_names = set()

for list_ in all_drug_lists:
    all_drug_names.add(list_[0])
    if len(list_) > 1:
        all_drug_names.add(list_[1])

all_drug_names = list(all_drug_names)

all_drug_names.sort()
```

In [90]:
```
len(all_drug_names)
```

Out[90]: 191

In [91]:
```
# this will create a full list with duplicates
# we need to do this intermediate before moving to the following step to remove duplicates
paired_drug_lists = df[df.isPaired].drugList.tolist()
paired_drug_lists.sort()
```

In [92]:
```
len(paired_drug_lists)
```

Out[92]: 20989

In [93]:
```
paired_drug_names = set()

for pair in paired_drug_lists:
    paired_drug_names.add(pair[0])
    paired_drug_names.add(pair[1])

paired_drug_names = list(paired_drug_names)

paired_drug_names.sort()
```

In [94]:
```
unpaired_drug_names = [drug for drug in all_drug_names if drug not in paired_drug_names]

unpaired_drug_names.sort()
```

In [95]:
```
len(paired_drug_names)
```

Out[95]: 189

In [96]:
```
len(unpaired_drug_names)
```

Out[96]: 2

Together, these two lists of names constitute all the drug names left to sort into brand and generic categories.

In order to sort the list of paired drugs into brand and generic, we'll establish a dictionary whose keys are all the drug names that appear in a generic/brand pair.

In [97]:
```
drug_dict = {}

for drug in paired_drug_names:
    drug_dict[drug] = set()
```

We'll assign values to those keys according to the pairings. For example, if drug name A is in a generic/brand pair with drug name B, then they will appear on each other's list of values in this dictionary.

In [98]:
```
for pair in paired_drug_lists:
    drug_dict[pair[0]].add(pair[1])
    drug_dict[pair[1]].add(pair[0])
```

Let's find out how many of these drug names are associated with exactly one other drug name.

In [99]:
```
len({drug for drug in drug_dict if len(drug_dict[drug]) == 1})
```

Out[99]: 176

That should mean that exactly the remainder are associated with multiple drug names. It would make sense that drug names that belong to multiple generic/brand pairs are themselves the generic name. On that assumption, we'll create a list of generic drug names.

In [100]:
```
generics = [drug for drug in drug_dict if len(drug_dict[drug]) > 1]

generics.sort()
```

In [101]:
```
len(generics)
```

Out[101]: 13

Now we'll check to make sure that the drug names we've just designated as "generic" do NOT belong to a generic/brand pair with *another* "generic".

In [102]:
```
count = 0

for drug in generics:
    for match in drug_dict[drug]:
        if match in generics:
            count += 1

print(count)
```

```
0
```

Great.

Then we can begin designating drug names as "brands" if they are in a generic/brand pair with a generic.

In [103]:
```
brands = set()

for generic in generics:
    for match in drug_dict[generic]:
        brands.add(match)

brands = list(brands)

brands.sort()
```

In [104]:
```
len(brands)
```

Out[104]: 166

Now let's see what drugs remain and how many records they are associated with.

In [105]:
```
uncategorized = list(set(drug for drug in paired_drug_names if drug not in generics and drug 

uncategorized.sort()
```

To be clear, these are drug names with the following properties:

- the drug name belongs to an exclusive brand/generic pair
- we have not yet identified which members of the pair are brand and generic

In [106]:
```
len(uncategorized)
```

Out[106]: 10

We should be able to list all of these drug names in their pairs.

In [107]:
```
repeated = set()
for drug in uncategorized:
    if drug not in repeated:
        print(drug, '||', list(drug_dict[drug])[0])
        repeated.add(drug)
        repeated.add(list(drug_dict[drug])[0])
```

```
Copper || ParaGard
Dienogest / estradiol || Natazia
Ethinyl estradiol / etonogestrel || NuvaRing
Mestranol / norethindrone || Necon 1 / 50
Ulipristal || ella
```

With so few pairs, we can Google the names to determine which names of a pair are generic and brand names.

In [108]:
```
new_brands = [
    'ParaGard', 'Natazia', 'NuvaRing', 'Necon 1 / 50', 'ella'
]

brands.extend(new_brands)

len(brands)
```

Out[108]: 171

In [109]:
```
for drug in new_brands:
    generics.append(list(drug_dict[drug])[0])

len(generics)
```

Out[109]: 18

At this point, we have sorted all the paired brand and generic drug names. What remains is to identify whether each of the single drug names is a generic or brand name.

Let's look at them.

In [110]:
```
unpaired_drug_names
```

Out[110]: ['Ethinyl estradiol / folic acid / levonorgestrel', 'Nonoxynol 9']

A simple Google search confirms these are both generic names, so we'll add them as such.

In [111]:
```
generics.extend(unpaired_drug_names)

generics.sort()
```

Now we create a more universal drug naming system whereby every record is identified with its generic name.

In [112]:
```
def generic_fix(drugList):
    if len(drugList) == 1 or drugList[0] in generics:
        return drugList[0]
    else:
        return drugList[1]

df['genericName'] = df.drugList.apply(lambda x: generic_fix(x))
```

In [113]:
```
def full_brand_fix(drugList):
    if len(drugList) == 1:
        return None
    elif drugList[0] in brands:
        return drugList[0]
    else:
        return drugList[1]

df['fullBrandName'] = df.drugList.apply(lambda x: full_brand_fix(x))
```

In [114]:
```
brand_dict = {}

for fullName in brands:
    name = fullName
    tail = name.split()[-1]
    while tail.isnumeric() or tail in ['Fe', 'Lo', 'One-Step', '/', '1.5', 'Contraceptive']:
        name = name[:len(name)-len(tail)-1]
        tail = name.split()[-1]
    head = name.split()[0]
    while head in ['Lo', '/']:
        name = name[len(head)+1:]
        head = name.split()[0]
    brand_dict[fullName] = name

df['shortBrandName'] = df.fullBrandName.apply(lambda x: None if x == None else brand_dict[x])
```

Duplicates due to multiple user entry

Now we'll turn to more possible duplicate instances. We suspect the same user has copy-pasted an identical review multiple times when that verbatim review appears for the same condition and (generic) drug name with the same rating. Let's look at all such instances.

In [115]:
```
len(df[df.duplicated(subset=['genericName', 'condition', 'review', 'rating'])])
```

Out[115]: 8

In [116]:
```
df[df.duplicated(subset=['genericName', 'condition', 'review', 'rating'])] \
[['genericName', 'condition', 'review', 'rating', 'date']]
```

Out[116]:

	genericName	condition	review	rating	date
38469	Levonorgestrel	Emergency Contraception	"me and my boyfriend were having sex and the c...	10.0	August 4, 2015
88315	Ethinyl estradiol / levonorgestrel	Birth Control	"I have had a great experience on Chateal. I u...	10.0	August 29, 2016
97393	Ethinyl estradiol / levonorgestrel	Birth Control	"I am almost on month 3 of this birth control ...	10.0	September 22, 2015
105971	Ethinyl estradiol / norethindrone	Birth Control	"Was on Junel Fe and it worked according to my...	1.0	December 6, 2016
126179	Levonorgestrel	Emergency Contraception	"Hey gals! In a 'the heat of the moment' situa...	10.0	June 6, 2015
129993	Ethinyl estradiol / norgestimate	Birth Control	"I began taking Mononesse April 28th 2015. I w...	1.0	October 4, 2015
155176	Levonorgestrel	Emergency Contraception	"I, like many others, was having protected sex...	1.0	July 12, 2015
193929	Etonogestrel	Birth Control	"I have recently got my second implant inserte...	8.0	November 2, 2016

The review texts all appear to be unique. As long as the review and its duplicate appear close in time to one another (within days), then these should be collapsed into a single review with the respective useful counts added together.

First we'll check on the dates. The following cell will show the respective dates of when these duplicated reviews appeared.

In [117]:
```
for ind in df[df.duplicated(subset=['genericName', 'condition', 'review', 'rating'])].index:
    two_indices = list(df[df.review == df.loc[ind].review].index)
    print(df.loc[two_indices[0]].date, '... and ...', df.loc[two_indices[1]].date)
```

```
August 4, 2015 ... and ... August 4, 2015
August 29, 2016 ... and ... August 29, 2016
September 22, 2015 ... and ... September 22, 2015
December 6, 2016 ... and ... December 6, 2016
June 6, 2015 ... and ... June 6, 2015
October 4, 2015 ... and ... October 4, 2015
July 12, 2015 ... and ... July 12, 2015
November 3, 2016 ... and ... November 2, 2016
```

They're all identical dates except one that is a day apart.

We'll collapse these into single records and add the useful counts.

In [118]:
```
for ind in df[df.duplicated(subset=['genericName', 'condition', 'review', 'rating'])].index:
    two_indices = list(df[df.review == df.loc[ind].review].index)
    x, y = two_indices[0], two_indices[1]
    count = int(df.loc[x].usefulCount + df.loc[y].usefulCount)
    df.at[x, 'usefulCount'] = count
    df.drop([y], inplace=True)
```

In [119]:
```
# DO NOT re-run this cell out of sequence
# to use the dataframe as it was at this stage, un-comment, run, and re-comment the cell that
df_bookmark_3 = df.copy()
```

In [120]:
```
# df = df_bookmark_3.copy()
```

Identifying birth control method

Consider redoing this, where you pass through everything one time and assign everything to its method (other than pill) and assign everything that remains to pill by default.

The problem with this would be the nonoxynol and the levonorgestrel. The former is a spermicide, which we're dropping because there are only two records. The latter we determined applied to emergency (not pill!) ... BUT what if there are brands associated with "the pill" that are also levonorgestrel? This might be worth investigating either way.

In [121]:
```
birth_control_dict = {
    'IUD': ['Skyla', 'Mirena', 'Kyleena', 'Liletta', 'ParaGard'],
    'patch': ['Ortho Evra', 'Xulane'],
    'implantable': ['Implanon', 'Nexplanon'],
    'vaginal': ['NuvaRing'],
    'injectable': ['Depo-Provera', 'depo-subQ provera', 'Provera'],
    'emergency': ['Plan B', 'ella', 'Fallback Solo', 'Aftera', 'Take Action', 'Next Choice',
                  'My Way', 'EContra EZ'],
    'pill': ['Yasmin', 'Ortho Tri-Cyclen', 'Alesse', 'Aviane', 'Sprintec', 'Tri-Sprintec', 'M
             'Seasonique', 'Yaz', 'Lutera', 'Portia', 'Camila', 'Apri', 'Beyaz', 'Desogen', '
             'TriNessa', 'Zarah', 'Estarylla', 'Mononessa', 'Gianvi', 'Jolivette', 'Loestrin'
             'Ortho-Cyclen', 'Ortho-Novum', 'Necon', 'Femcon', 'Marlissa', 'Aubra', 'Viorele'
             'Norlyda', 'Ortho Cyclen', 'Lybrel', 'Pirmella', 'Larin', 'Tarina', 'Previfem',
             'Lessina', 'Elinest', 'Cryselle', 'Ortho-Cept', 'Falmina', 'Altavera', 'Tri-Lo-Ma
             'CamreseLo', 'Philith', 'Dasetta', 'Gildess', 'Ovral', 'Jencycla', 'Tri-Linyah',
             'Orsythia', 'Sronyx', 'Velivet', 'Reclipsen', 'Nikki', 'Levlen', 'Loryna', 'Julel
             'Zenchent', 'Tri-Previfem', 'Lyza', 'Seasonale', 'Mono-Linyah', 'Alyacen', 'Opci
}
```

In [122]:
```
def method_fix_1(shortBrandName):
    for method in birth_control_dict:
        if shortBrandName in birth_control_dict[method]:
            return method
    else:
        return None

df['method'] = df.shortBrandName.apply(lambda x: None if x == None else method_fix_1(x))
```

In [123]:
```
generic_dict = {
    'patch': ['Ethinyl estradiol / norelgestromin'],
    'IUD': ['Copper'],
    'implantable': ['Etonogestrel'],
    'emergency': ['Ulipristal'],
    'vaginal': ['Ethinyl estradiol / etonogestrel'],
    'pill': ['Ethinyl estradiol / levonorgestrel',
             'Drospirenone / ethinyl estradiol / levomefolate calcium',  'Mestranol / norethi
             'Ethinyl estradiol / norgestimate', 'Ethinyl estradiol / norethindrone',
             'Norethindrone', 'Drospirenone / ethinyl estradiol', 'Desogestrel / ethinyl estra
             'Ethinyl estradiol / norgestrel', 'Ethinyl estradiol / folic acid / levonorgestre
             'Ethinyl estradiol / ethynodiol', 'Dienogest / estradiol'],
    'injectable': ['Medroxyprogesterone']
}
```

In [124]:
```
def method_fix_2(generic):
    for method in generic_dict:
        if generic in generic_dict[method]:
            return method
    else:
        return None

df['method'] = df.apply(lambda x: method_fix_2(x.genericName) if x.method == None else x.meth
```

In [125]:
```
df[
    (df.method.isna())
].genericName.value_counts()
```

Out[125]:
```
genericName
Levonorgestrel    173
Nonoxynol 9         2
Name: count, dtype: int64
```

In [126]:
```
for ind in df[df.genericName == 'Nonoxynol 9'].index:
    show_review(ind)
```

```
Review #172606 | Rating: 1.0 | Upvotes: 2

I have a 44 year old daughter that is proof that Delfen Foam is not effective.

Review #209857 | Rating: 9.0 | Upvotes: 5

This was and is still a  nightmare contraceptive.  Bought this and used with my boyfriend. U
sed 3 in one night.  We both had side effects.  We are exclusive if not. ..I would have thou
ght I caught something.  Felt good after using but 2 hours later the excess is flowing out &
leaves me feeling slightly irritated.  I wash again but still feel a persistent irritation.
Day 2. he had a weird sensation while peeing.  Both of us have itchy genitals and it only ge
ts worse from there.  I almost feel like a yeast infection.  My anal region itches.my vagina
l area feels dry but it is not.  It is just highly irritated.  I have used raw yogurt, acido
philus pulls and so feel irritated.  Never using this again. It is day 4 and these symptoms
are still annoying
```

We could create a spermicide label, but there would only be two records for it, so we'll just drop them.

In [127]:
```
df.drop([172606, 209857], inplace=True)
```

In [128]:
```
for ind in df[
    (df.genericName == 'Levonorgestrel') &
    (df.method.isna())
].index[:10]:
    show_review(ind)
```

Review #585 | Rating: 10.0 | Upvotes: 0

My girlfriend and I had sex during her fertile period, the condom broke and we did not realize. She took the pill around 8 hours later and it worked, she got her period! I have read in some journals that the pill may not work during the fertile period so be careful! Anyhow for us worked :)

Review #1151 | Rating: 10.0 | Upvotes: 1

I NEVER write reviews but today I decided to because when I was looking through all the reviews here it really helped me. so I am going to pay it back and hope to help someone else .... I have never taken this pill before, never really had to but there is always a first time for everything ! So, I took it about 3 hours after my bf finished inside of me . I was freaking out !! I freaked out everyday until today when I finally got my period .. I had sex 9/10, and was supposed to get my period 9/26, I got it 9/25. No side effects except mild nausea , and cramps all the way until today ! But it worked. So let us thank God and all these people that work making these drugs for us ! Hope i could help someone with this review

Review #1741 | Rating: 10.0 | Upvotes: 0

I am 255lbs/5'1 height and 33 years old. I say this because I read many reviews and doctor's explanation that the pill sometimes does not work on obese or overweight people. It worked for me. I took it within 4 hours of being intimate with my boyfriend. You feel like you are pregnant because for at least 5 days I felt like activity was going on in my uterus, and the day before my period arrived I had really bad cramps, when I have not had cramps since I was in high school, at 120lbs and a size 2. Yes, it throws off your period. I was not due to have my period for another 2.5 weeks. However, my period came 7 days after being intimate and taking the pill, immediately. So, it works! You might be a little paranoid and feel like your pregnant because of cramps and what feels like activity going on in your uterus, but it works!

Review #2298 | Rating: 10.0 | Upvotes: 46

My boyfriend and I had sex on 1/11/14 and the condom fell off. Around 1 hours later I took the morning after pill and I am happy to say that I got my period today 3 days sooner! (This was also around the time the calendar said I was ovulating) so do not freak out! As long as you take it as soon as possible you will be fine!

Review #5925 | Rating: 10.0 | Upvotes: 8

I took the emergency contraception pill almost 5 days before my ovulation. I took it within 3 hours of the mishap with the condom. I was worried sick thinking I could be pregnant. My period was meant to come on a Monday, but it came on Sunday, which was 6 days late. I was so scared, I nearly cried, but I did end up getting my period. I am so relieved. All that stress did not help. If you have taken the Emergency Cobtraceptive Pill, do not worry if your period is a bit delayed.

Review #6509 | Rating: 10.0 | Upvotes: 1

I took the pill at the beginning of December. My boyfriend and I had been using protection but we both felt like something was different one day so I went and got the pill. I got my period one day earlier than normal almost three weeks later. We were both happy and not worried until I was supposed to start my period in January. We were both very stressed because I had been cramping yet I was more than a week late. One day we were going to go get a pregnancy test but I started my period at school. My period has become much heavier and I have worse cramps than before. Still I was very happy to get my period. So do not worry if you are a couple days late.

Review #7825 | Rating: 1.0 | Upvotes: 0

Took this pill at about 40 hrs post said accident. We are now pregnant. As I am a mother to two, we were not planning on anymore kids and are extremely careful. We have taken plan B in the past and I guess we should have this time as well.

Review #7870 | Rating: 10.0 | Upvotes: 0

Hi I promised that if it worked I would write a review here. I had sex the day after my period (April 25) and I took it just an hour later, he did not ejaculate inside me but it was unprotected and there was precum. I took 4 pills of levonorgestrel in the morning then 4 again 12 hours later. My period was supposed to come May 16 but it was late by 11 days. Levo will

```
really mess up your cycle and give you sore breasts, mood swings and everything that might m
ake you pregnant. Take a pregnancy test 21 days after sex and then again a week after! Do no
t worry girls, just stick to the facts and get it as soon as possible before your ovulation
(it might not work after). 10/10! Stay safe and do not make decisions about sex without stay
ing protected.

Review #10671 | Rating: 10.0 | Upvotes: 5

My boyfriend and I had one too many drinks and we decided to have unprotected sex without us
ing the pull out method! Well, I took the pill about 10 hours after for my very first time.
I had no major side effects after taking the pill. After an hour or two of taking the pill I
became very sleepy. I had my time of month a week earlier, and I was more hormonal than usua
l. I did not imagine the pill being a little expensive but I rate it a 10 because it worked
and it is sure worth the cost.

Review #11447 | Rating: 5.0 | Upvotes: 0

On June 5 at midnight, My boyfriend & I fooled around without a condom on my fertile days he
did not finish or anything or put it all the way in but getting pregnant through even some p
re cum is possible just unlikely. Just to be safe I took one around 10am on June 7 because I
already have a 4 month old. I feel okay. A little dizzy , cramps nauseous, not anything craz
y. I am sure it will go well. Hoping I can avoid taking it again though! Will post another e
xperience about when my period comes this month!
```

All of these examples describe emergency contraception.

In [129]:
```
generic_dict['emergency'].append('Levonorgestrel')
df['method'] = df.apply(lambda x: method_fix_2(x.genericName) if x.method == None else x.metho
```

In [130]:
```
df[
    (df.method.isna())
].genericName.value_counts()
```

Out[130]:
```
Series([], Name: count, dtype: int64)
```

Now that all records have been assigned a method, we can turn on this feature of the `show_review` function.

In [131]:
```
show_method=True
```

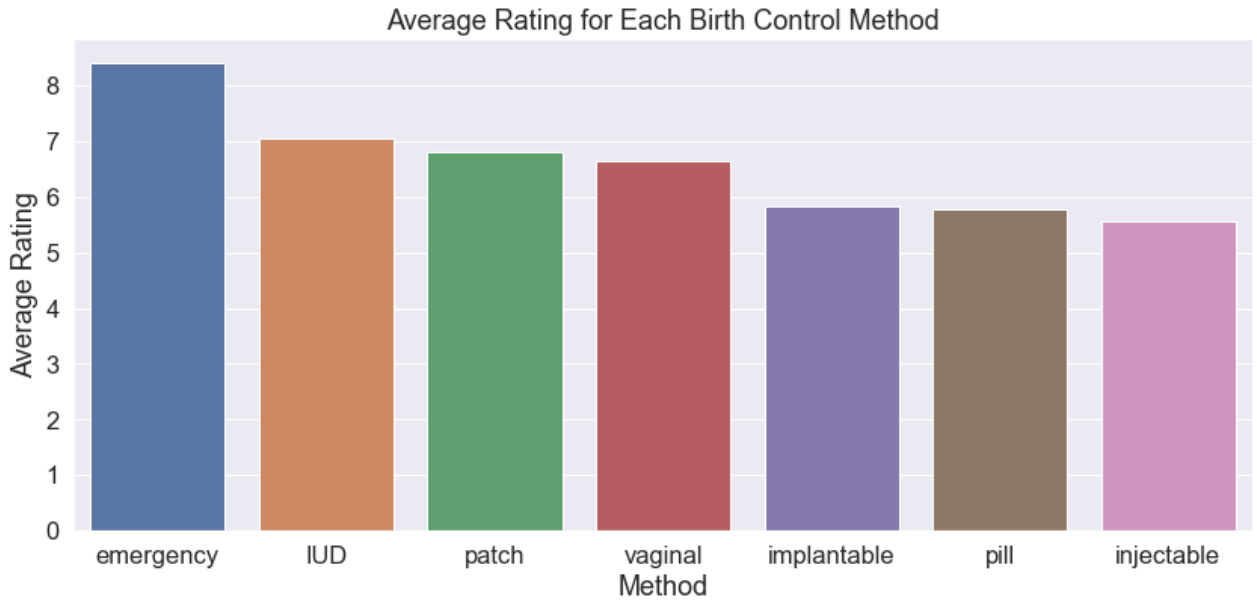
In [132]:
```
# DO NOT re-run this cell out of sequence
# to use the dataframe as it was at this stage, un-comment, run, and re-comment the cell that
df_bookmark_4 = df.copy()
```

In [133]:
```
# df = df_bookmark_4.copy()
```

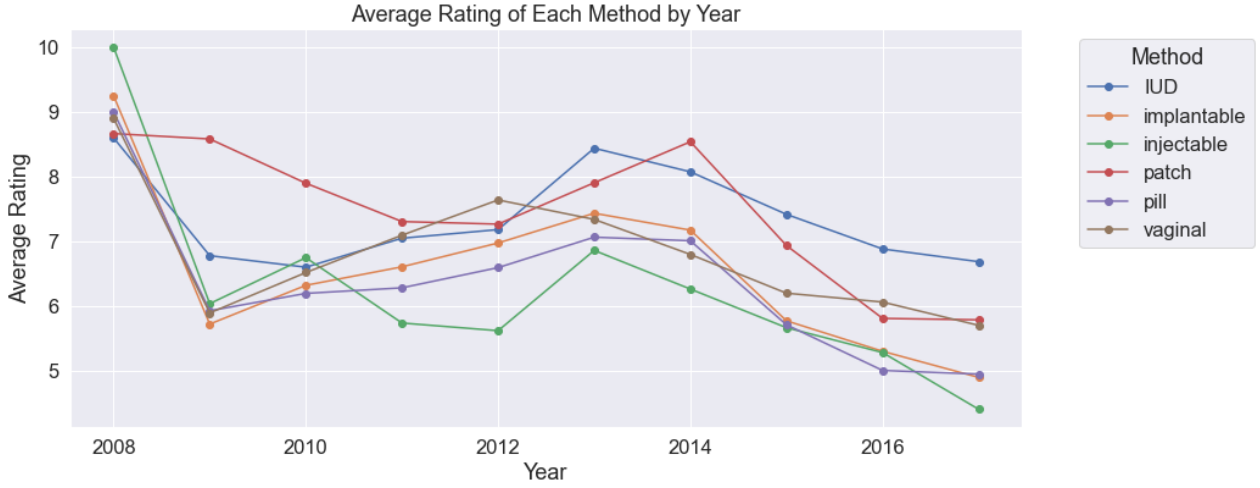
Exploration

Rating over time

In [134]:
```
# Calculate average rating for each method
average_ratings = df.groupby('method')['rating'].mean().reset_index()

# Create a bar chart using Seaborn
plt.figure(figsize=(14, 6))

# Set the font scale to increase text size
sns.set(font_scale=1.5)
sns.barplot(
    x='method',
    y='rating',
    data=average_ratings,
    order=average_ratings.sort_values('rating', ascending=False)['method'],
    palette='deep'
)
plt.xlabel('Method')
plt.ylabel('Average Rating')
plt.title('Average Rating for Each Birth Control Method')
plt.show()
```



In [135]:
```
# Extract year from datetime
df['year'] = df['datetime'].dt.year

# Group by 'year' and 'method', calculate the average rating for each group
average_ratings = df[df.method != 'emergency'].groupby(['year', 'method'])['rating'].mean().r

# Pivot the DataFrame to have 'method' as columns and 'year' as index
pivot_table = average_ratings.pivot_table(index='year', columns='method', values='rating')

# Plot the data
pivot_table.plot(kind='line', marker='o', figsize=(14, 6))
plt.title('Average Rating of Each Method by Year')
plt.xlabel('Year')
plt.ylabel('Average Rating')
plt.legend(title='Method', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.grid(True)
plt.show()
```

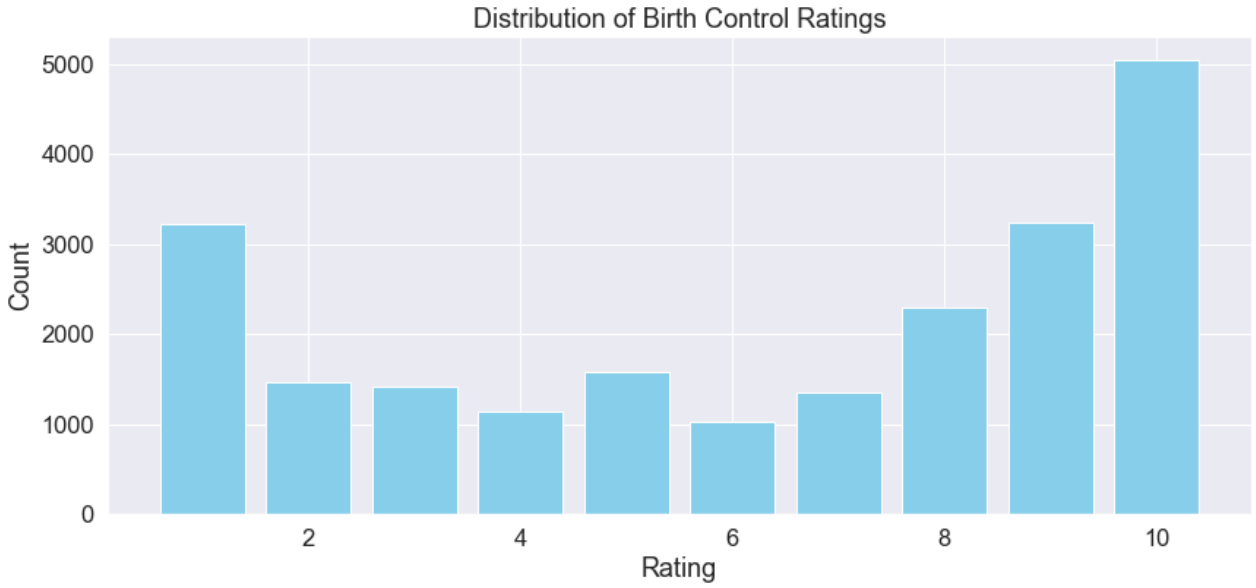


Observations:

- Ratings generally declined after the number of ratings increased significantly in 2015.
- Emergency contraception has largely been more popular than other methods.
- IUDs, patches, and vaginal contraceptives tend to be more popular than pills and implantables.
- Injectable methods are consistently the least popular.

Rating and upvotes

In [136]:
```
# Count the occurrences of each rating
rating_counts = df['rating'].value_counts()

# Sort the index for better visualization
rating_counts = rating_counts.sort_index()

# Plotting the bar graph
plt.figure(figsize=(14, 6))
plt.bar(rating_counts.index, rating_counts.values, color='skyblue')
plt.xlabel('Rating')
plt.ylabel('Count')
plt.title('Distribution of Birth Control Ratings')
plt.show()
```

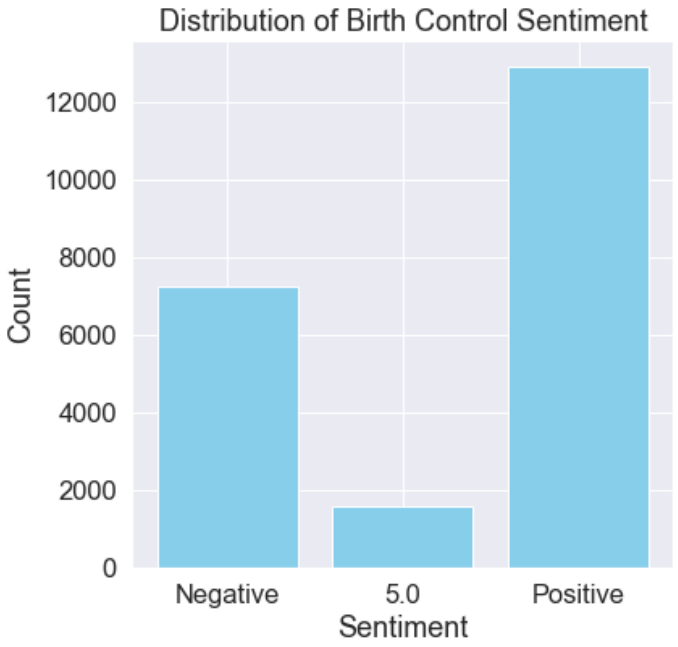


Observations:

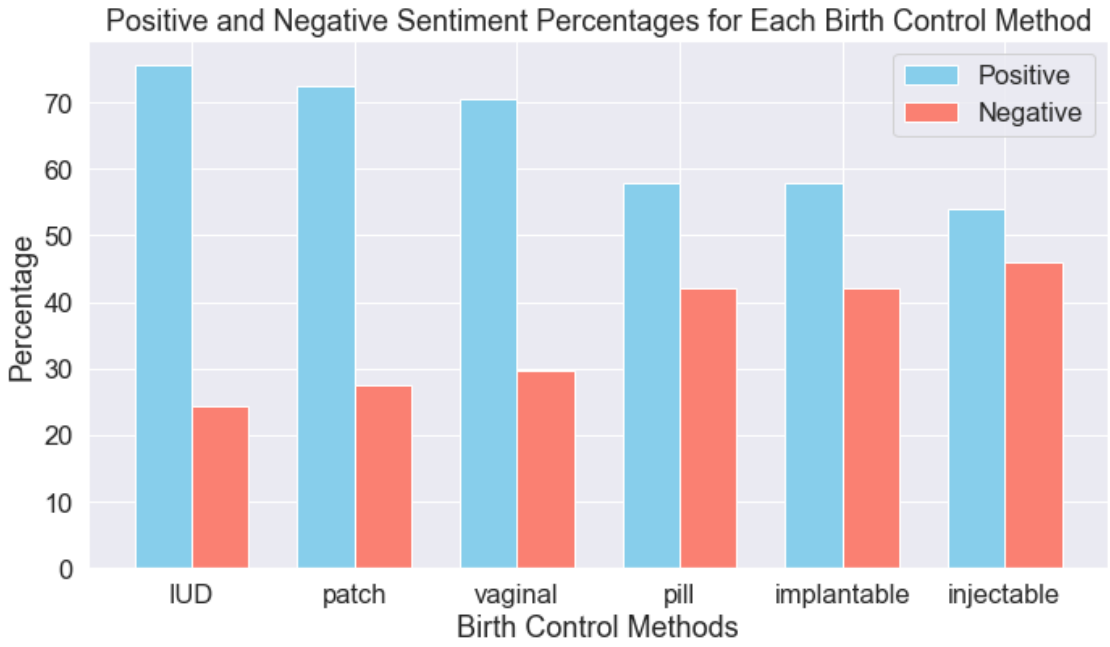
- There tend to be more positive ratings.
- Negative ratings are concentrated on the worst rating (1.0).
- There is a slight peak at 5.0. This indicates an attempt at neutrality.

In light of this distribution, we could recommend a 3-, 4-, or 5-point rating system that might be a better fit.

In [137]:
```
rating_counts = [
    len(df[df.rating < 5]),
    len(df[df.rating == 5]),
    len(df[df.rating > 5])
]

# Plotting the bar graph
plt.figure(figsize=(6, 6))
plt.bar([0, 1, 2], rating_counts, color='skyblue')
plt.xlabel('Sentiment')
plt.xticks([0, 1, 2], ['Negative', '5.0', 'Positive'])  # Setting custom tick labels
plt.ylabel('Count')
plt.title('Distribution of Birth Control Sentiment')
plt.show()
```



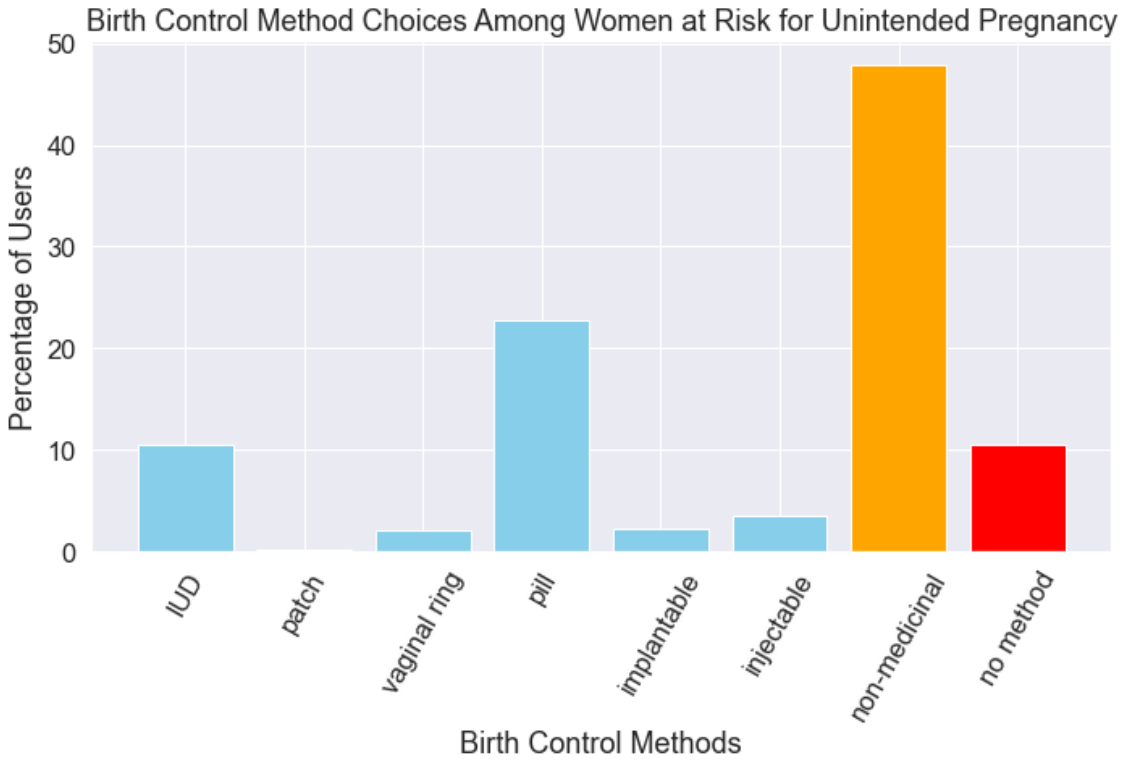
In [138]:
```
methods = ['IUD', 'patch', 'vaginal', 'pill', 'implantable', 'injectable']

positive_counts, negative_counts = [], []

for method_ in methods:
    positive_counts.append(len(df[(df.method == method_) & (df.rating > 5)]))
    negative_counts.append(len(df[(df.method == method_) & (df.rating < 5)]))

total_counts = [pos + neg for pos, neg in zip(positive_counts, negative_counts)]

# Calculate percentages for positive and negative counts
positive_percentages = [pos / total * 100 for pos, total in zip(positive_counts, total_counts
negative_percentages = [neg / total * 100 for neg, total in zip(negative_counts, total_counts

# Plotting the grouped bar graph
bar_width = 0.35
index = range(len(methods))

plt.figure(figsize=(10, 6))

plt.bar(index, positive_percentages, bar_width, label='Positive', color='skyblue')
plt.bar([i + bar_width for i in index], negative_percentages, bar_width, label='Negative', co

plt.xlabel('Birth Control Methods')
plt.ylabel('Percentage')
plt.title('Positive and Negative Sentiment Percentages for Each Birth Control Method')
plt.xticks([i + bar_width / 2 for i in index], methods)
plt.legend()
plt.tight_layout()
plt.show()
```



This shows the positive and negative sentiment directed at the various birth control methods. IUDs are the most "popular", by this metric, and injectables are the least popular.

In the cell below we will explore how commonly each method is used by women who are at risk for unintended pregnancy.

In [139]:
```
methods = ['IUD', 'patch', 'vaginal ring', 'pill', 'implantable', 'injectable', 'non-medicina
use_percentages = [10.6, 0.2, 2.1, 22.7, 2.3, 3.5, 47.9, 10.5]

colors = ['skyblue'] * (len(methods) - 2) + ['orange'] + ['red']  # Set colors for each bar

plt.figure(figsize=(10, 7))

plt.bar(methods, use_percentages, color=colors)

plt.xlabel('Birth Control Methods')
plt.ylabel('Percentage of Users')
plt.title('Birth Control Method Choices Among Women at Risk for Unintended Pregnancy')
plt.xticks(rotation=60)  # Rotate x-axis labels for better readability
plt.tight_layout()
plt.show()
```



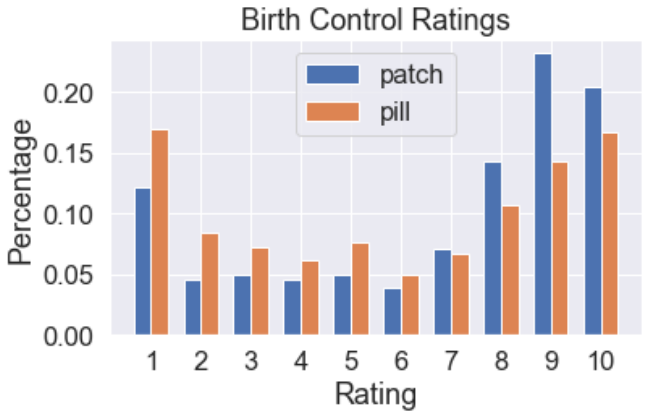
The above data comes from guttmacher.org (https://www.guttmacher.org/sites/default/files/factsheet/fb_contr_use_0.pdf).

The orange bar represents the percentage of women who are using non-medicinal methods (e.g. condoms, the rhythm method) to avoid unintended pregnancy. This represents a significant market segment for Pfizer. These women are all using methods that require some kind of behavior modification specific to instances of having sex. The 6 methods featured in our analysis all take effect in advance, whether sex happens or not.

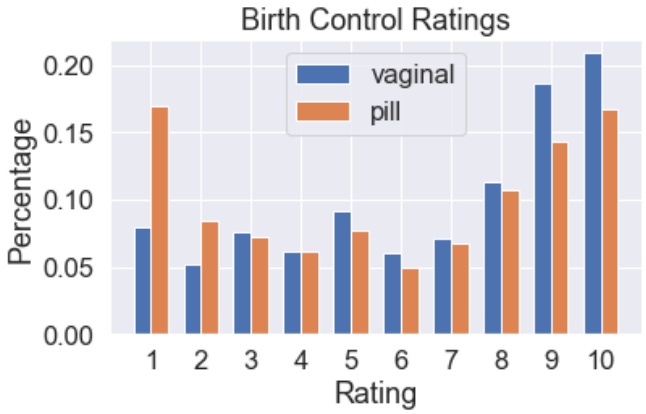
The red bar represents the percentage of women who are at risk for unintended pregnancy but are not engaging in any preventative method. They also represent a significant potential customer base.

In [140]:
```
def compare_methods(method_1, method_2):

    # Grouping data by BirthControl and Rating, then counting occurrences
    grouped = df[(df.method == method_1) | (df.method == method_2)].groupby(['method', 'rating

    # Calculating percentages for each birth control type
    patch_percentage = grouped.loc[method_1] / grouped.loc[method_1].sum()
    pill_percentage = grouped.loc[method_2] / grouped.loc[method_2].sum()

    # Creating the grouped bar plot
    fig, ax = plt.subplots()
    bar_width = 0.35
    index = range(1, 11)

    bar1 = ax.bar(index, patch_percentage, bar_width, label=method_1)
    bar2 = ax.bar([i + bar_width for i in index], pill_percentage, bar_width, label=method_2)

    ax.set_xlabel('Rating')
    ax.set_ylabel('Percentage')
    ax.set_title('Birth Control Ratings')
    ax.set_xticks([i + bar_width / 2 for i in index])
    ax.set_xticklabels(index)
    ax.legend()

    plt.tight_layout()
    plt.show()
```

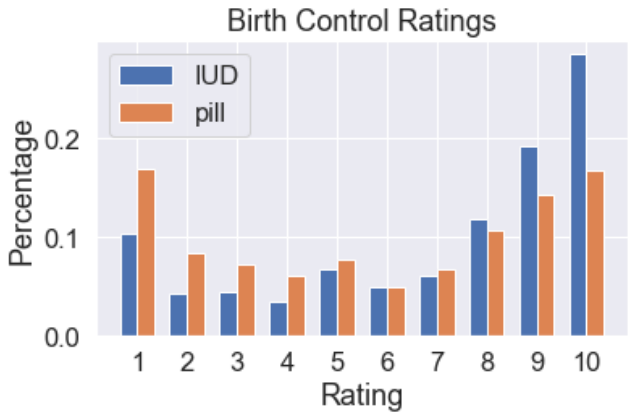
In [141]:
```
compare_methods('patch', 'pill')
```



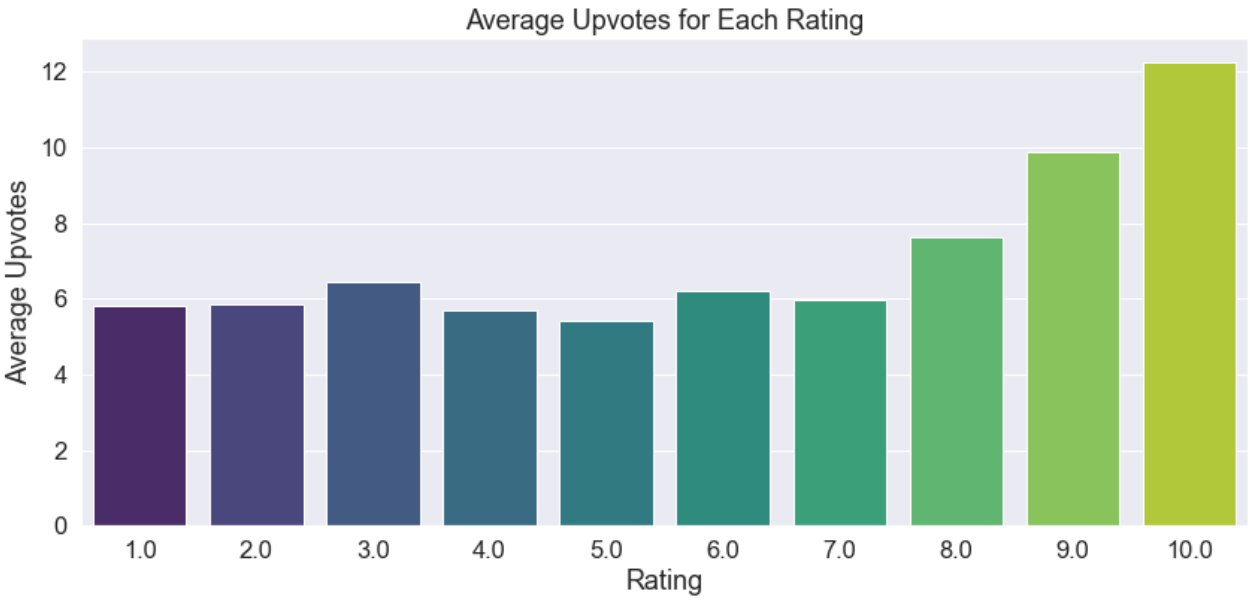
In [142]:
```
compare_methods('vaginal', 'pill')
```



In [143]:
```
compare_methods('IUD', 'pill')
```



In [144]:
```
# Calculate average usefulCount for each rating
average_useful_count = df.groupby('rating')['usefulCount'].mean().reset_index()

# Create a bar chart using Seaborn
plt.figure(figsize=(14, 6))
sns.barplot(x='rating', y='usefulCount', data=average_useful_count, palette='viridis')
plt.xlabel('Rating')
plt.ylabel('Average Upvotes')
plt.title('Average Upvotes for Each Rating')
plt.show()
```

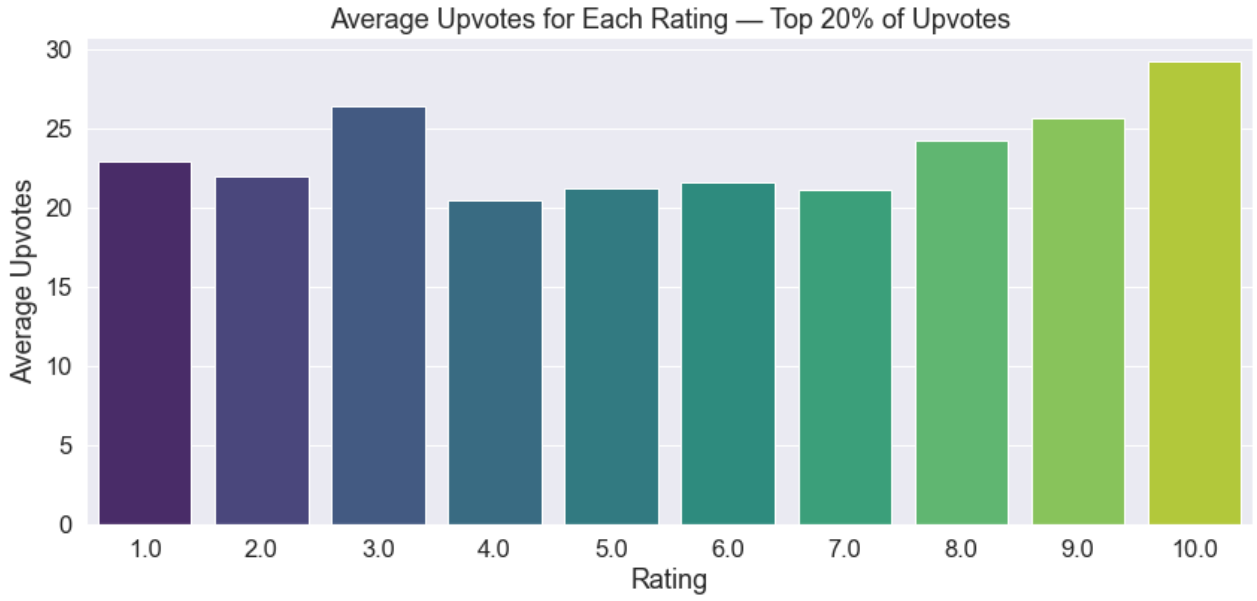


Observations:

- There is no real gain in usefulness other than for ratings 8 and above, which we saw earlier are also more prevalent than all other ratings except for 1.0, the lowest.

Let's look at only the top 20% most useful reviews.

In [145]:
```
culate average usefulCount for each rating
ge_useful_count = df[df.usefulCount > df.usefulCount.quantile(.80)].groupby('rating')['usefulC

ate a bar chart using Seaborn
igure(figsize=(14, 6))
arplot(x='rating', y='usefulCount', data=average_useful_count, palette='viridis')
label('Rating')
label('Average Upvotes')
itle('Average Upvotes for Each Rating — Top 20% of Upvotes')
how()
```



There's a relative spike among ratings of 3.0. Let's look at some highly upvoted reviews with a rating of 3.0.

In [146]:
```
for ind in df[df.rating == 3].sort_values(by='usefulCount', ascending=False).index[:10]:
    show_review(ind)
```

Review #197726 | Method: IUD | Rating: 3.0 | Upvotes: 204

I have had the Mirena for about 2 years now and have had nothing but problems with it. My ha
ir is falling out. I have gained weight. I do not want to have sex. I am depressed I now hav
e anxiety attacks. I have not had a period since getting it but I would rather have one then
be moody all the time!

Review #72776 | Method: vaginal | Rating: 3.0 | Upvotes: 174

For those of you experiencing negative side effects...
I was on NuvaRing for 4 years. During that time I became depressed, and cramps and back pai
n were unbearable. I have discontinued use of NR and feel better. All side effects are gon
e! When I discussed this with my Dr., she stated that some women naturally produce a normal
level of hormones. When these women are on birth control, it can cause all of these side ef
fects due to having an elevated level of hormones. The opposite effect happens to women tha
t do not naturally produce a normal level of hormones, leading them to actually feel better
and not have all of these side effects. Which is why some women do well and feel great on bi
rth control and others do not.

Review #6187 | Method: IUD | Rating: 3.0 | Upvotes: 140

I first would like to thank all of you that posted comments. After reading them, I felt I am
not alone. It is not just happening to me, there is nothing wrong with me. I am 34, with one
child and had Mirena for 5 months. The insertion was almost painless and the first month was
fine. Right now, I have gained 10lbs, loss of sex drive, have headaches, back pain, insomni
a, constipation, hair loss and general depression - OF COURSE that is going to cause to have
mood swings. I am waiting on an appointment to have it removed. ANOTHER IMPORTANT ASPECT is
that I had suffered from postpartum depression after the birth of my child 4 years ago and h
onestly I was worried that these side effects (especially depression) were a sign that I wa
s still not okay.

Review #71674 | Method: implantable | Rating: 3.0 | Upvotes: 132

Probably one of the worst decisions I have made in my adult life. I have always had great sk
in and on this medication it exploded with both random dryness and severe acne. My emotions
were a roller coaster (more than just normal girl stuff). My sex drive plummeted. Constant s
potting. Yeah It prevented me from getting pregnant probably because I was always on my peri
od. All in all bad experience.

Review #88347 | Method: injectable | Rating: 3.0 | Upvotes: 106

I had nothing but negative effects from the Depo shot that are continuing even after the dur
ation of action of the Depo. I am now experiencing terrible migraines (I rarely, if ever had
even minor headaches). They have been occurring multiple times per day including nausea and
diarrhea. I experienced these same symptoms for the first time after I received my first sh
ot. I also experienced neck stiffness, severe mood swings, bouts of depression that seemed t
o arrive from nowhere, no sex drive what so ever, and worst of all menstrual bleeding for an
entire month straight which was only counteracted by a high dosage of birth control pills.

Review #28138 | Method: vaginal | Rating: 3.0 | Upvotes: 86

I hate to be one of the many negative commenters on here, but I want others to know my side
effects that were related to the ring, that i did not at first think was related to it.

Severe anxiety and racing heart. back pain. Zero sex drive! My alcohol tolerance down to zer
o with severe vomiting, shivering and hangovers. VERY moody especially week before period.

I have been off all birth control for about 2-3 weeks...ALL symptoms have disappeared...amaz
ingly...and now I know it is related to the birth control, not me going insane.

Review #32788 | Method: injectable | Rating: 3.0 | Upvotes: 79

It started off great, for the first month. But then by the second month, my period became a
monster. It lasted for so long and started on a very random day that was not even close to m
y regular period cycle date. Also, I am on an emotional roller coaster ride and almost const
antly angry at something or someone. I do not like this at all.

Review #8410 | Method: emergency | Rating: 3.0 | Upvotes: 73

I took it just to be sure almost 40 hrs after the incident. I did not get pregnant. That sai

d, this was the first time I used any type of oral contraceptive EVER. So I had horrible sid e effects. Probably not because of he medicine as such but may be because of the high dosage of it. A day after I was completely exhausted and had spotting a week later. That week was t errible. I had gastric trouble, acidity, bloating, sort of menstrual cramps pretty much all the symptoms of pregnancy. Turns out these were the side effects for the medication as my bo dy was not used to sudden harmonal changes. So I sincerely suggest just use protection and b e on a regular pill. Be coutious if you have never used any type of contraceptive.

Review #6984 | Method: vaginal | Rating: 3.0 | Upvotes: 70

I am always tired and moody! I always feel sad and get angry and rageful very easily. I am e motional and I no longer feel the desire to do the things I used to enjoy, waking up is a ta sk every morning, my job has become more difficult and I feel like I have lost all my motiva tion, I cannot wait too be off the Nuvaring I am hoping I can go on an IUD or something bett er. I have not gained any weight and I have been on it for two years but the side effects ju st are not worth it. My husband and I seem to argue more and more. My breasts hurt all the t ime I feel nauseous and sick constantly and the smells of certain food make me sick. I might as well be pregnant.

Review #53333 | Method: implantable | Rating: 3.0 | Upvotes: 67

I have had Nexplanon since 11/5/14 and the whole first week I had extreme nausea, then bled for 3 weeks, was okay for about a month and now I am back to nonstop bleeding, extreme nause a, very anxious, crying all the time, acne, and I am just overall very unhappy and no longer myself. Having it removed next Monday but I am not sure I can wait.

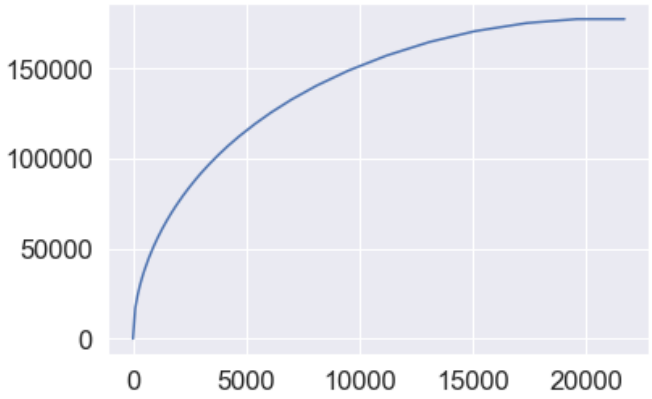
Nothing jumps out about these reviews except that they are certainly negative and could all probably have been 1.0 ratings.

In the cell below we'll examine the distribution of upvotes among the reviews.

In [147]:
```
uv = list(df.usefulCount.values)
uv.sort(reverse=True)

for threshold in list(range(10, 100, 10)):
    n = 0
    while sum(uv[:n]) < threshold / 100 * df.usefulCount.sum():
        n += 1
    print(str(int(n * 100 / len(df)))+'%','of the reviews received', str(threshold)+'%', 'of a
    print('This includes reviews with', uv[n], 'or more upvotes\n')

print('The remaining', str(100 - int(n * 100 / len(df)))+'%','of the reviews received', uv[n]
and made up the remaining 10% of all the upvotes.')
```

```
0% of the reviews received 10% of all the upvotes.
This includes reviews with 79 or more upvotes

2% of the reviews received 20% of all the upvotes.
This includes reviews with 41 or more upvotes

4% of the reviews received 30% of all the upvotes.
This includes reviews with 27 or more upvotes

8% of the reviews received 40% of all the upvotes.
This includes reviews with 20 or more upvotes

12% of the reviews received 50% of all the upvotes.
This includes reviews with 15 or more upvotes

19% of the reviews received 60% of all the upvotes.
This includes reviews with 11 or more upvotes

27% of the reviews received 70% of all the upvotes.
This includes reviews with 9 or more upvotes

38% of the reviews received 80% of all the upvotes.
This includes reviews with 6 or more upvotes

54% of the reviews received 90% of all the upvotes.
This includes reviews with 4 or more upvotes

The remaining 46% of the reviews received 4 or fewer upvotes and made up the remaining 10% o
f all the upvotes.
```

In [148]:
```
X = range(0,len(df),100)
Y = []
for x in X:
    Y.append(sum(uv[:x]))
fig, ax = plt.subplots()
ax.plot(X,Y);
```



This shows the cumulative sum of upvotes. For example, if we go 5% of the reviews in from the left on the x-axis, we would reach up to 33% of the total upvotes on the y-axis.

Word clouds and term frequency analysis

Stop words and preparatory lists, features and functions

We need stop lists for different purposes.

For certain ngram analysis, we might like to consider keeping stop words like "no" and "not". Doing so could prove very useful in recognizing the distinction between "I would recommend X" and "I would not recommend X". This will be `stop_list_light`.

In other cases, we really want to trim it down, and stop as many words as possible. This will be `stop_list_heavy`.

In [149]:
```
punctuation_list = [char for char in string.punctuation]
punctuation_list.extend(['', '``', "''", '...'])

# obtain the standard list of stopwords
nltk.download('stopwords', quiet=True)
# start our own list of stopwords with these words
stop_list_heavy = stopwords.words('english')
# stop words to keep
# 44-59 be/have/do verbs
# 64-178 prepositions/subordinate conjunctions/modals
stop_list_light = stop_list_heavy.copy()
stop_list_light = stop_list_light[:44] + stop_list_light[60:64]
# add punctuation characters
for char in string.punctuation:
    stop_list_light.append(char)
    stop_list_heavy.append(char)
# add misc other tokens
stop_list_light.extend(['', 'll', 're', 've', 'ha', 'wa', '``', "''"])
stop_list_heavy.extend(['', 'll', 're', 've', 'ha', 'wa', '``', "''"])
```

The stop words in the list below will be helpful in demonstrating what topics commonly feature in certain reviews. We can choose to ignore certain ordinary words, brand names, and words associated with discussing reproductive health generally (e.g. period, month). This will help us focus on the terms that better illuminate the reviewers' concerns.

In [150]:
```
extra_stops= ['pill', 'month', 'control', 'birth', 'day', 'first', 'week', 'year', 'get', \
              'time', 'side', 'would', 'effect', 'like', 'never', '3', 'got', 'started', 'also
              'feel', 'since', 'really', 'take', 'back', 'every', 'two', 'went', 'little', 'ex
              'even', 'put', 'recommend', 'taking', 'last', 'much', 'thing', 'almost', 'ever'
              'getting', 'still', 'could', 'go', 'made', '4', 'took', 'lot', '5', 'felt', 'are
              'three', 'using', 'say', 'second', 'mirena', 'skyla', 'iud', 'nuvaring', 'nuva'
              'thought', 'experienced', 'make', 'used', 'doe', 'implanon', 'nexplanon', 'alway
              'something', 'shot', 'always', 'depo', 'provera', 'away', 'injection', 'patch',
              'ortho', 'evra', 'think', 'yaz', 'noticed', 'taken', 'implant', 'paragard', 'pla
              'love'
             ]
```

`review_lower` will only make everything lower-case but not otherwise remove stop words, punctuation, or alter (lemmatize) anything.

In [151]:
```
df['review_lower'] = df.review.apply(lambda x: x.lower())
```

In [152]:
```
df.review_lower.head()
```

Out[152]:
```
2     "i used to take another oral contraceptive, wh...
3     "this is my first time using any form of birth...
6     "he pulled out, but he cummed a bit in me. i t...
9     "i had been on the pill for many years. when m...
14    "started nexplanon 2 months ago because i have...
Name: review_lower, dtype: object
```

Here we'll make a function that helps create lists of tokens from subsets of the dataframe.

In [153]:
```
# a function that takes a list of documents and 1) tokenizes them, 2) lemmatizes them, and 3)
# for example, to execute this function use make_tokens(df.review.tolist())
# IN: a list of documents // OUT: a list of tokens

lemmatizer = WordNetLemmatizer()

def make_tokens(docs_list, stop_list=stop_list_light):
    # join documents into a single string
    docs_joined = ' '.join(docs_list)
    # tokenize the single string into a list of tokens
    tokens = nltk.word_tokenize(docs_joined)
    # lemmatize the list of tokens
    tokens_lemmatized = [lemmatizer.lemmatize(word) for word in tokens]
    # stop the list of tokens
    tokens_stopped = [word for word in tokens_lemmatized if word not in stop_list]
    return tokens_stopped
```

In [154]:
```
%%time
# ⏰ record the time for this cell -- usually 1 minute

method_tokens = {}

for method_ in df.method.unique():
    stop_list_ = stop_list_heavy.copy()
    stop_list_.extend(extra_stops)
    tokens = make_tokens(df[df.method == method_].review_lower.tolist(), stop_list_)
    method_tokens[method_] = {}
    method_tokens[method_]['positive'] = make_tokens(df[(df.method == method_) \
                                                        & (df.rating > 5)].review_lower.to_lis
    method_tokens[method_]['negative'] = make_tokens(df[(df.method == method_) \
                                                        & (df.rating < 5)].review_lower.to_lis
```

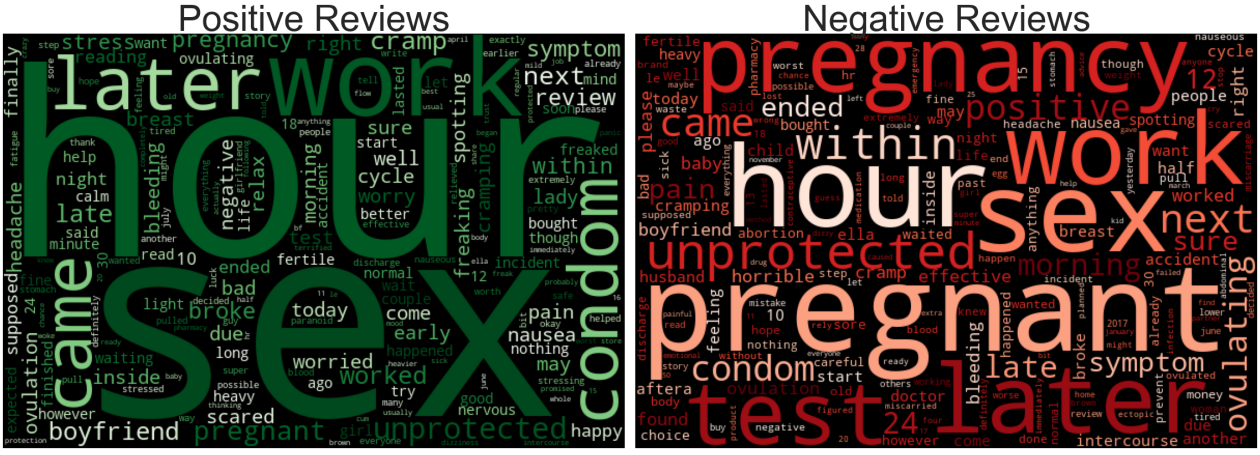
```
CPU times: user 51.9 s, sys: 302 ms, total: 52.2 s
Wall time: 52.4 s
```

Word clouds

The function below will display positive and negative word clouds for each birth control method.

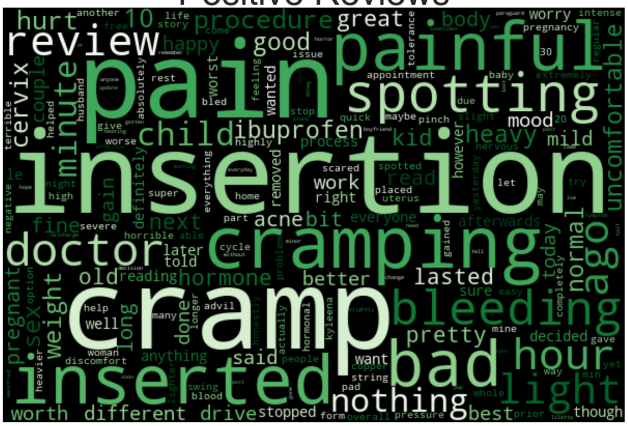
In [155]:
```
def method_word_clouds(method_):

    positive_text = (' ').join(method_tokens[method_]['positive'])
    negative_text = (' ').join(method_tokens[method_]['negative'])

    documents = [positive_text, negative_text]

    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(documents)

    positive_tfidf_scores = tfidf_matrix[0]  # TF-IDF scores for positive document
    negative_tfidf_scores = tfidf_matrix[1]  # TF-IDF scores for negative document

    feature_names = vectorizer.get_feature_names_out()

    # Create dictionaries with words and their corresponding TF-IDF scores for each document
    positive_word_scores = dict(zip(feature_names, positive_tfidf_scores.toarray()[0]))
    negative_word_scores = dict(zip(feature_names, negative_tfidf_scores.toarray()[0]))

    # Generate word clouds for positive and negative documents
    positive_wordcloud = WordCloud(
        width=600,
        height=400,
        colormap='Greens',
        collocations=True
    ).generate_from_frequencies(positive_word_scores)
    negative_wordcloud = WordCloud(
        width=600,
        height=400,
        colormap='Reds',
        collocations=True
    ).generate_from_frequencies(negative_word_scores)

    # show positive and negative wordclouds for apple side by side
    fig, ax = plt.subplots(figsize=(32,16), ncols=2)
#     fig.subplots_adjust(top=0.85, bottom=0.15, hspace=0.3)
    ax[0].imshow(positive_wordcloud)
    ax[0].set_title('Positive Reviews', fontsize=65)
    ax[0].axis('off')
    ax[1].imshow(negative_wordcloud)
    ax[1].set_title('Negative Reviews', fontsize=65)
    ax[1].axis('off')
    plt.tight_layout()
    plt.show()
```

In [156]:
```
method_word_clouds('emergency')
```

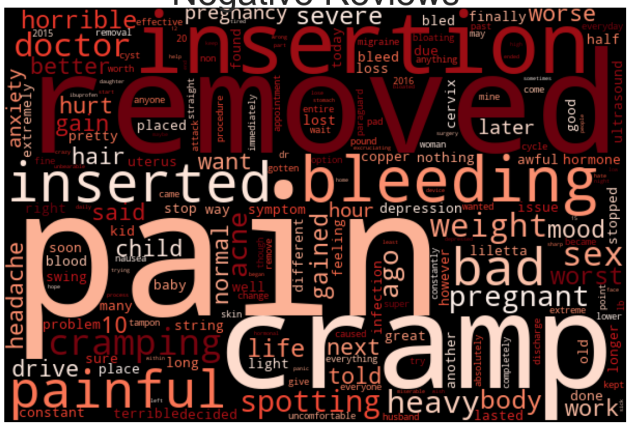


In [157]:
```
method_word_clouds('IUD')
```

Positive Reviews



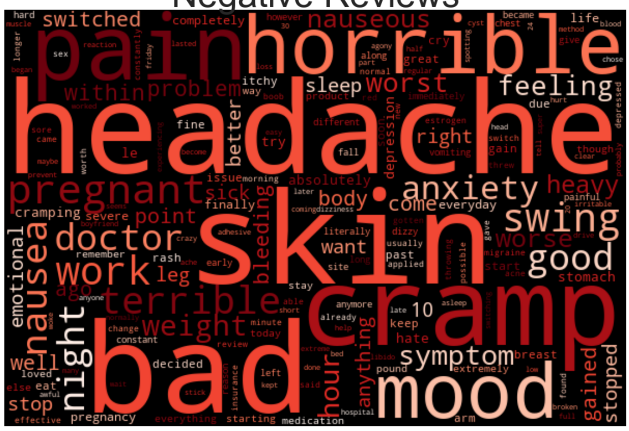
Negative Reviews



In [158]:
```
method_word_clouds('patch')
```

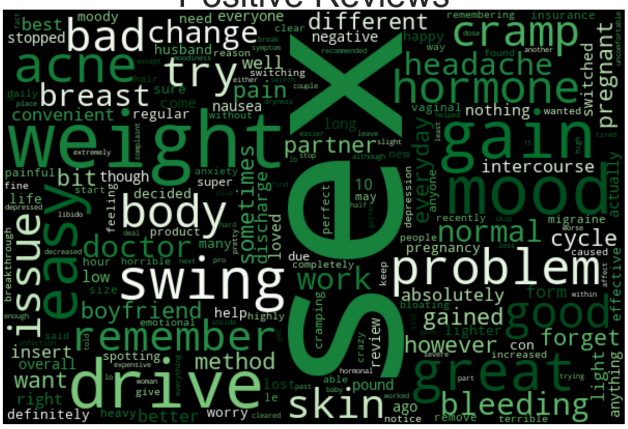
Positive Reviews

Negative Reviews



In [159]:
```
method_word_clouds('vaginal')
```

Positive Reviews



Negative Reviews

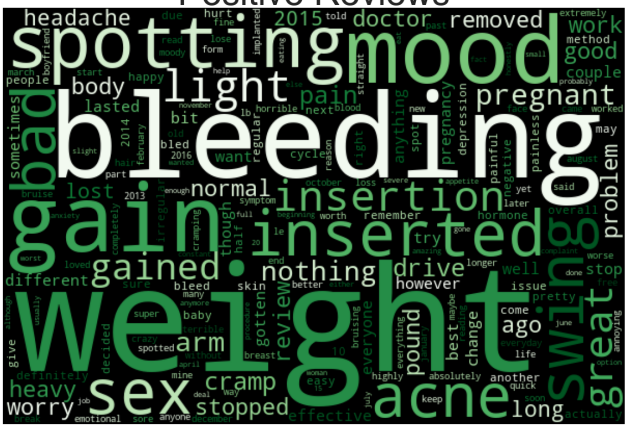
In [160]:
```
method_word_clouds('pill')
```

Positive Reviews

Negative Reviews

In [161]:
```
method_word_clouds('implantable')
```

Positive Reviews

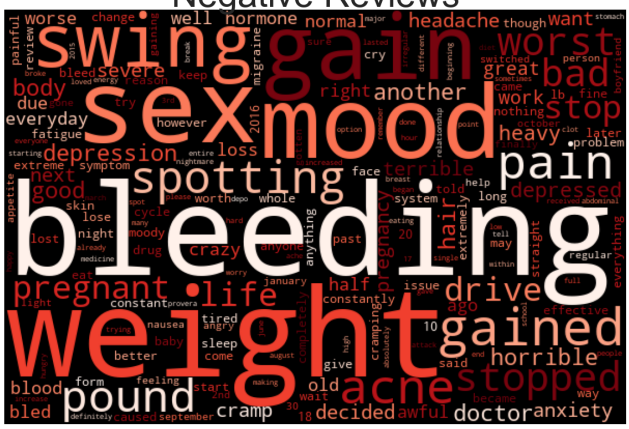


Negative Reviews

In [162]:
```
method_word_clouds('injectable')
```

Positive Reviews

Negative Reviews



In [163]:
```
for ind in df[
    (df.review_lower.str.contains('weight')) & \
    (df.method == 'patch') & \
    (df.rating > 5)
  ].sort_values(by='usefulCount', ascending=False).index[:3]:
    show_review(ind)
```

```
Review #163 | Method: patch | Rating: 10.0 | Upvotes: 115

This is absolutely the best birth control I have ever used. I switched from Nexplanon to Ort
ho Evra, and if you are thinking of doing the same, I highly recommend it.  Let me list the
reasons why:
1. Weight loss. I have lost 5lbs in one month, without even trying!  I am back to the weight
I was before I started birth control...but my breasts are still birth control sized.
2. Sex drive. Mine had been non-existent since I went on Nexplanon in September. Now, strong
er than ever. That might be a downside though, now that I think about it.
3. Skin- no acne!
4. Predictable cycle. (On Nexplanon, I never knew when I was going to menstruate, which was
about 90% of the time). Now, a non issue.
5. Do not have to remember everyday.

Review #38183 | Method: patch | Rating: 8.0 | Upvotes: 69

I was on Ortho Evra for almost 6 years (since my early 20s).  I have had very few issues wit
h it... no weight gain, no moodiness, no stale sex drive.  All I have experienced were sligh
t skin irritation (where patch was placed) and spotting.

I switched to NuvaRing under the guidance of my physician (because of my concerns over the F
DA warnings).  It was as easy as the patch, but I quickly gained 5-7 pounds over 2 months of
usage.  Then my doctor put me on Microgestin 1/20 FE.  It was the first and will be the last
birth control pill I ever take!  I have gained 15 pounds over 6 months, experienced excessiv
e night sweats and lost all interest in sex altogether.

Now, I might return to the patch.

Review #62994 | Method: patch | Rating: 10.0 | Upvotes: 45

No mood swings, no acne, no weight-gain, controlled period, protects me from being pregnant,
only changed once a week, same sex-drive, and increased a cup size! I love it. Sticky part w
ashes away and I rotate between shoulder blades, it does not fall off me, works great for me
and I recommend to anyone who has had other forms of birth control issues, this just might w
ork for you.
```

In [164]:
```
for ind in df[
    (df.review_lower.str.contains('bleeding')) & \
    (df.method == 'pill') & \
    (df.rating < 5)
  ].sort_values(by='usefulCount', ascending=False).index[:3]:
    show_review(ind)
```

```
Review #70814 | Method: pill | Rating: 3.0 | Upvotes: 56

This birth control did prevent me from getting pregnant which is the main concern. However,
I have never bled so much in my life. I am someone that has always had perfect periods witho
ut birth control. I went from that to bleeding three weeks out of the month and it never wou
ld stop. It was not necessarily heavy bleeding, but it is very aggravating to bleed constant
ly. This medicine also made me have acne and I normally have very clear skin. It made me hav
e mood swings that my husband noticed. He even asked me when his nice wife was coming back.
I decided after giving this three months to go to a permanent birth control method.

Review #6182 | Method: pill | Rating: 4.0 | Upvotes: 38

So I have been taking Lyza for 5 months now. If I am 1hour or more late on taking my pill th
en I will automatically start a bleeding cycle for 5 days at least. I have no clue when my p
eriod actually is anymore. I also have had a 15 pound weight gain. I have also been losing h
air, been depressed, and other odd things. I do not like this birth control at all, does not
work well with my body.

Review #12175 | Method: pill | Rating: 1.0 | Upvotes: 34

I am 21 years old and my son is now five months old. I started taking Jolivette at the begin
ning of February. I have experienced abnormal, heavy bleeding that seems like it never ends,
feels like I have bled more than I have not bled. The bleeding will stop for two to three da
ys then return and last for about two weeks. I have also been feeling more angry and stresse
d than ever since I started taking this birth control. I am switching to a different birth c
ontrol as soon as possible because Jolivette is making life miserable, and the way I am runn
ing through boxes of tampons and pads is ridiculous and draining the wallet.
```

Term frequency analysis

Word clouds are great, but a little unscientific. Here we'll explore the actual list of most important terms to each category of birth control method as determined by TFIDF scores.

In [165]:
```
reviews, corpus, tokens, tokens_joined = {}, {}, {}, {}

for method_ in df.method.unique():
    pos_join = ' '.join(method_tokens[method_]['positive'])
    neg_join = ' '.join(method_tokens[method_]['negative'])
    corpus[method_] = pos_join + ' ' + neg_join # the above as a single string with spaces
    tokens[method_] = nltk.word_tokenize(corpus[method_]) # the above as a list of words
    tokens_joined[method_] = ' '.join(tokens[method_]) # the above as a single string with spa

tokens_joined_list = [tokens_joined[method_] for method_ in df.method.unique()]
```

In [166]:
```
tfidf_vectorizer = TfidfVectorizer(ngram_range=(1,2), stop_words=stop_list_heavy)
tfidf_matrix = tfidf_vectorizer.fit_transform(tokens_joined_list)
```

In [167]:
```
%%time

for i, doc in enumerate(tokens_joined_list):
    print(f"Important words for {df.method.unique()[i]} method:")
    feature_names = tfidf_vectorizer.get_feature_names_out()
    feature_index = tfidf_matrix[i, :].nonzero()[1]
    tfidf_scores = zip(feature_index, [tfidf_matrix[i, x] for x in feature_index])
    top_words = sorted(tfidf_scores, key=lambda x: x[1], reverse=True)[:10]
    for word_index, score in top_words:
        print(f"{feature_names[word_index]} (TF-IDF Score: {score:.2f})")
    print()
```

Important words for pill method:
weight (TF-IDF Score: 0.29)
acne (TF-IDF Score: 0.25)
mood (TF-IDF Score: 0.21)
cramp (TF-IDF Score: 0.17)
gain (TF-IDF Score: 0.17)
swing (TF-IDF Score: 0.15)
mood swing (TF-IDF Score: 0.15)
sex (TF-IDF Score: 0.15)
bad (TF-IDF Score: 0.15)
bleeding (TF-IDF Score: 0.13)

Important words for patch method:
weight (TF-IDF Score: 0.19)
bad (TF-IDF Score: 0.17)
skin (TF-IDF Score: 0.16)
mood (TF-IDF Score: 0.16)
breast (TF-IDF Score: 0.15)
headache (TF-IDF Score: 0.15)
sex (TF-IDF Score: 0.13)
mood swing (TF-IDF Score: 0.13)
great (TF-IDF Score: 0.13)
swing (TF-IDF Score: 0.13)

Important words for emergency method:
sex (TF-IDF Score: 0.34)
hour (TF-IDF Score: 0.31)
work (TF-IDF Score: 0.26)
later (TF-IDF Score: 0.23)
condom (TF-IDF Score: 0.19)
came (TF-IDF Score: 0.18)
unprotected (TF-IDF Score: 0.18)
condom broke (TF-IDF Score: 0.17)
pregnant (TF-IDF Score: 0.17)
unprotected sex (TF-IDF Score: 0.17)

Important words for implantable method:
weight (TF-IDF Score: 0.33)
bleeding (TF-IDF Score: 0.27)
mood (TF-IDF Score: 0.20)
gain (TF-IDF Score: 0.17)
sex (TF-IDF Score: 0.17)
swing (TF-IDF Score: 0.16)
mood swing (TF-IDF Score: 0.16)
spotting (TF-IDF Score: 0.16)
inserted (TF-IDF Score: 0.16)
removed (TF-IDF Score: 0.15)

Important words for vaginal method:
sex (TF-IDF Score: 0.40)
weight (TF-IDF Score: 0.23)
mood (TF-IDF Score: 0.20)
drive (TF-IDF Score: 0.18)
sex drive (TF-IDF Score: 0.18)
gain (TF-IDF Score: 0.15)
mood swing (TF-IDF Score: 0.14)
swing (TF-IDF Score: 0.14)
great (TF-IDF Score: 0.13)
easy (TF-IDF Score: 0.12)

Important words for IUD method:
insertion (TF-IDF Score: 0.44)
pain (TF-IDF Score: 0.31)
cramp (TF-IDF Score: 0.30)
inserted (TF-IDF Score: 0.24)
cramping (TF-IDF Score: 0.20)
painful (TF-IDF Score: 0.18)
bad (TF-IDF Score: 0.16)
spotting (TF-IDF Score: 0.15)
bleeding (TF-IDF Score: 0.14)
doctor (TF-IDF Score: 0.12)

```
Important words for injectable method:
weight (TF-IDF Score: 0.39)
bleeding (TF-IDF Score: 0.24)
gain (TF-IDF Score: 0.22)
mood (TF-IDF Score: 0.17)
spotting (TF-IDF Score: 0.16)
gained (TF-IDF Score: 0.15)
weight gain (TF-IDF Score: 0.15)
sex (TF-IDF Score: 0.15)
swing (TF-IDF Score: 0.14)
mood swing (TF-IDF Score: 0.14)

CPU times: user 2min 9s, sys: 331 ms, total: 2min 9s
Wall time: 2min 10s
```

Feature Engineering

In [168]:
```
engineered_features = []
```

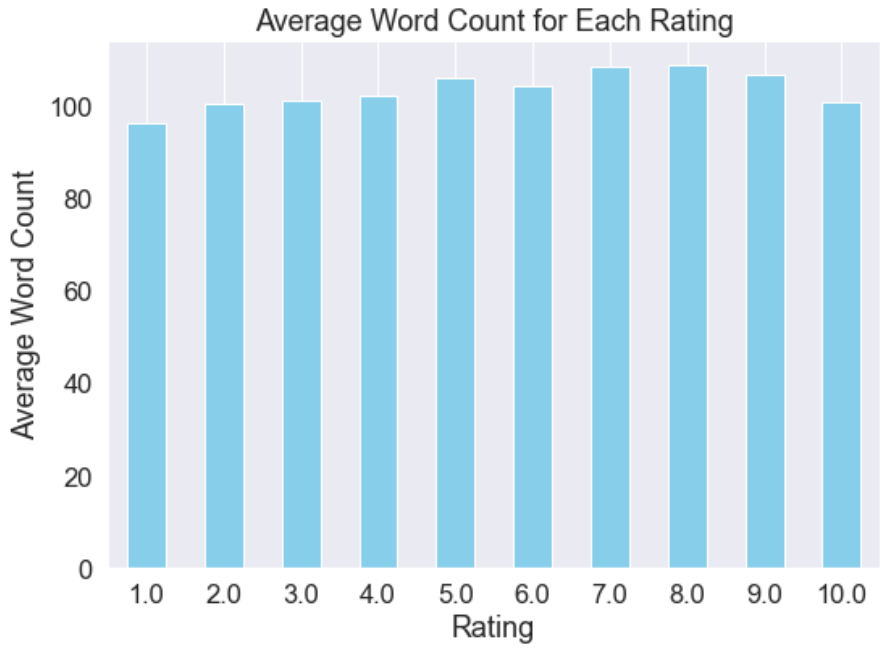
In [169]:
```
def rating_plot(col, feature_name):

    # Grouping by 'rating' and calculating the mean of 'but_count' for each rating
    count_by_rating = df.groupby('rating')[col].mean()

    # Creating the bar chart
    plt.figure(figsize=(8, 6))
    count_by_rating.plot(kind='bar', color='skyblue')
    plt.title(f'Average {feature_name} for Each Rating')
    plt.xlabel('Rating')
    plt.ylabel(f'Average {feature_name}')
    plt.xticks(rotation=0)  # Rotate x-axis labels if needed
    plt.grid(axis='y')  # Add gridlines for better readability
    plt.tight_layout()
    plt.show()
```

Word count

In [170]:
```
df['word_tokens'] = df.review_lower.apply(
    lambda x: [word for word in nltk.word_tokenize(x) if word not in punctuation_list])
```

In [171]:
```
df['word_count'] = df.word_tokens.apply(lambda x: len(x))
```

In [172]:
```
rating_plot('word_count', 'Word Count')
```



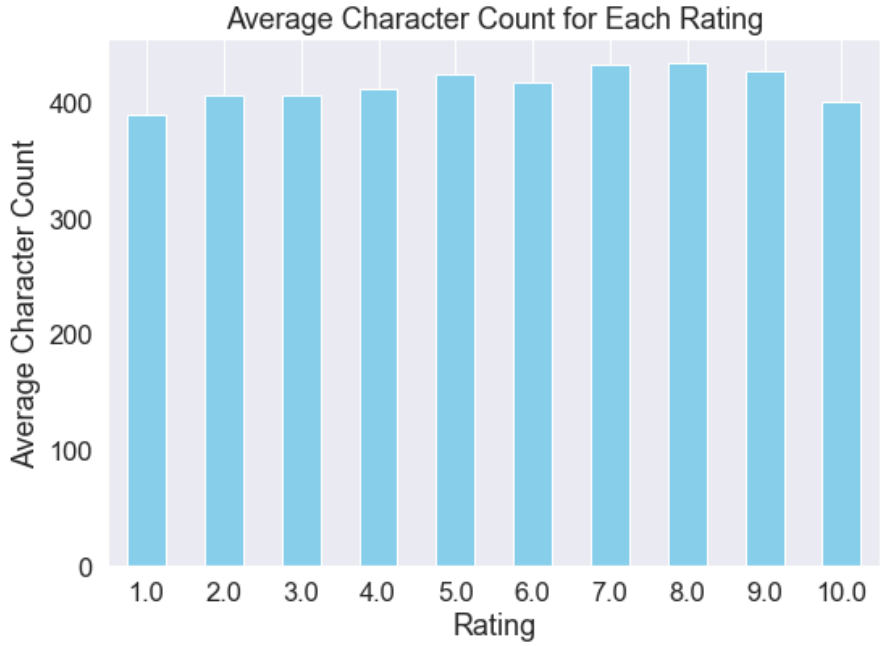
Word count doesn't correlate very much with rating.

In [173]:
```
engineered_features.append('word_count')
```

Character count

In [174]:
```
df['character_count'] = df.review_lower.apply(lambda x: len(re.findall(r'\w', x)))
```

In [175]:
```
rating_plot('character_count', 'Character Count')
```



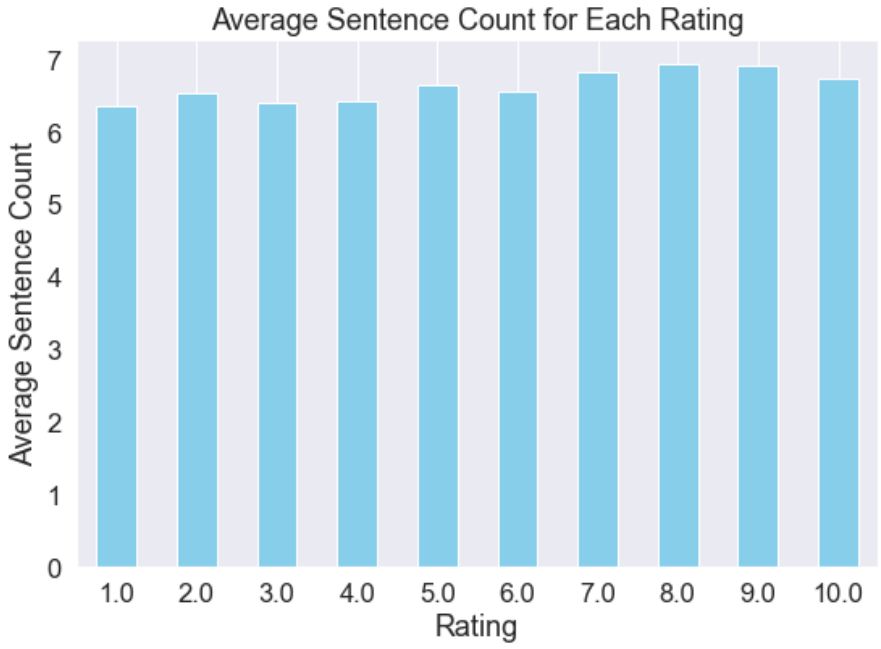
Character count doesn't correlate much with rating.

In [176]:
```
engineered_features.append('character_count')
```

Sentence count

In [177]:
```
df['sentence_count'] = df.review_lower.apply(lambda x: len(sent_tokenize(x)))
```

In [178]:
```
rating_plot('sentence_count', 'Sentence Count')
```



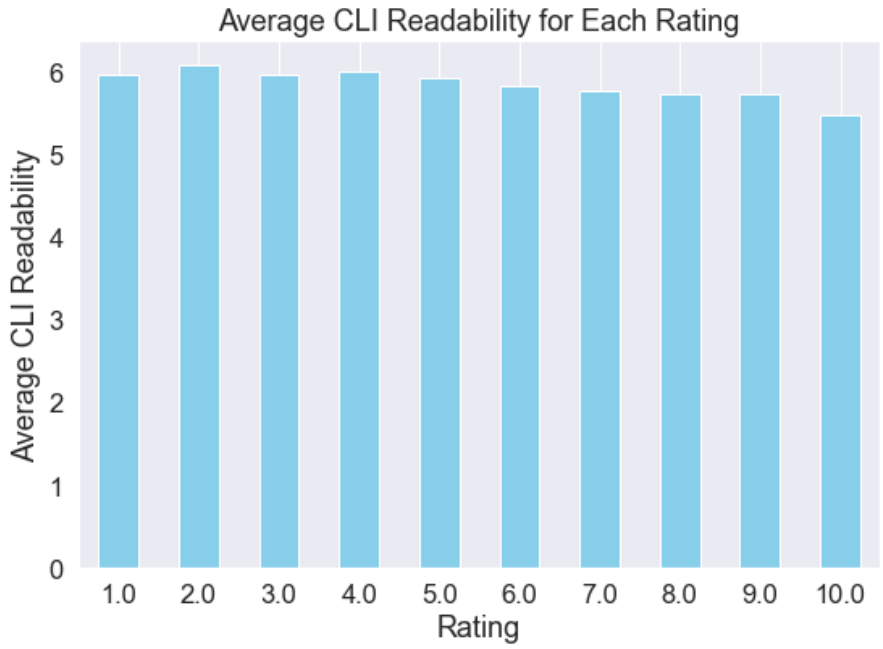
Sentence count doesn't correlate much with rating.

In [179]:
```
engineered_features.append('sentence_count')
```

Coleman-Liau Index

In [180]:
```
df['CLI_readability'] = df.apply(
    lambda x: 5.88 * x.character_count / x.word_count - 29.6 * x.sentence_count / x.word_coun
```

In [181]:
```
rating_plot('CLI_readability', 'CLI Readability')
```



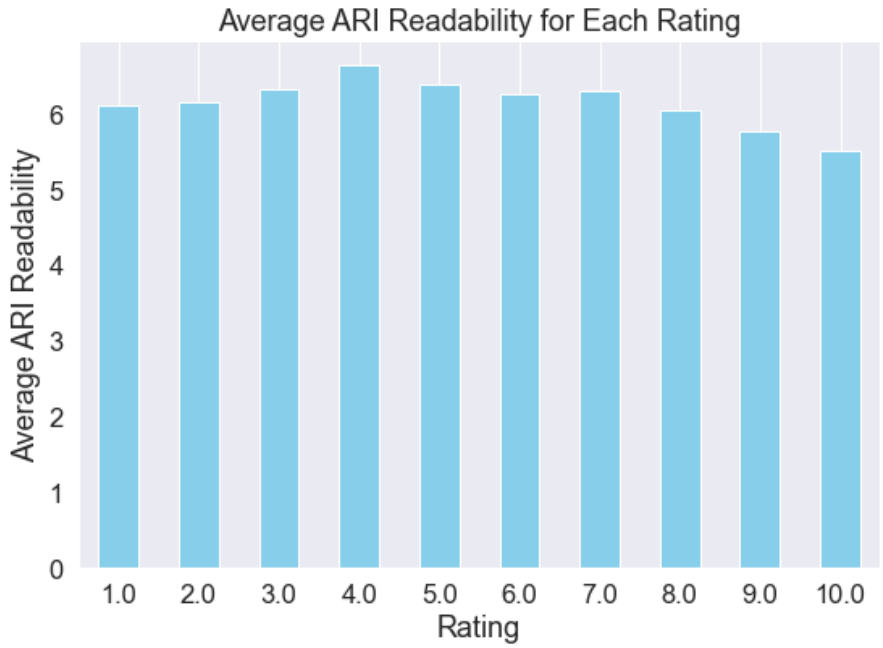
The CLI Readability score tests how easily text can be understood. It appears to slightly decrease with rating.

In [182]:
```
engineered_features.append('CLI_readability')
```

Automated Readability Index

In [183]:
```
df['ARI_readability'] = df.apply(
    lambda x: 4.71 * x.character_count / x.word_count + 0.5 * x.word_count / x.sentence_count
```

In [184]:
```
rating_plot('ARI_readability', 'ARI Readability')
```



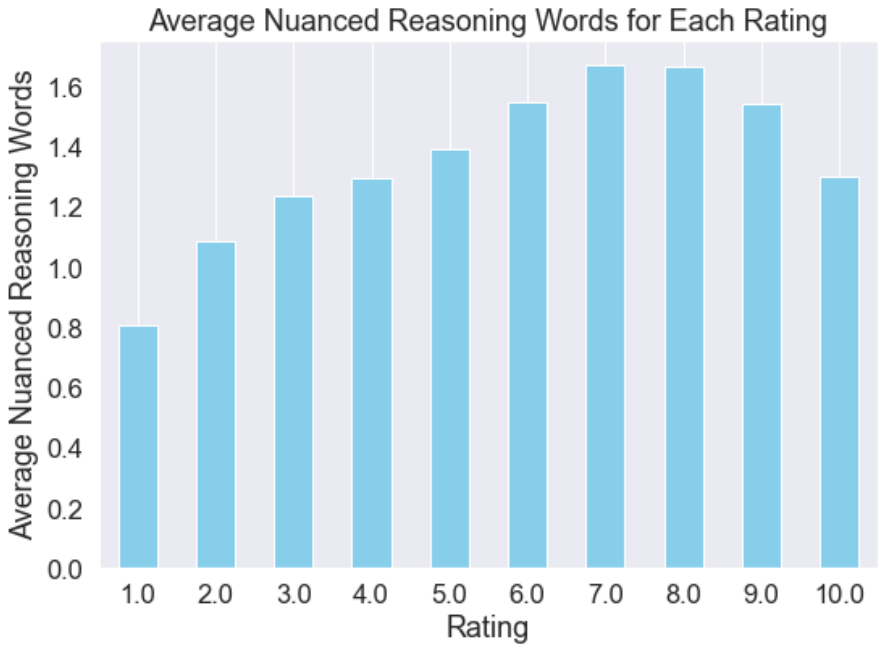
The ARI measures what US grade level is required to read a piece of text. This appears to increase through the middle rating values and then decrease. This suggests that more extreme ratings (which are more prevalent) tend to be written at a lower level of sophistication.

In [185]:
```
engineered_features.append('ARI_readability')
```

Words that indicate even-handed reasoning

In [186]:
```
# List of words to search for in reviews
nuanced_words = ['but', 'even though', 'conversely', 'nonetheless', 'notwithstanding', 'in sp
                 'however', 'although', 'yet', 'nevertheless', 'on the other hand', 'despite'

# Function to count occurrences of nuanced words in a review
def count_nuanced_words(text):
    text_lower = text.lower()
    count = sum(text_lower.count(word) for word in nuanced_words)
    return count

# Assuming df is your DataFrame and 'review_text' is the column containing the reviews
df['nuanced_words_count'] = df['review_lower'].apply(count_nuanced_words)
```

In [187]:
```
rating_plot('nuanced_words_count', 'Nuanced Reasoning Words')
```



In [188]:
```
df.rating.median()
```

Out[188]:
```
7.0
```

Here we've gathered several words that indicate the reviewer is grappling with contrasts of some nature. This appears to be most prevalent near the median of the rating values.

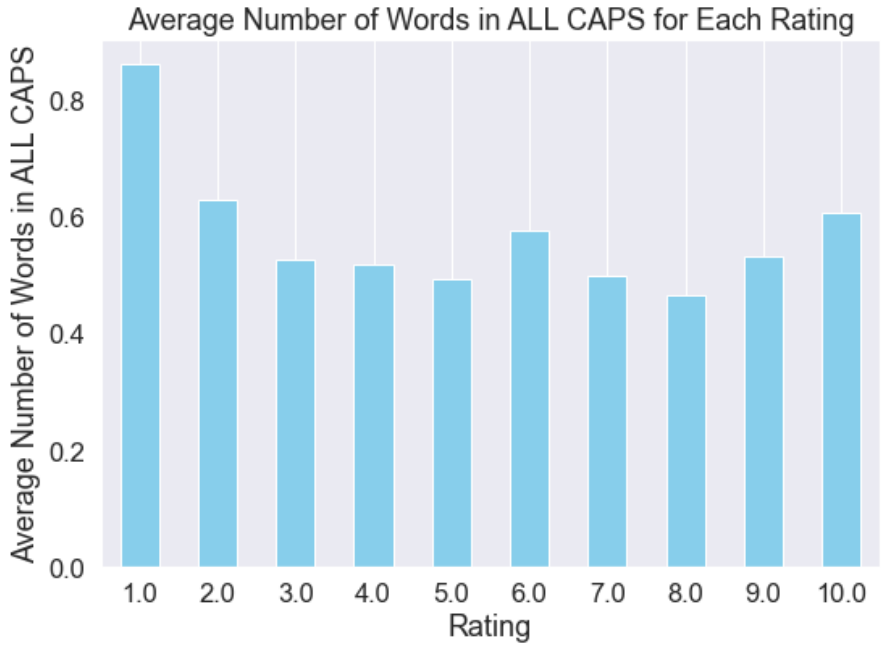
We may be seeing evidence that 7.0 is closer to "neutral" than 5.0, which could make our modeling much easier! If we split the data into positive and negative sentiments at 7.0 instead of 5.0, our model would have a much smaller class imbalance.

In [189]:
```
engineered_features.append('nuanced_words_count')
```

Words in all-caps

In [190]:
```
def all_caps_fix(review):
    review_tokenized = [word for word in nltk.word_tokenize(review) if word not in punctuatio
    all_caps_words = [word for word in review_tokenized if len(word) > 2 and word.isupper()]
    return len(all_caps_words)

df['all_caps_word_count'] = df.review.apply(lambda x: all_caps_fix(x))
```

In [191]:
```
rating_plot('all_caps_word_count', 'Number of Words in ALL CAPS')
```



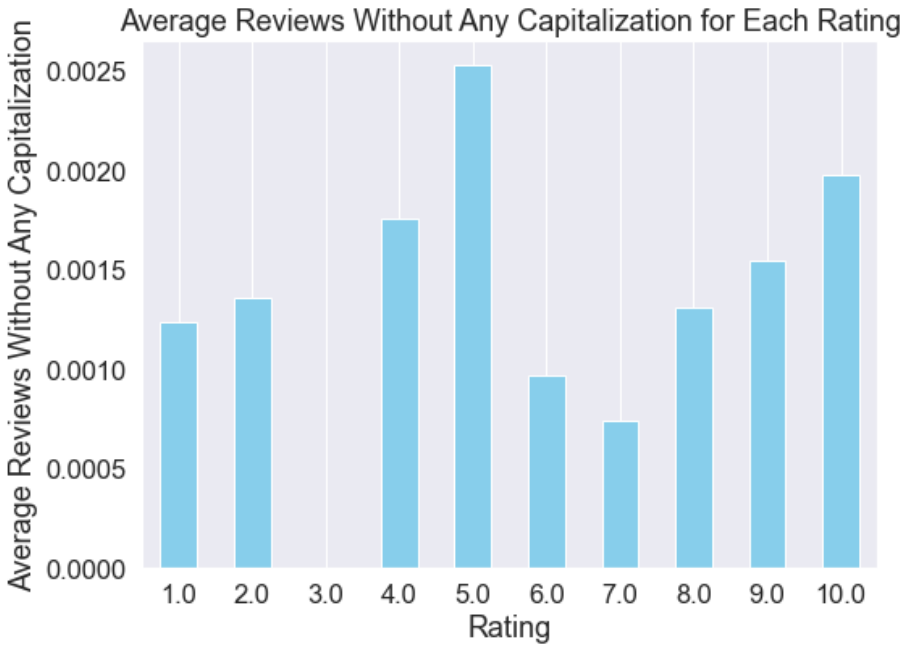
The greatests instances of words written in all capital letters takes place in reviews that accompany a 1.0 rating. All-caps writing is consistent with emotional emphasis and often frustration and anger. The relative peak near 6.0 is curious and not easy to explain.

In [192]:
```
engineered_features.append('all_caps_word_count')
```

Lack of capitalization

In [193]:
```
df['no_caps'] = df.apply(lambda x: 1 if x.review == x.review_lower else 0, axis=1)
```

In [194]:
```
rating_plot('no_caps', 'Reviews Without Any Capitalization')
```



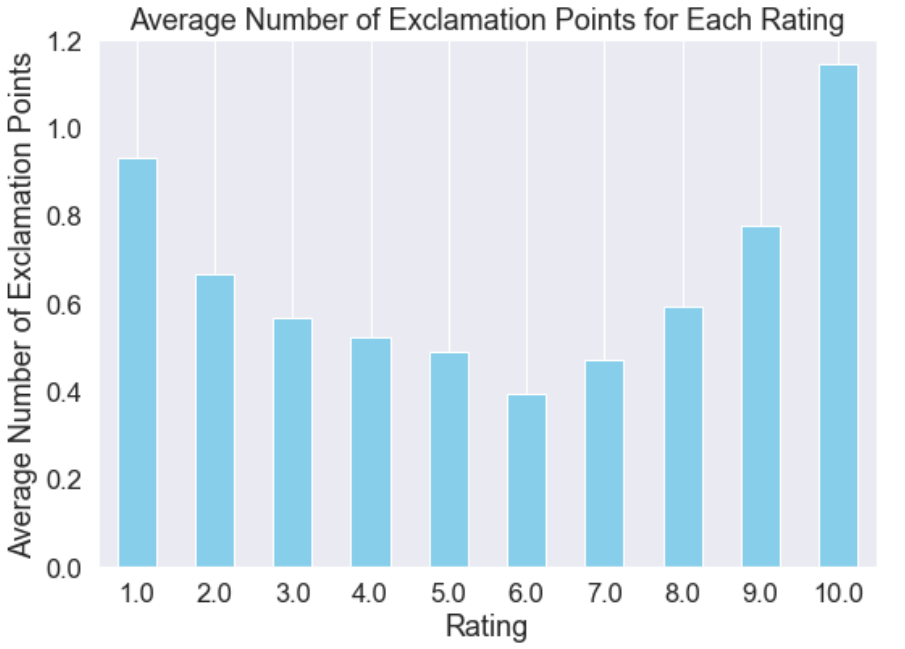
Writing without any capitalization at all is certainly becoming more common even among very well-educated writers, but can still indicate a lack of sophistication. Here we don't see much a trend at all.

In [195]:
```
engineered_features.append('no_caps')
```

Exclamation count

In [196]:
```
df['exclaim_count'] = df.review.apply(lambda x: x.count('!'))
```

In [197]:
```
rating_plot('exclaim_count', 'Number of Exclamation Points')
```



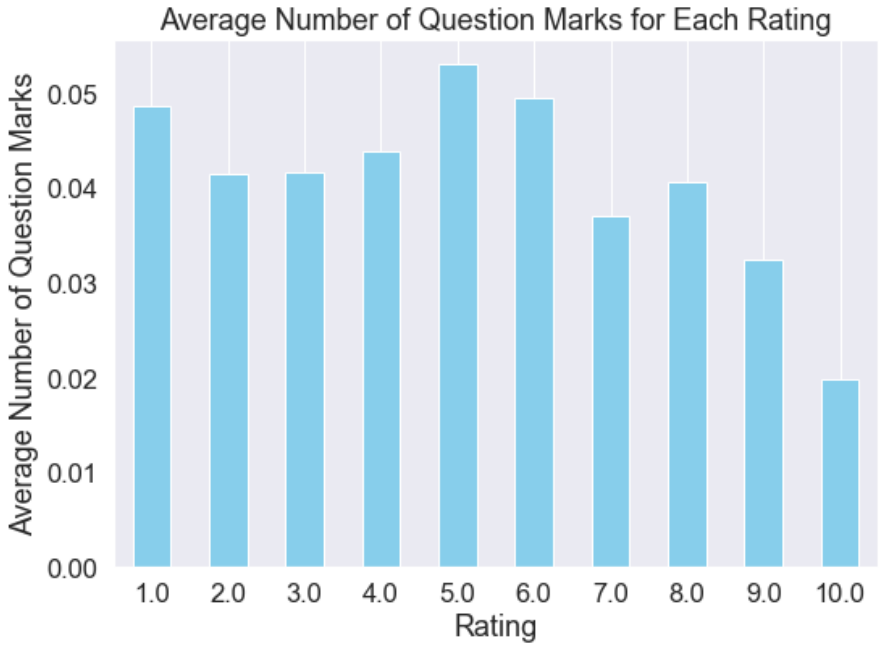
Exclamation points are, not surprisingly, clearly much more prevalent near the extreme ratings.

In [198]:
```
engineered_features.append('exclaim_count')
```

Question count

In [199]:
```
df['question_count'] = df.review.apply(lambda x: x.count('?'))
```

In [200]:
```
rating_plot('question_count', 'Number of Question Marks')
```



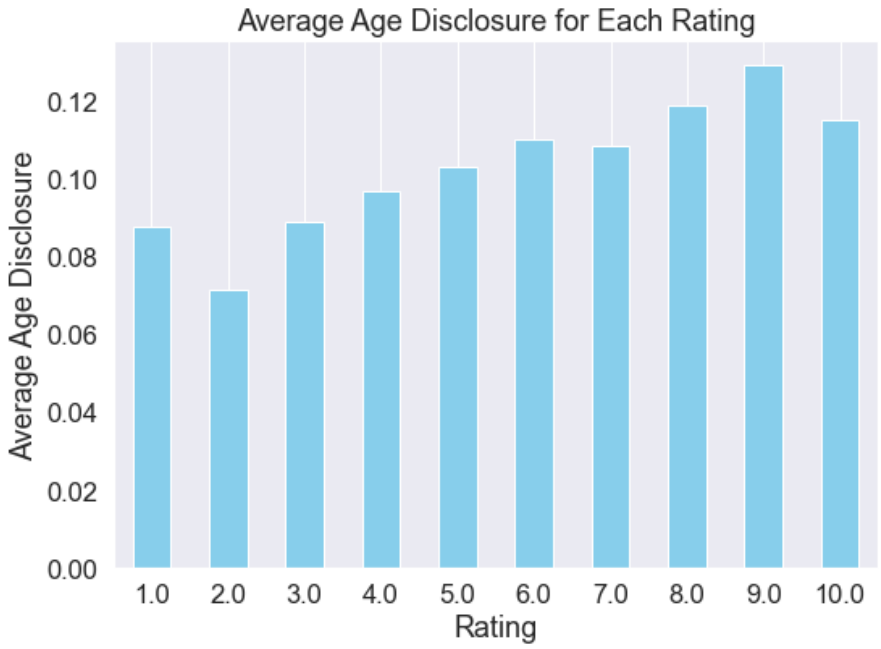
It's not entirely clear what the use of question marks might signify to us. The trend here seems somewhat to be that question mark use declines with rating.

In [201]:
```
engineered_features.append('question_count')
```

Age disclosure

In [202]:
```
def age_disclosure_feature(review):
    if re.search('[1-4][0-9] +y(ea)?rs? +(old|of age)', review) or \
    re.search('(pregnant( +at)?|age( +of)?|being) +[1-4][0-9]', review) or \
    re.search('(teen|ty|one|two|three|four|five|six|seven|eight|nine) +y(ea)?rs? +(old|of +age
    re.search('i +(am +|was +|will +be +|would have been +)(not even +)?(now +)?(only +)?(just
        return True
    else:
        return False

df['age_disclosure'] = df.review_lower.apply(lambda x: 1 if age_disclosure_feature(x) else 0)
```

In [203]:
```
rating_plot('age_disclosure', 'Age Disclosure')
```



We used the Regex expressions in the code above to attempt to identify which reviewers are disclosing their age in the text of the review. It is more likely that this somehow affects the `usefulCount` feature (number of upvotes), but this does not currently factor into our analysis.

In [204]:
```
engineered_features.append('age_disclosure')
```

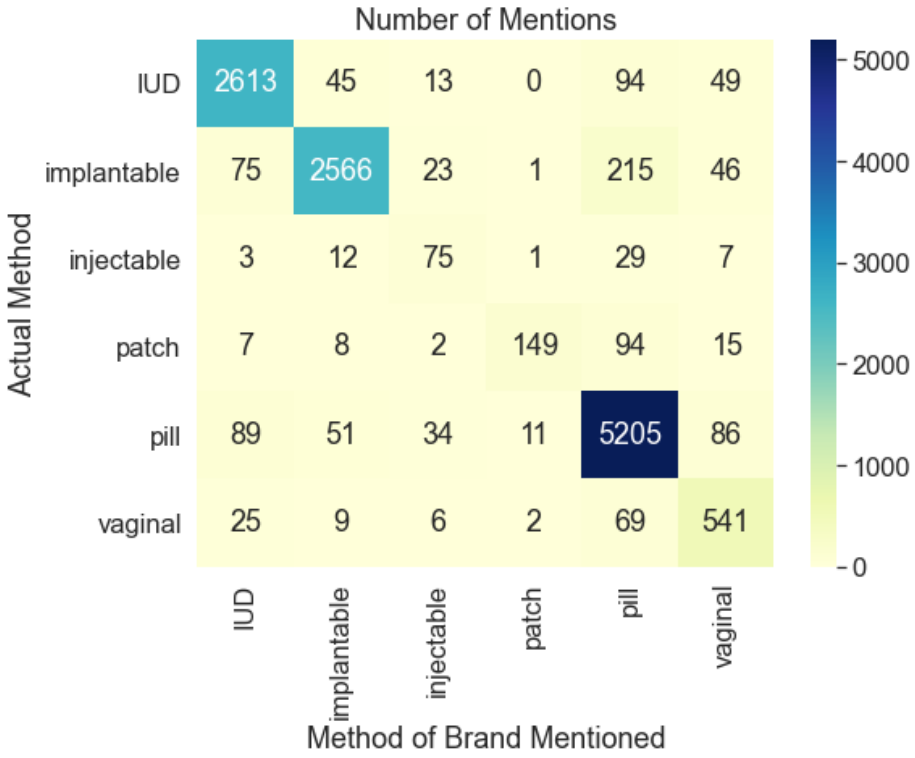
Inclusion of brand name

Since we've extracted a list of so many birth control brand names, we can look at whether the reviews actually mention any of those brand names and how much that correlates with the actual method corresponding to those brand names.

We'll be careful to exclude brand names like "my way" and "next choice" that are likely to appear often in discussions of other methods.

In [205]:
```
method_brand_dict = {}

def brand_by_method_fix(review, method_brand_list):
    for brand in method_brand_list:
        if brand == 'plan':
            continue
        if brand in review:
            return 1
    return 0

for method_ in df.method.unique():
    if method_ == 'emergency':
        continue
    elif method_ == 'patch':
        method_brand_list = ['Evra', 'Xulane']
    else:
        method_brand_list = list(df[(df.method == method_) & \
                                    (df.shortBrandName.notna())].shortBrandName.unique())
    method_brand_string = ' '.join([word.lower() for word in method_brand_list])
    method_brand_list = list(set(method_brand_string.split(' ')))
    df[method_] = df.review_lower.apply(lambda x: brand_by_method_fix(x, method_brand_list))
    engineered_features.append(method_)
```

In [206]:
```
# Create a pivot table to count occurrences of 1s for each method and dwarf
pivot_table = df[df.method != 'emergency'][['method', 'pill', 'patch', 'IUD', 'implantable',

# Create a heatmap visualization
plt.figure(figsize=(8, 6))
sns.heatmap(pivot_table, annot=True, cmap='YlGnBu', fmt='g')
plt.xlabel('Method of Brand Mentioned')
plt.ylabel('Actual Method')
plt.title('Number of Mentions')
plt.show()
```



In [207]:
```
len(df[df.method == 'IUD'])
```

Out[207]:
```
3701
```

The table above indeed shows that mentioning certain brand names is a strong indicator of which birth control method is being discussed in the review. For example, 3,701 of the reviews were of IUD drugs. In the top row of this table, we see that IUD drug brand names were mentioned in 2,613 of these reviews, while drug brand names for the pill were mentioned (possibly in comparison) only 94 times.

In [208]:
```
# DO NOT re-run this cell out of sequence
# to use the dataframe as it was at this stage, un-comment, run, and re-comment the cell that
df_bookmark_5 = df.copy()
```

In [209]:
```
# df = df_bookmark_5.copy()
```

Modeling

`do_grids`

Setting `do_grids = True` tells the notebook to run the grid searches that tune the various models, which can take many hours to finish.

In [210]:
```
do_grids = False
```

Method prediction model

First we'll create a model that predicts which birth control method is being discussed in a review.

In [211]:
```
df['target'] = df.method
```

In [212]:
```
X_train, X_test, y_train, y_test = \
train_test_split(df[['review'] + engineered_features], df['target'], test_size=0.2, random_sta
```

In [213]:
```
# save this value to compare to future model crossval scores
plurality_cv = round(y_train.value_counts(normalize=True)[1],4)
# show the sentiment breakdown
round(y_train.value_counts(normalize=True),4)
```

Out[213]:
```
target
pill           0.4524
implantable    0.2040
IUD            0.1706
emergency      0.0794
vaginal        0.0387
injectable     0.0316
patch          0.0234
Name: proportion, dtype: float64
```

Metric = F1-macro

The F1 score balances precision and recall. The F1-macro is a way of applying this to multiple classes. While our classes are rather imbalanced, the F1-macro calculates a simple average of each F1 score, giving equal weight to the smaller classes.

Preprocess data

In [214]:
```
# reset variables

text_preprocessor = None
numerical_preprocessor = None
preprocessor = None
pipeline = None
accuracy = None
feature_names = None
coefficients = None
decision_function_values = None
importance_df = None
feature_importance = None
```

In [215]:
```
max_features = None
stop_words = stop_list_light
ngram_range = (1,3)
```

In [216]:
```
text_preprocessor = TfidfVectorizer(
    max_features=max_features,
    ngram_range=ngram_range
)

numerical_preprocessor = StandardScaler()

preprocessor = ColumnTransformer(
    transformers=[
        ('text', text_preprocessor, 'review'),
        ('numerical', numerical_preprocessor, engineered_features)
    ]
)
```

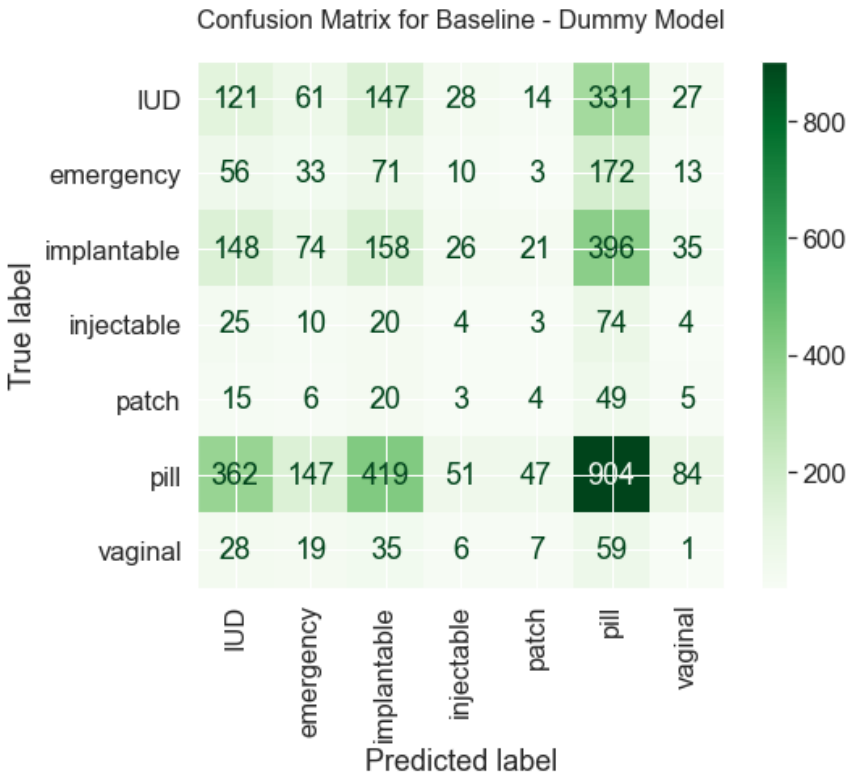
In [217]:
```
def run_model_1(model, title):
    pipeline = Pipeline([
        ('preprocessor', preprocessor),
        ('model', model)
    ])

    pipeline.fit(X_train, y_train)
    # generate predictions for the test data

    y_pred = pipeline.predict(X_test)
    # display the training and test accuracy scores
    print(f"Training Score: {round(pipeline.score(X_train, y_train),4)} \
    \nTest Score:     {round(pipeline.score(X_test, y_test),4)}")

    # generate predictions for the test data
    y_pred = pipeline.predict(X_test)

    # calculate different evaluation metrics
    accuracy = accuracy_score(y_test, y_pred)
    f1_macro = f1_score(y_test, y_pred, average='macro')


    # display different evaluation metrics
    print(f"\nAccuracy Score: {round(accuracy, 4)}")
    print(f"F1 Macro Score: {round(f1_macro, 4)}")


    # plot the confusion matrix
    cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
    fig, ax = plt.subplots(figsize=(8, 6))
    disp.plot(cmap='Greens', ax=ax)
    plt.title(f"Confusion Matrix for {title} Model", fontsize=16, pad=20)
    plt.xticks(rotation=90)
    plt.show()
```

In [218]:
```
def tune_model_1(model, param_grid):
    pipeline = Pipeline([('preprocessor', preprocessor), ('model', model)])
    param_grid = param_grid
    gridsearch = GridSearchCV(estimator=pipeline, param_grid = param_grid, cv=5, scoring='f1_
    gridsearch.fit(X_train,  y_train)
    gridsearch.best_params_
    return gridsearch.best_params_
```

Baseline – dummy classifier

In [219]:
```
%%time

run_model_1(DummyClassifier(strategy='stratified'), 'Baseline - Dummy')
```

```
Training Score: 0.2858
Test Score:     0.2879

Accuracy Score: 0.2812
F1 Macro Score: 0.1382
```

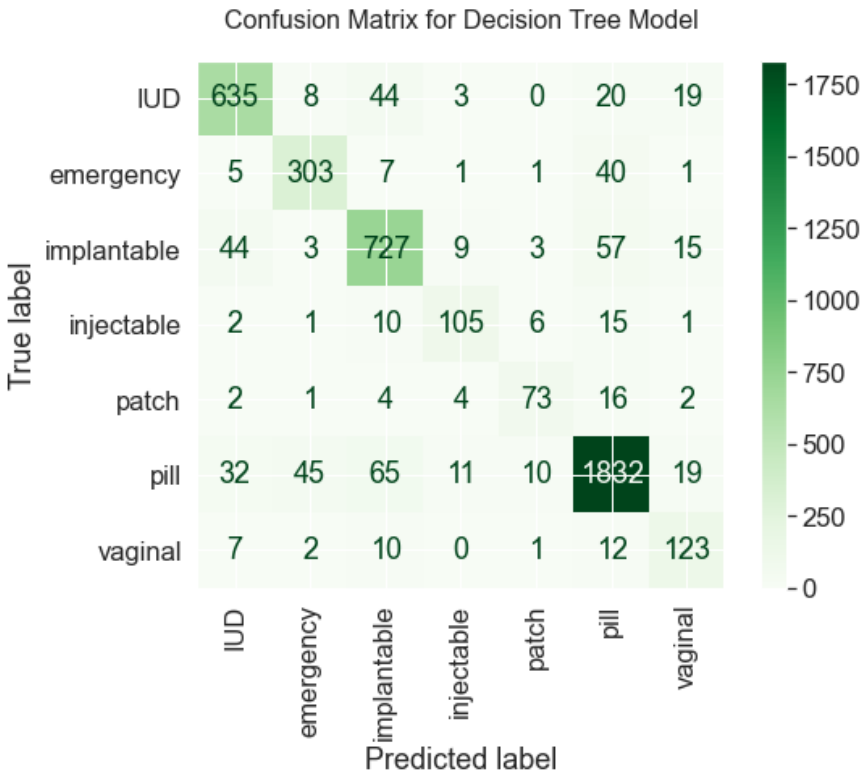



```
CPU times: user 16.6 s, sys: 893 ms, total: 17.5 s
Wall time: 17.3 s
```

Decision tree

In [220]:
```
%%time

run_model_1(DecisionTreeClassifier(random_state=SEED), 'Decision Tree')
```

```
Training Score: 0.9999
Test Score:     0.8719

Accuracy Score: 0.8719
F1 Macro Score: 0.817
```




```
CPU times: user 1min 19s, sys: 990 ms, total: 1min 20s
Wall time: 1min 20s
```

The baseline model performs just fine. Predictably, it frequently mistakes other methods for the pill, although surprisingly this happens relatively less often for reviews of emergency contraceptives, which are technically also pills.

We see a little confusion between IUDs and implantables, likely due to the language of "inserting" these methods, etc.

Importantly, even the recall of the smallest class, the patch method, is rather good (72%).

As the training accuracy for this metric is nearly 100%, it is also clearly rather overfit.

Tuning the decision tree model

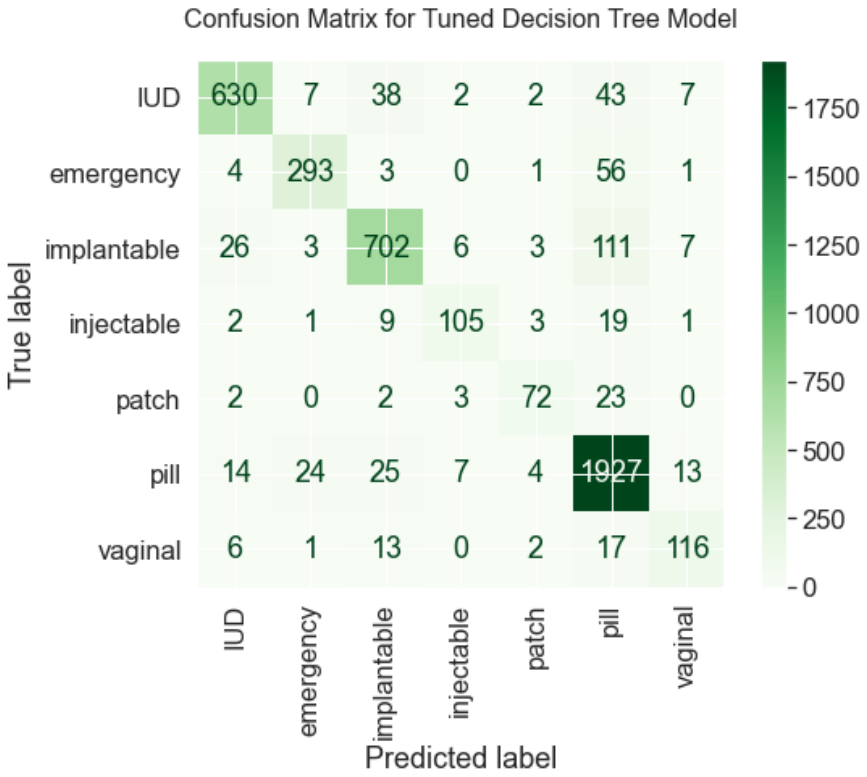
In [221]:
```
%%time

if do_grids == True:
    best_params_ = tune_model_1(DecisionTreeClassifier(random_state=SEED), param_grid={
        'model__criterion': ['gini', 'entropy'],
        'model__max_depth': [10, 20, None],
        'model__min_samples_leaf': [1, 2, 3]
    })
else:
    best_params_ = "{'model__criterion': 'gini', 'model__max_depth': 20, 'model__min_samples_
print(best_params_)
```

```
{'model__criterion': 'gini', 'model__max_depth': 20, 'model__min_samples_leaf': 1}
CPU times: user 227 µs, sys: 100 µs, total: 327 µs
Wall time: 284 µs
```

Decision tree — tuned

In [222]:
```
%%time

run_model_1(DecisionTreeClassifier(
    random_state=SEED,
    criterion='gini',
    max_depth=20,
    min_samples_leaf=1
),
           'Tuned Decision Tree'
           )
```

```
Training Score: 0.9483
Test Score:     0.8827

Accuracy Score: 0.8827
F1 Macro Score: 0.835
```



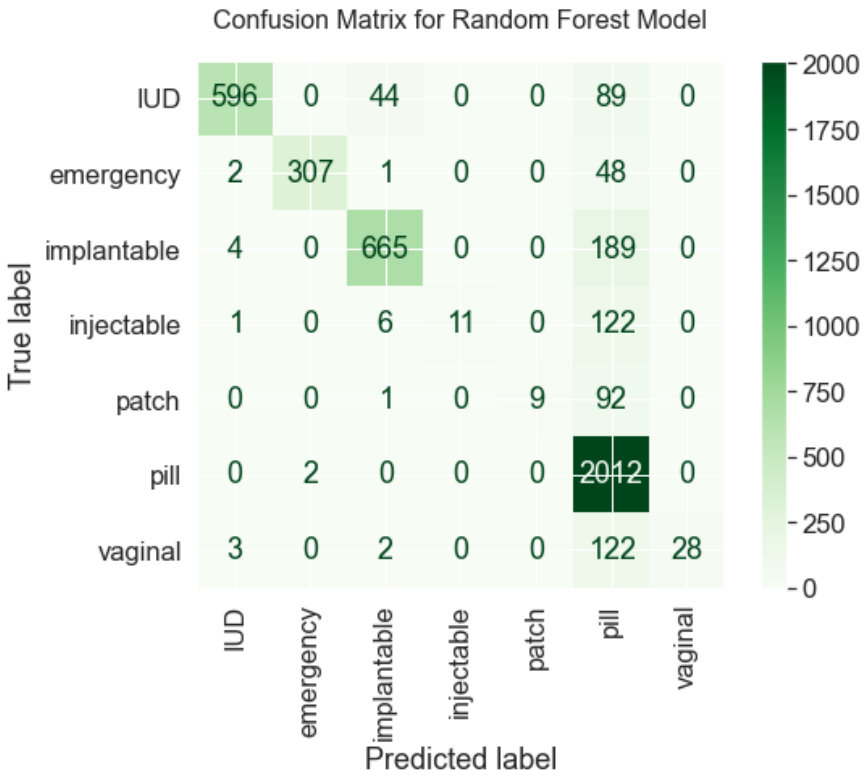

```
CPU times: user 55.7 s, sys: 851 ms, total: 56.5 s
Wall time: 56.4 s
```

These results are barely distinguishable from the baseline.

Random forest

In [223]:
```
%%time

run_model_1(RandomForestClassifier(random_state=SEED), 'Random Forest')
```

```
Training Score: 0.9999
Test Score:     0.8329

Accuracy Score: 0.8329
F1 Macro Score: 0.5899
```




```
CPU times: user 4min 20s, sys: 1.93 s, total: 4min 22s
Wall time: 4min 23s
```

The random forest model also overfits the model, yielding results that are certainly worse than the baseline model.

Tuning the random forest model

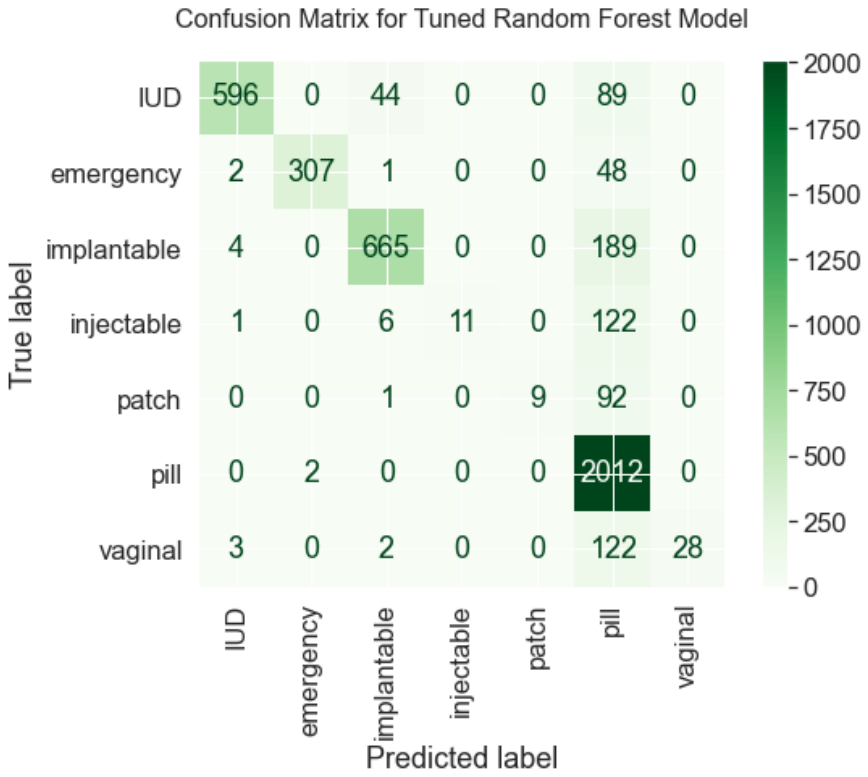
In [224]:
```
%%time

if do_grids == True:
    best_params_ = tune_model_1(RandomForestClassifier(random_state=SEED), param_grid={
        'model__criterion': ['gini', 'entropy'],
        'model__max_depth': [10, 20, None],
        'model__min_samples_leaf': [1, 2, 3]
    })
else:
    best_params_ = "{'model__criterion': 'gini', 'model__max_depth': None, 'model__min_samples
print(best_params_)
```

```
{'model__criterion': 'gini', 'model__max_depth': None, 'model__min_samples_leaf': 1}
CPU times: user 163 µs, sys: 73 µs, total: 236 µs
Wall time: 211 µs
```

Random forest — tuned

In [225]:
```
%%time

run_model_1(RandomForestClassifier(
    random_state=SEED,
    criterion='gini',
    max_depth=None,
    min_samples_leaf=1
),
           'Tuned Random Forest'
           )
```

```
Training Score: 0.9999
Test Score:     0.8329

Accuracy Score: 0.8329
F1 Macro Score: 0.5899
```



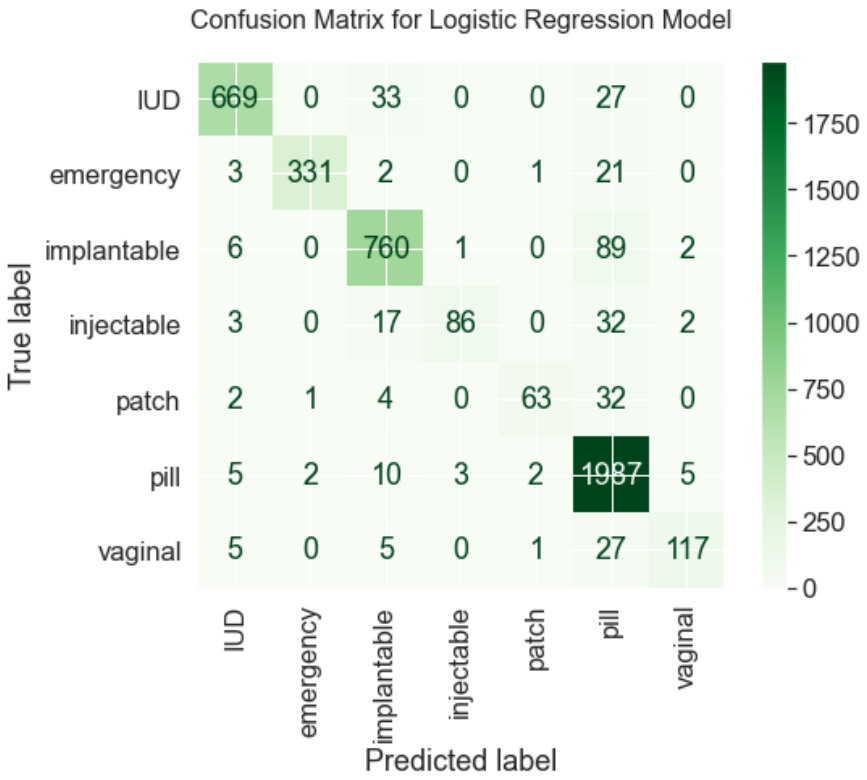

```
CPU times: user 4min 19s, sys: 1.77 s, total: 4min 20s
Wall time: 4min 21s
```

Tuning this model didn't improve much. It is still worse than the baseline decision tree model.

Logistic regression

In [226]:
```
%%time

run_model_1(LogisticRegression(random_state=SEED, max_iter=1000), 'Logistic Regression')
```

```
Training Score: 0.9504
Test Score:     0.9213

Accuracy Score: 0.9213
F1 Macro Score: 0.8662
```




```
CPU times: user 10min 35s, sys: 2min 15s, total: 12min 51s
Wall time: 5min 32s
```

This is much better than the baseline. The F1-macro score is higher (87%) and the accuracy is higher (92%). Notably, the accuracy of the smallest class is lower (62%).

Tuning the logistic regression model

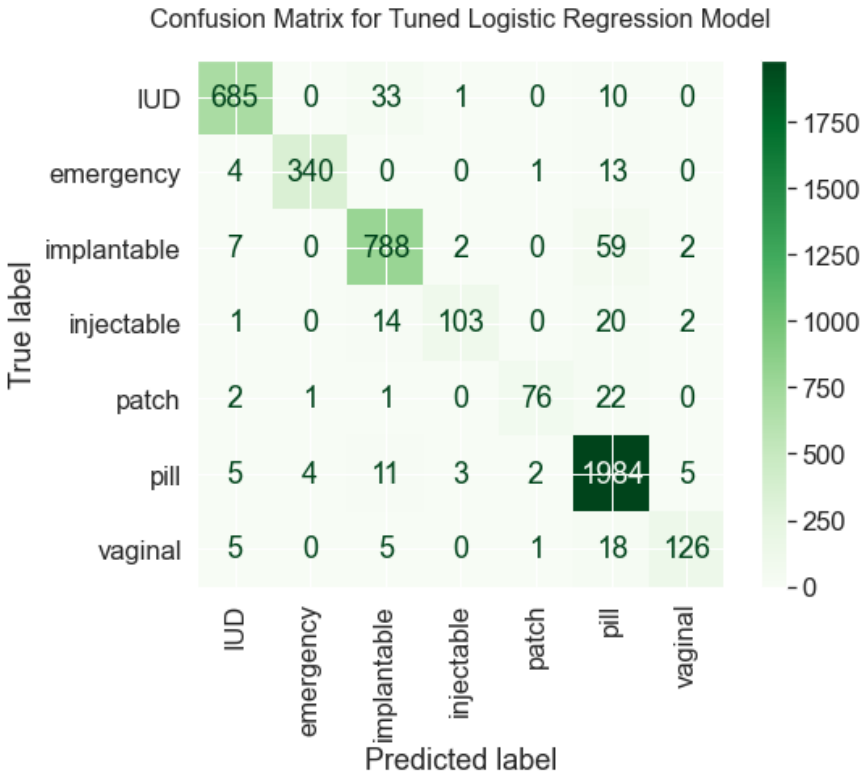
In [227]:
```
%%time

if do_grids == True:
    best_params_ = tune_model_1(LogisticRegression(random_state=SEED, max_iter=1000), param_g
        'model__C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization parameter
        'model__penalty': ['l1', 'l2'],  # Regularization penalty ('l1' for Lasso, 'l2' for R
        'model__solver': ['liblinear', 'saga']  # Algorithm to use in the optimization proble
    })
else:
    best_params_ = "{'model__C': 10, 'model__penalty': 'l2', 'model__solver': 'liblinear'}"
print(best_params_)
```

```
{'model__C': 10, 'model__penalty': 'l2', 'model__solver': 'liblinear'}
CPU times: user 236 µs, sys: 133 µs, total: 369 µs
Wall time: 293 µs
```

Logistic regression — tuned

In [228]:
```
%%time

run_model_1(LogisticRegression(
    random_state=SEED,
    max_iter=1000,
    C=10,
    penalty='l2',
    solver='liblinear'
),
           'Tuned Logistic Regression'
           )
```

```
Training Score: 0.9985
Test Score:     0.9417

Accuracy Score: 0.9417
F1 Macro Score: 0.9045
```




```
CPU times: user 4min 22s, sys: 1min 16s, total: 5min 39s
Wall time: 1min 39s
```

The F1-macro (0.91) and accuracy (94%) metrics are the highest yet. Also, the accuracy of the smallest class (patch) (75%) is also higher than in the baseline.

Method prediction model evaluation

Our final model is the tuned logistic regression model. It returns the highest F1-macro score (0.91), which was our primary scoring metric. The F1-macro score averages F1 scores for all classes, unweighted. A weighted score might have forsaken accuracy on smaller classes (e.g. patch) in favor of scoring better on larger classes, but we wanted our model to have a more even distribution. Its overall accuracy (94%) is very impressive. The confusion matrix shows the most erroneous labels (predictably) occurring in the largest class (pill); the only other somewhat significant confusion (<5%) occurs when IUDs are confused with implantables.

Sentiment prediction model

In [229]:
```
rating_threshold = [2, 10]

print(len(df[df.rating <= rating_threshold[0]]), len(df[df.rating >= rating_threshold[1]]))
```

```
4698 5047
```

The above would be the easiest split. It uses fewer values and reduces class imbalance.

In [230]:
```
rating_threshold = [7, 8]

neg_count = len(df[df.rating <= rating_threshold[0]])
pos_count = len(df[df.rating >= rating_threshold[1]])

print(neg_count, pos_count)
```

```
11206 10573
```

The above would be the best way to reduce class imbalance while using all of the data.

In [231]:
```
rating_threshold = [4, 6]

neg_count = len(df[df.rating <= rating_threshold[0]])
pos_count = len(df[df.rating >= rating_threshold[1]])

print(neg_count, pos_count)
```

```
7249 12948
```

The above would use nearly all the data and give a better approximation of positive v. negative. It introduces a more significant class imbalance, though.

In [232]:
```
neg_freq = neg_count / (neg_count + pos_count)
pos_freq = pos_count / (neg_count + pos_count)
```

In [233]:
```
df.drop(df[
    (df.rating > rating_threshold[0]) & \
    (df.rating < rating_threshold[1])
].index, inplace=True)

df['target'] = df.rating.apply(lambda x: 1 if x <= rating_threshold[0] else 0)
```

In [234]:
```
X_train, X_test, y_train, y_test = \
train_test_split(df[['review'] + engineered_features], df['target'], test_size=0.2, random_sta
```

In [235]:
```
# save this value to compare to future model crossval scores
plurality_cv = round(y_train.value_counts(normalize=True)[1],4)
# show the sentiment breakdown
round(y_train.value_counts(normalize=True),4)
```

Out[235]:
```
target
0    0.6416
1    0.3584
Name: proportion, dtype: float64
```

Metric = ROC-AUC

The ROC-AUC metric depicts the tradeoff between the false positive rate and the true positive rate. We used this metric to tune our sentiment prediction models. We also closely observed overall accuracy, and recall of *negative* sentiment (coded "1"), as that constituted the minority class, which was more likely to be overlooked than the majority class.

Preprocess data

In [236]:
```
# reset variables

text_preprocessor = None
numerical_preprocessor = None
preprocessor = None
pipeline = None
accuracy = None
feature_names = None
coefficients = None
decision_function_values = None
importance_df = None
feature_importance = None
```

In [237]:
```
max_features = None
stop_words = stop_list_light
ngram_range = (1,3)
class_weight = {
    0: neg_freq,
    1: pos_freq/neg_freq
}
```

In [238]:
```
text_preprocessor = TfidfVectorizer(
    max_features=max_features,
    ngram_range=ngram_range
)

numerical_preprocessor = StandardScaler()

preprocessor = ColumnTransformer(
    transformers=[
        ('text', text_preprocessor, 'review'),
        ('numerical', numerical_preprocessor, engineered_features)
    ]
)
```

In [239]:
```
def run_model_2(model, title):
    pipeline = Pipeline([
        ('preprocessor', preprocessor),
        ('model', model)
    ])

    pipeline.fit(X_train, y_train)
    # generate predictions for the test data
    y_pred = pipeline.predict(X_test)
    # display the training and test accuracy scores
    print(f"Training Score: {round(pipeline.score(X_train, y_train),4)} \
    \nTest Score:     {round(pipeline.score(X_test, y_test),4)}")

    # generate predictions for the test data
    y_pred = pipeline.predict(X_test)
    y_pred_proba = pipeline.predict_proba(X_test)[:, 1]  # Probabilities for log loss

    # calculate different evaluation metrics
    accuracy = accuracy_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc_auc = roc_auc_score(y_test, y_pred_proba)

    # display different evaluation metrics
    print(f"\nAccuracy Score: {round(accuracy, 4)}")
    print(f"Recall Score: {round(recall, 4)}")
    print(f"F1 Score: {round(f1, 4)}")
    print(f"ROC-AUC Score: {round(roc_auc, 4)}")

    # plot the normalized confusion matrix
    cm = confusion_matrix(y_test, y_pred, labels=pipeline.classes_)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipeline.classes_)
    fig, ax = plt.subplots(figsize=(8, 6))  # Modify the figsize as per your preference
    disp.plot(cmap='Greens', ax=ax)
    plt.title(f"Confusion Matrix for {title} Model", fontsize=16, pad=20)
    plt.xticks(rotation=90)
    plt.show()



    # Calculate the False Positive Rate (FPR), True Positive Rate (TPR), and Thresholds
    fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)

    # Calculate the Area Under the ROC Curve (ROC-AUC)
    roc_auc = auc(fpr, tpr)

    # Plot the ROC curve
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='red', linestyle='--', lw=2, label='Random Guessing')
    plt.xlabel('False Positive Rate (FPR)')
    plt.ylabel('True Positive Rate (TPR)')
    plt.title(f"ROC Curve for {title} Model", fontsize=16, pad=20)
    plt.legend(loc='lower right')
    plt.show()


    feature_names = preprocessor.named_transformers_['text'].get_feature_names_out().tolist()

    n_features = 25

    # ------------------------------------------------------------
    if title == 'Baseline - Dummy':
        return
    if hasattr(model, 'feature_importances_'):
        coefficients = model.feature_importances_
    elif hasattr(model.coef_, 'toarray'):
        coefficients = model.coef_.toarray().flatten()
    elif hasattr(model, 'coef_'):
        coefficients = model.coef_.flatten()
    else:
```

```
        return
    # ----------------------------------------------------------

    importance_df = pd.DataFrame(feature_names, columns=['Word'])
    importance_df['Importance'] = np.e**(abs(coefficients))
    importance_df['Coefficient'] = coefficients

    feature_importance = importance_df.sort_values(
        by = ["Importance"], ascending=False
    ).head(n_features)

    fig, ax = plt.subplots(figsize=(15,10), ncols=2)
    ax[0].set_title(f'Coefficients for {title} Model')
    ax[0].set_ylabel('Word')
    ax[0].set_xlabel('Coefficient')
    sns.barplot(x='Coefficient', y='Word', data=feature_importance,
                palette='coolwarm', ax=ax[0])
    #plotting feature importances
    ax[1].set_title(f'Feature Importances for {title} Model')
    ax[1].set_ylabel('Word')
    ax[1].set_xlabel('Importance')
    sns.barplot(x='Importance', y='Word', data=feature_importance,
                palette='coolwarm', ax=ax[1])
    plt.tight_layout()
```

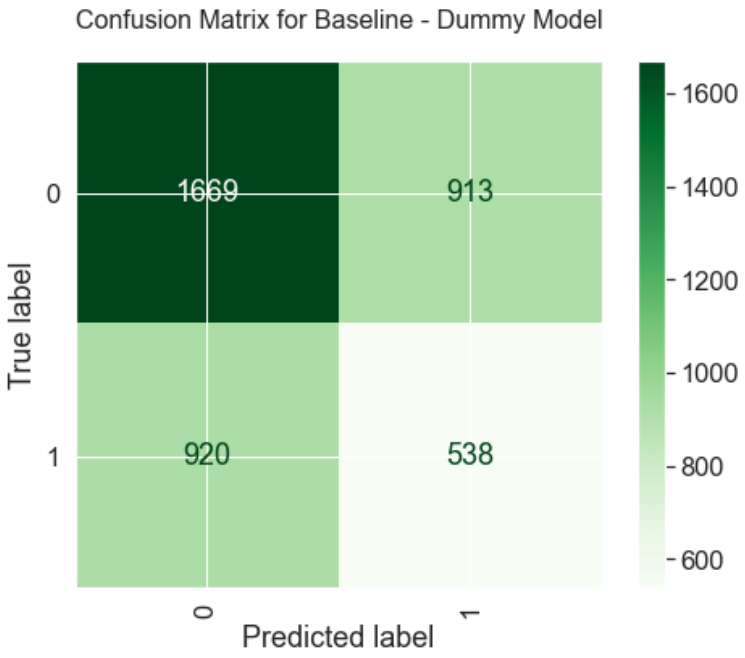
In [240]:

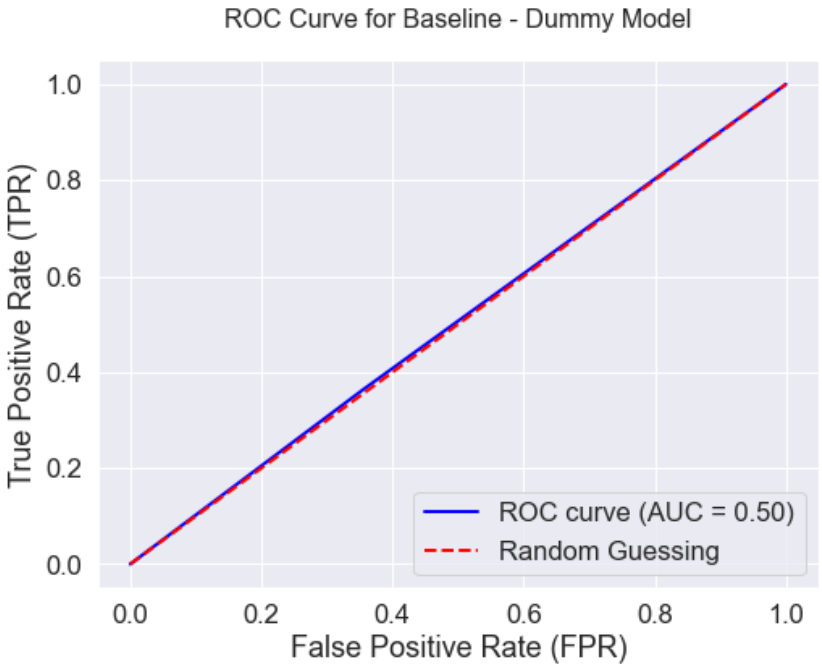
```
def tune_model_2(model, param_grid):
    pipeline = Pipeline([('preprocessor', preprocessor), ('model', model)])
    param_grid = param_grid
    gridsearch = GridSearchCV(
        estimator=pipeline,
        param_grid = param_grid,
        cv=5,
        scoring='roc_auc',
    )
    gridsearch.fit(X_train,  y_train)
    gridsearch.best_params_
    return gridsearch.best_params_
```

Baseline – dummy classifier

In [241]:
```
%%time

run_model_2(DummyClassifier(strategy='stratified'), 'Baseline - Dummy')
```

```
Training Score: 0.5368
Test Score:     0.5267

Accuracy Score: 0.5463
Recall Score: 0.369
F1 Score: 0.3699
ROC-AUC Score: 0.5043
```



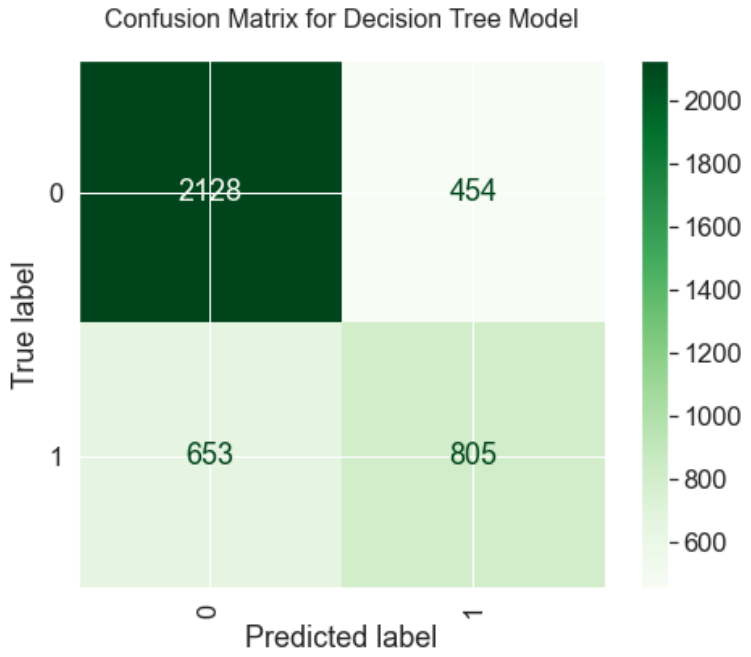


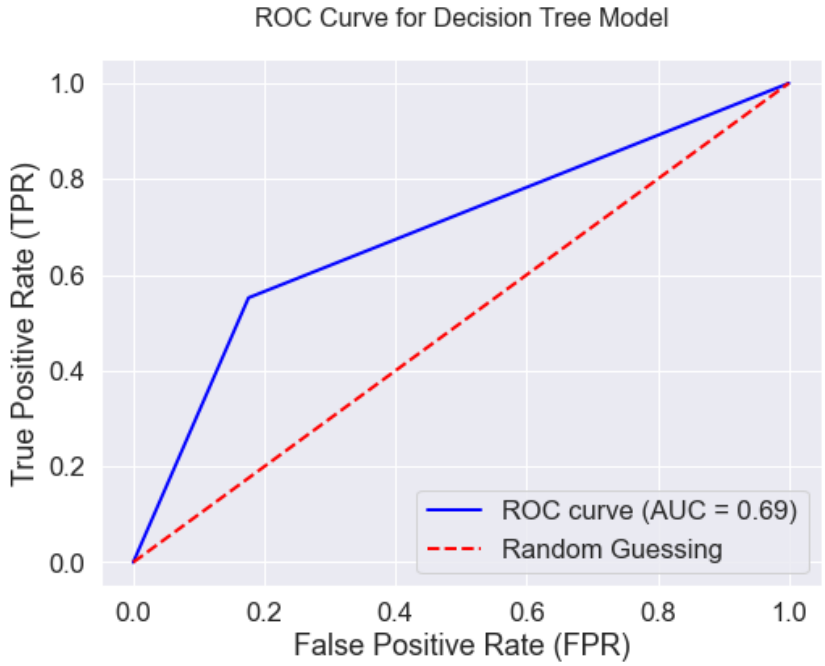

```
CPU times: user 18.1 s, sys: 711 ms, total: 18.8 s
Wall time: 18.4 s
```

Decision tree

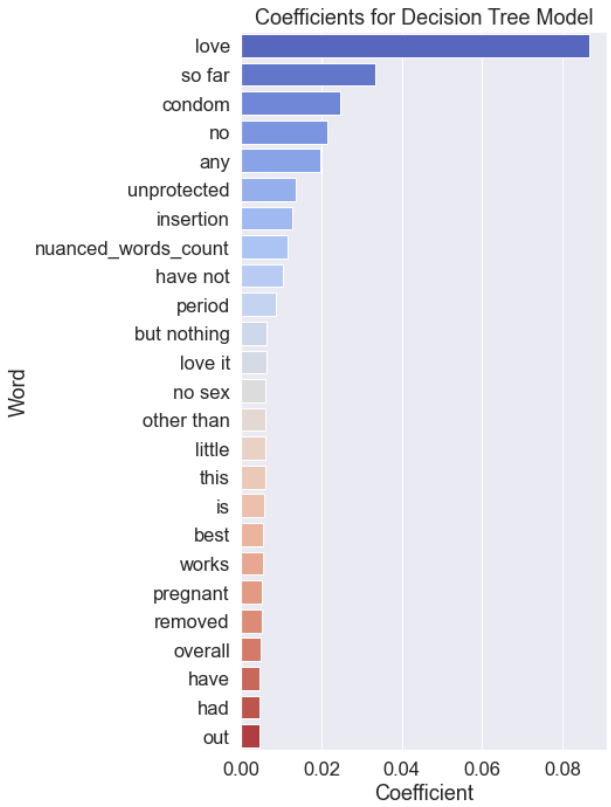
In [242]:
```
%%time

run_model_2(
    DecisionTreeClassifier(random_state=SEED, class_weight=class_weight),
    'Decision Tree'
)
```

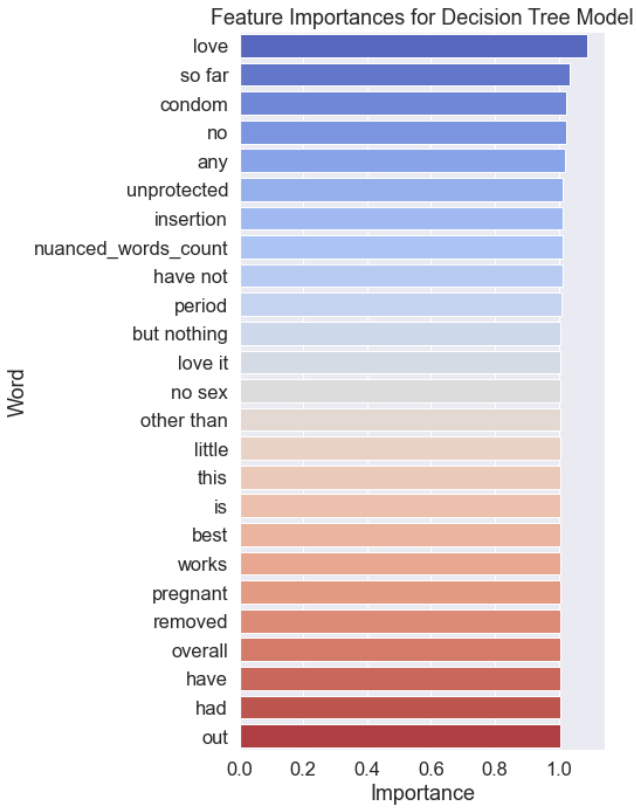
```
Training Score: 1.0
Test Score:     0.726

Accuracy Score: 0.726
Recall Score: 0.5521
F1 Score: 0.5926
ROC-AUC Score: 0.6881
```






```
CPU times: user 2min 27s, sys: 1.3 s, total: 2min 28s
Wall time: 2min 28s
```





Tuning the decision tree model

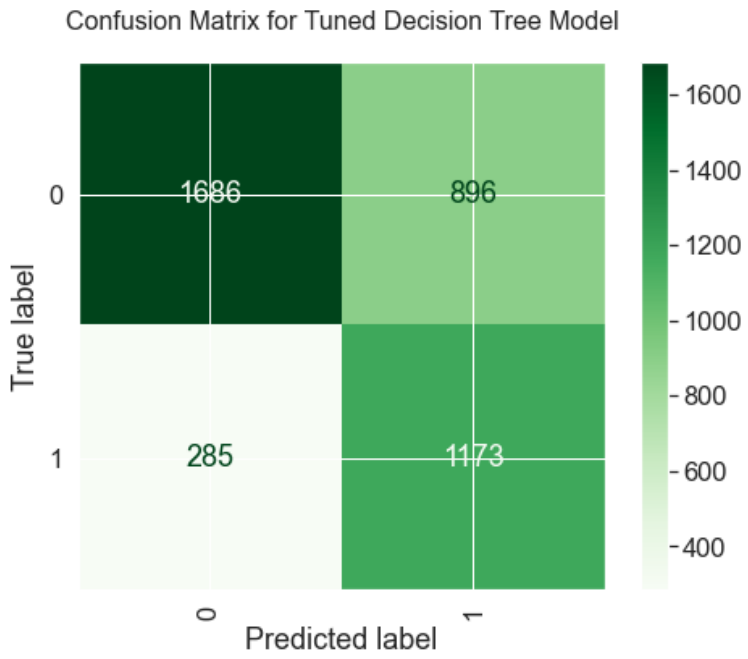
In [243]:
```
%%time

if do_grids == True:
    best_params_ = tune_model_2(DecisionTreeClassifier(random_state=SEED, class_weight=class_
        'model__criterion': ['gini', 'entropy'],
        'model__max_depth': [10, 20, None],
        'model__min_samples_leaf': [1, 2, 3]
    })
else:
    best_params_ = "{'model__criterion': 'gini', 'model__max_depth': 20, 'model__min_samples_
print(best_params_)
```

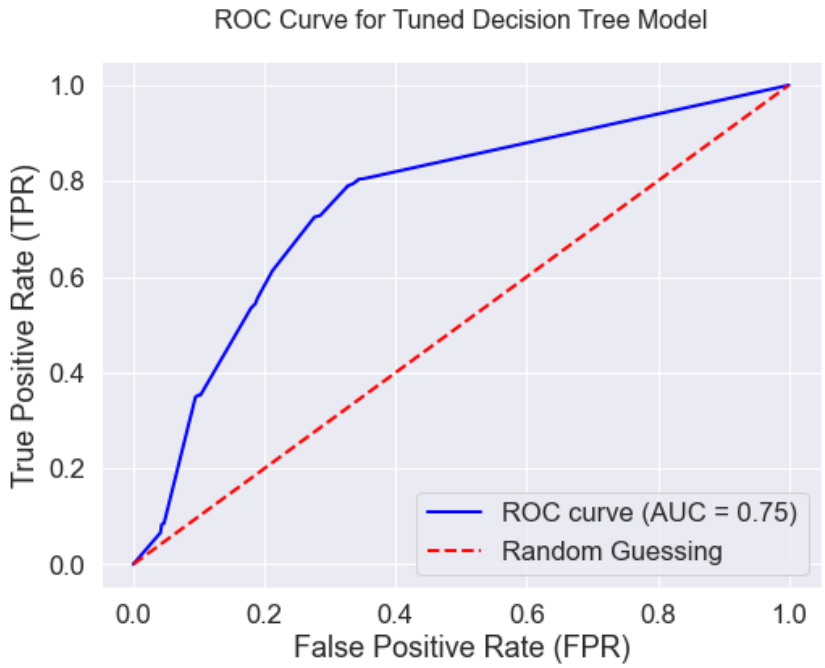
```
{'model__criterion': 'gini', 'model__max_depth': 20, 'model__min_samples_leaf': 2}
CPU times: user 304 µs, sys: 157 µs, total: 461 µs
Wall time: 372 µs
```

Decision tree — tuned

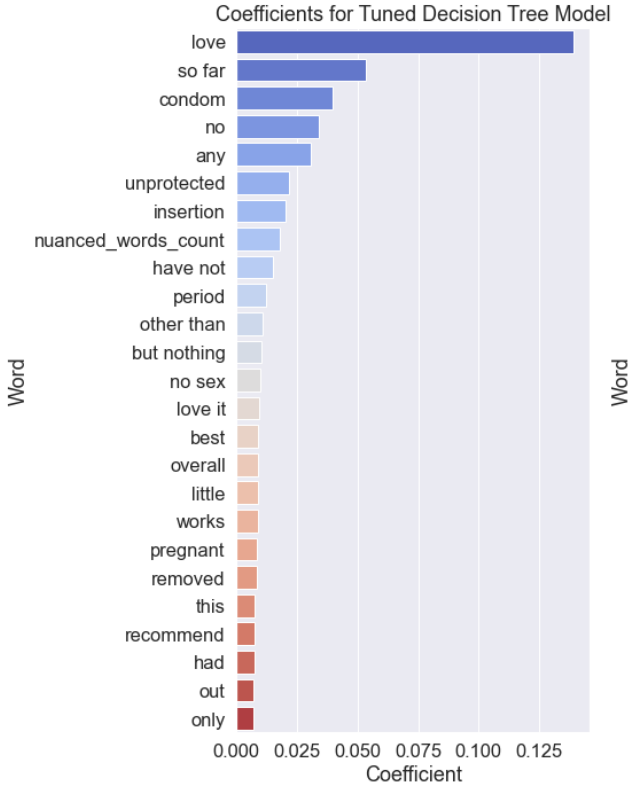
In [244]:
```
%%time

run_model_2(DecisionTreeClassifier(
    random_state=SEED,
    class_weight=class_weight,
    criterion='gini',
    max_depth=20,
    min_samples_leaf=2
),
           'Tuned Decision Tree'
           )
```

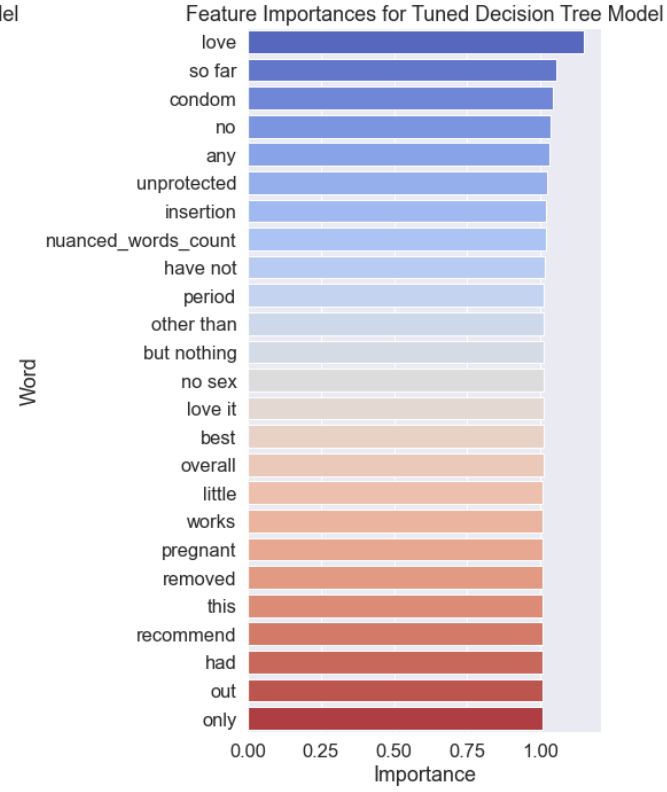
```
Training Score: 0.7904
Test Score:     0.7077

Accuracy Score: 0.7077
Recall Score: 0.8045
F1 Score: 0.6652
ROC-AUC Score: 0.7536
```






```
CPU times: user 49.6 s, sys: 1.08 s, total: 50.7 s
Wall time: 50.2 s
```

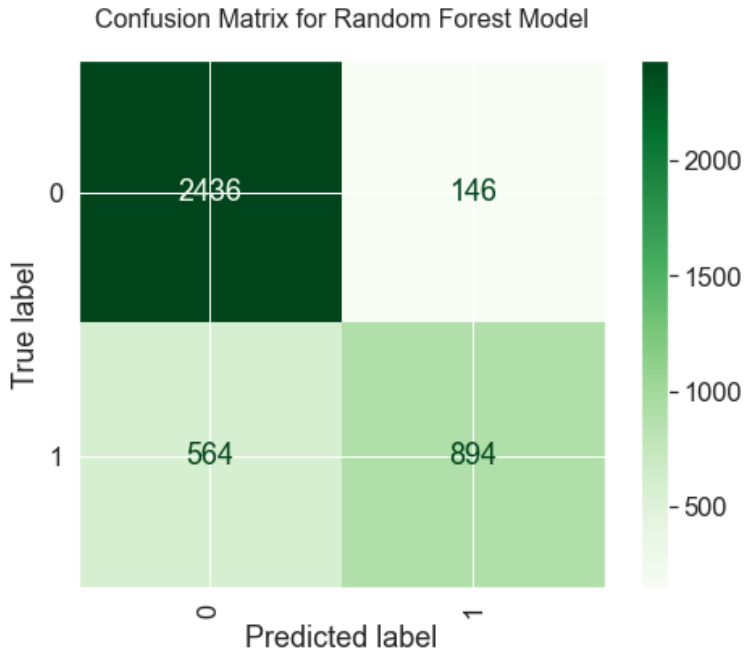


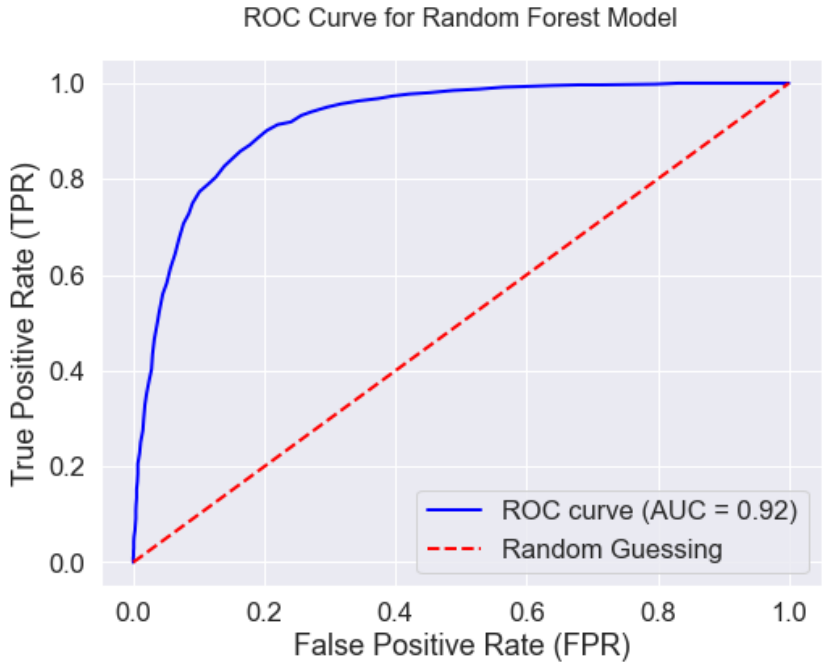


Random forest

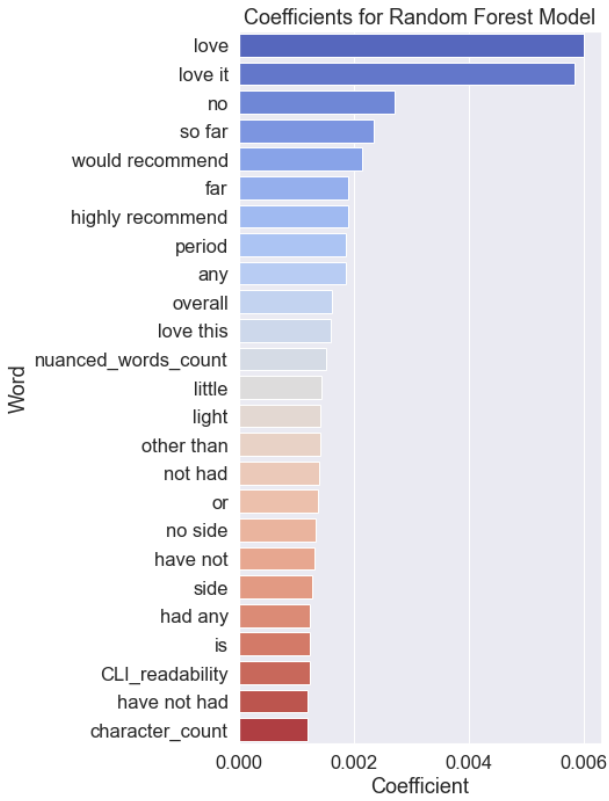
In [245]:
```
%%time

run_model_2(
    RandomForestClassifier(random_state=SEED, class_weight=class_weight),
    'Random Forest'
)
```

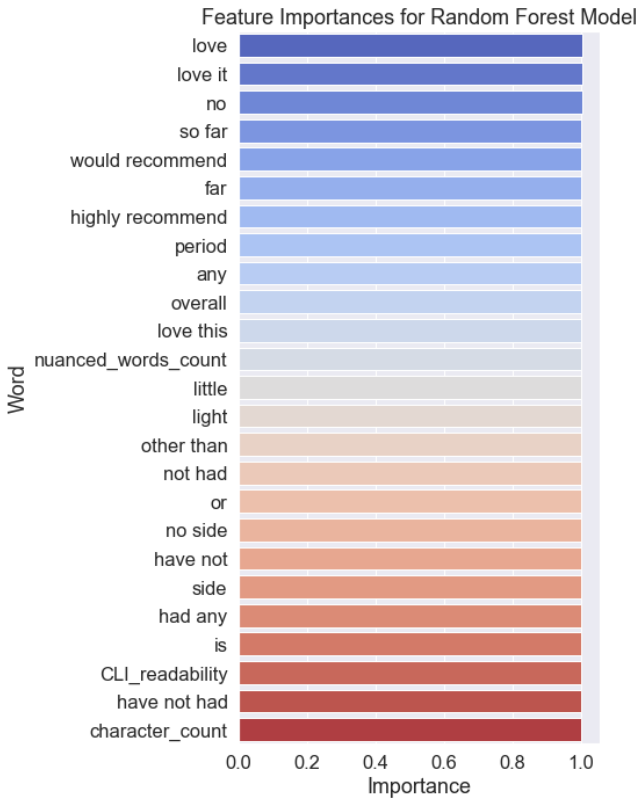
```
Training Score: 1.0
Test Score:     0.8243

Accuracy Score: 0.8243
Recall Score: 0.6132
F1 Score: 0.7158
ROC-AUC Score: 0.922
```






```
CPU times: user 3min 8s, sys: 2.95 s, total: 3min 11s
Wall time: 3min 12s
```





Tuning the random forest model

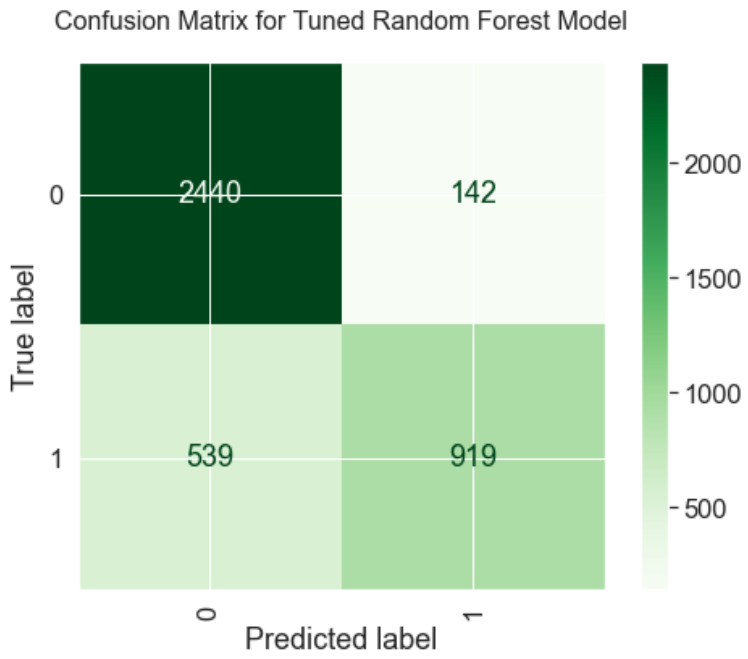
In [246]:
```
%%time

if do_grids == True:
    best_params_ = tune_model_2(RandomForestClassifier(random_state=SEED, class_weight=class_
        'model__criterion': ['gini', 'entropy'],
        'model__max_depth': [10, 20, None],
        'model__min_samples_leaf': [1, 2, 3]
    })
else:
    best_params_ = "{'model__criterion': 'entropy', 'model__max_depth': None, 'model__min_samp
print(best_params_)
```

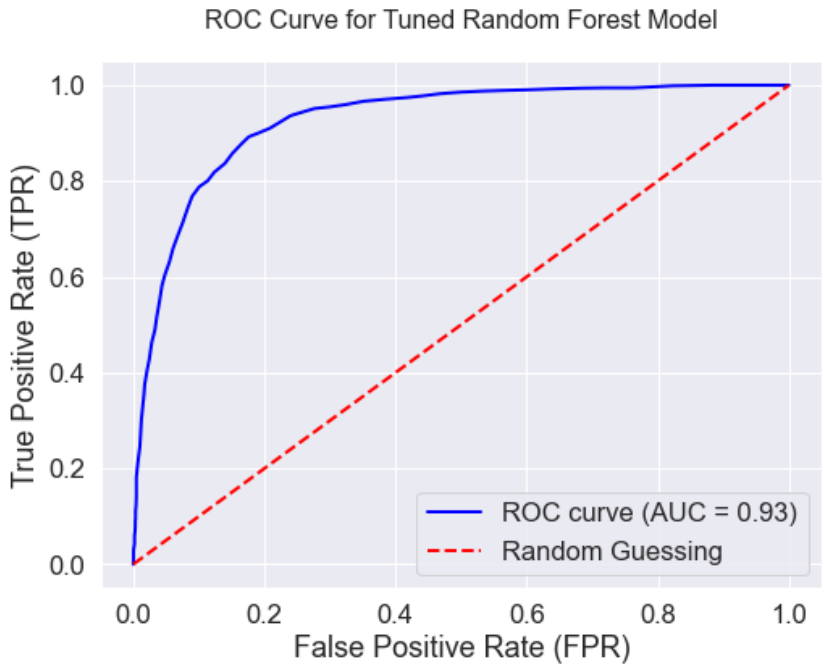
```
{'model__criterion': 'entropy', 'model__max_depth': None, 'model__min_samples_leaf': 1}
CPU times: user 299 µs, sys: 141 µs, total: 440 µs
Wall time: 359 µs
```

Random forest — tuned

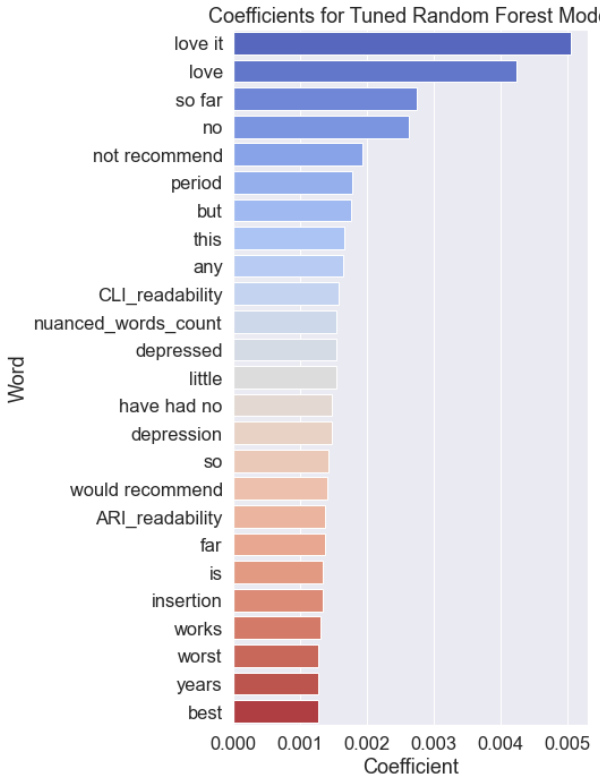
In [247]:
```
%%time

run_model_2(RandomForestClassifier(
    random_state=SEED,
    class_weight=class_weight,
    criterion='entropy',
    max_depth=None,
    min_samples_leaf=1
),
           'Tuned Random Forest'
           )
```

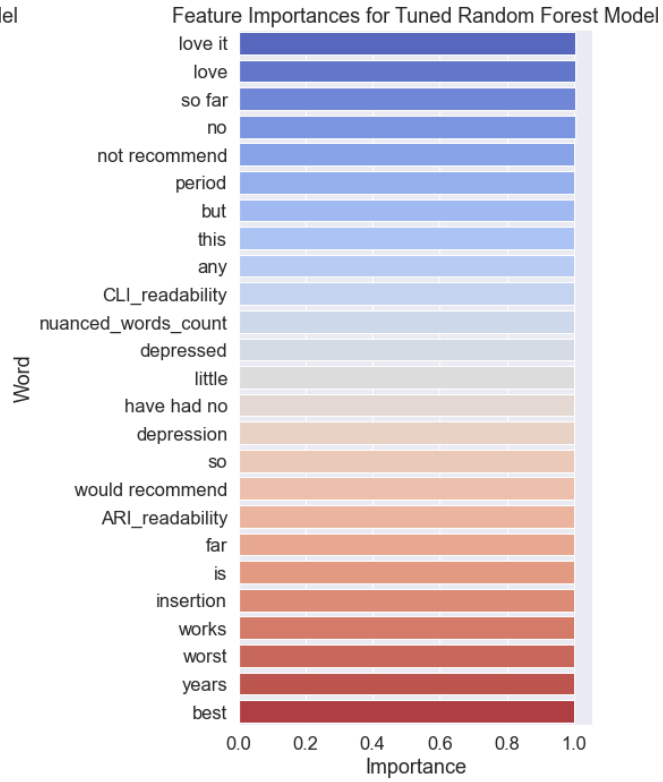
```
Training Score: 1.0
Test Score:     0.8314

Accuracy Score: 0.8314
Recall Score: 0.6303
F1 Score: 0.7297
ROC-AUC Score: 0.9263
```






```
CPU times: user 3min 13s, sys: 2.85 s, total: 3min 16s
Wall time: 3min 16s
```

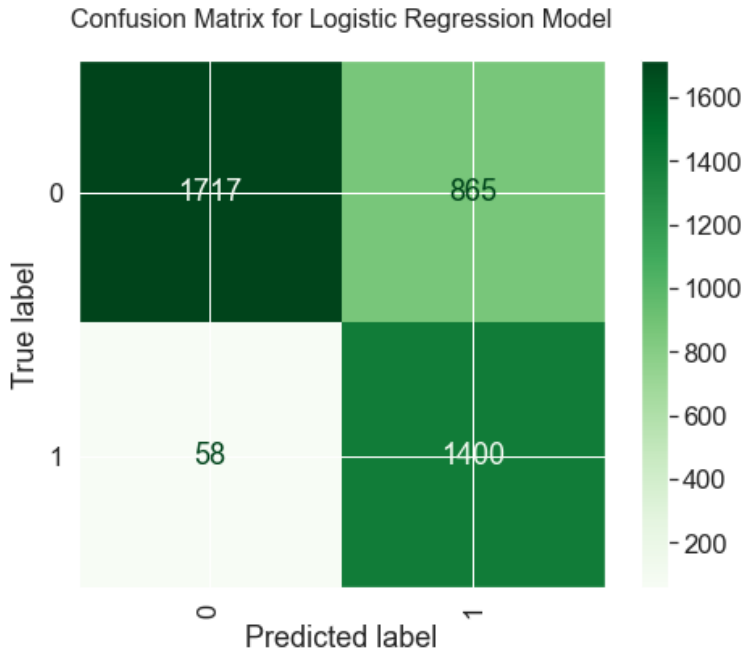


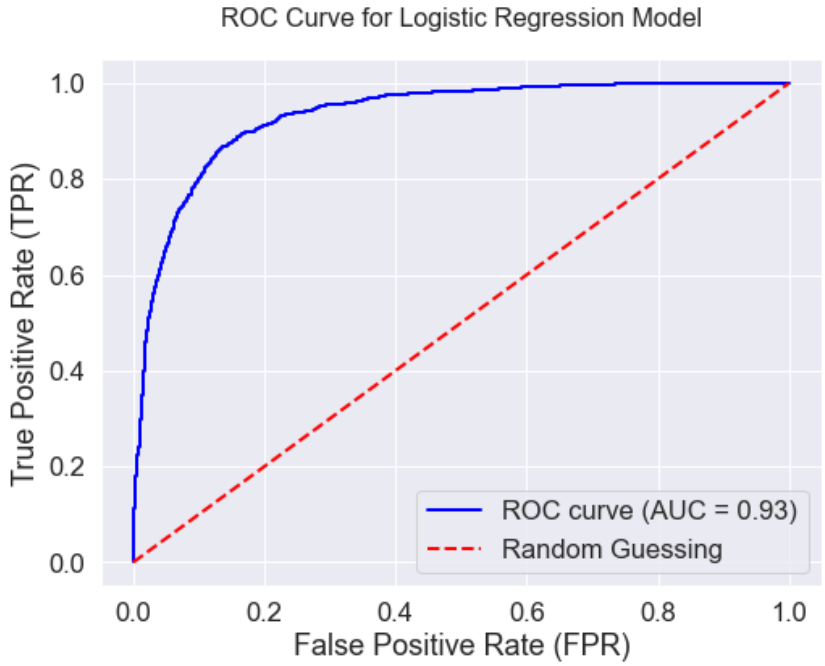


Logistic regression

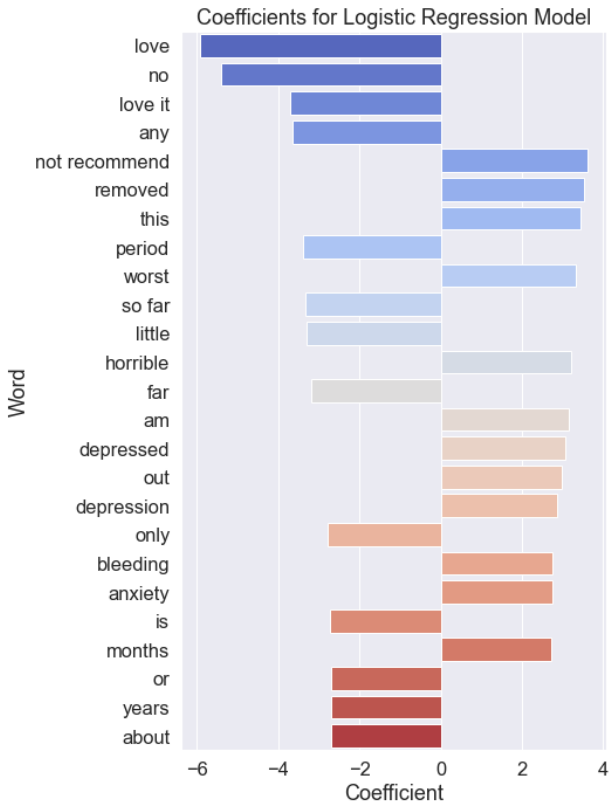
In [248]:
```
%%time

run_model_2(
    LogisticRegression(random_state=SEED, class_weight=class_weight, max_iter=1000),
    'Logistic Regression'
)
```

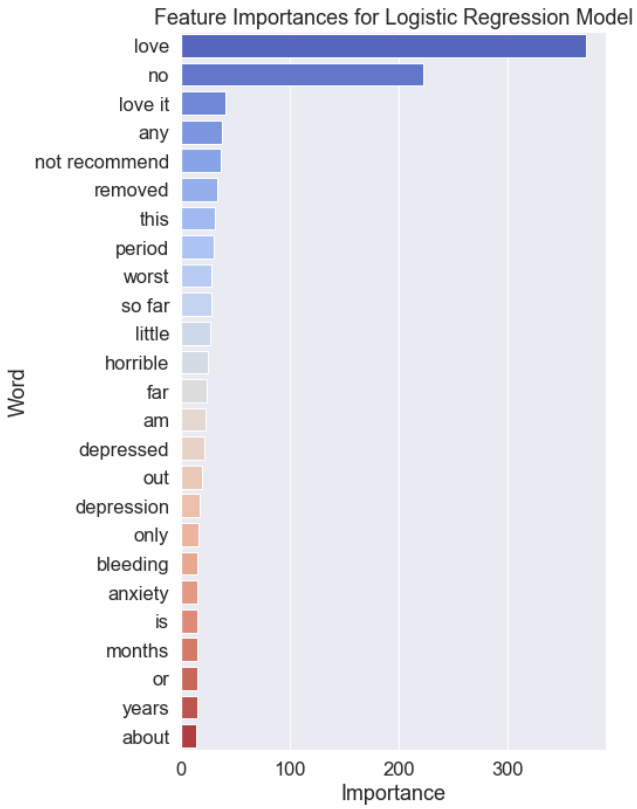
```
Training Score: 0.7857
Test Score:     0.7715

Accuracy Score: 0.7715
Recall Score: 0.9602
F1 Score: 0.7521
ROC-AUC Score: 0.9331
```






```
CPU times: user 1min 50s, sys: 53.3 s, total: 2min 44s
Wall time: 49.2 s
```





Tuning the logistic regression model

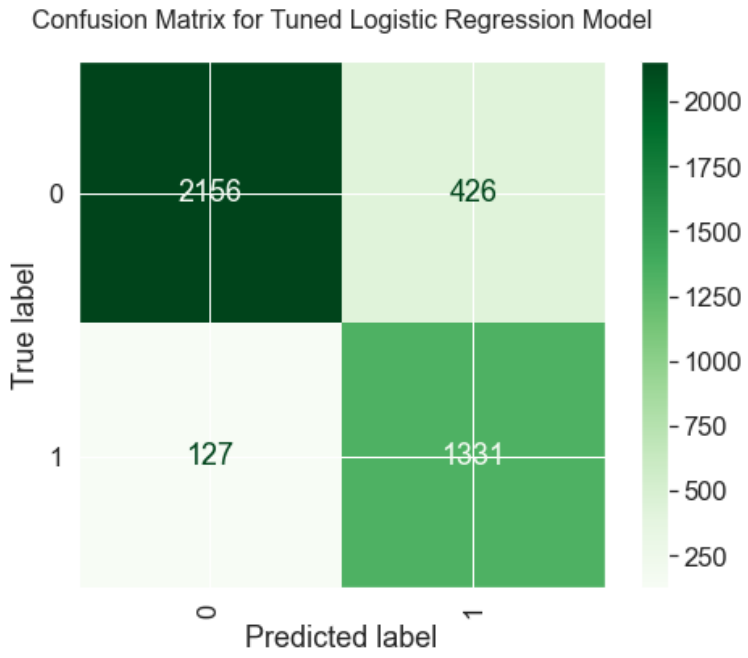
In [249]:
```
%%time

if do_grids == True:
    best_params_ = tune_model_2(LogisticRegression(random_state=SEED, max_iter=1000), param_g
        'model__C': [0.001, 0.01, 0.1, 1, 10, 100],  # Regularization parameter
        'model__penalty': ['l1', 'l2'],  # Regularization penalty ('l1' for Lasso, 'l2' for R
        'model__solver': ['liblinear', 'saga']  # Algorithm to use in the optimization proble
    })
else:
    best_params_ = "{'model__C': 10, 'model__penalty': 'l2', 'model__solver': 'liblinear'}"
print(best_params_)
```

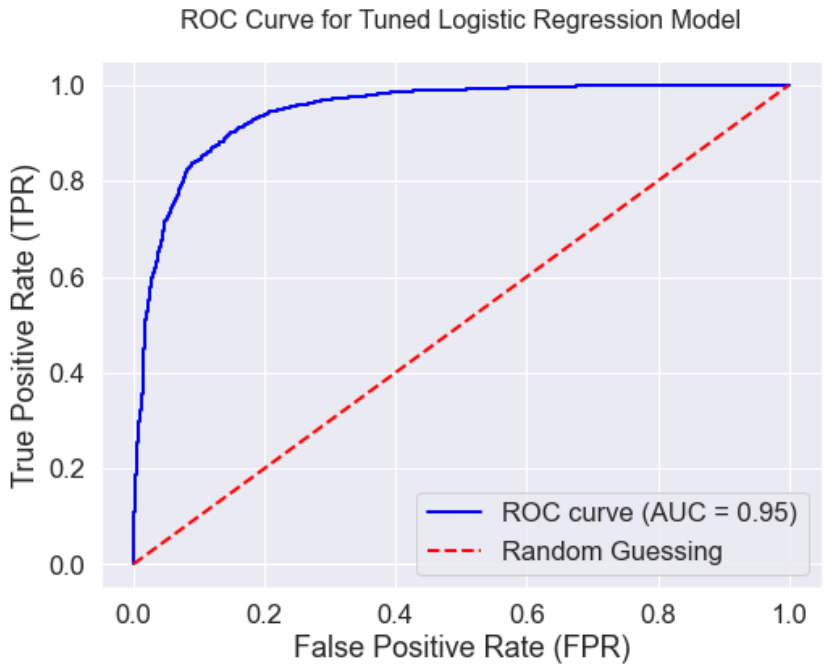
```
{'model__C': 10, 'model__penalty': 'l2', 'model__solver': 'liblinear'}
CPU times: user 208 µs, sys: 85 µs, total: 293 µs
Wall time: 260 µs
```

Logistic regression — tuned

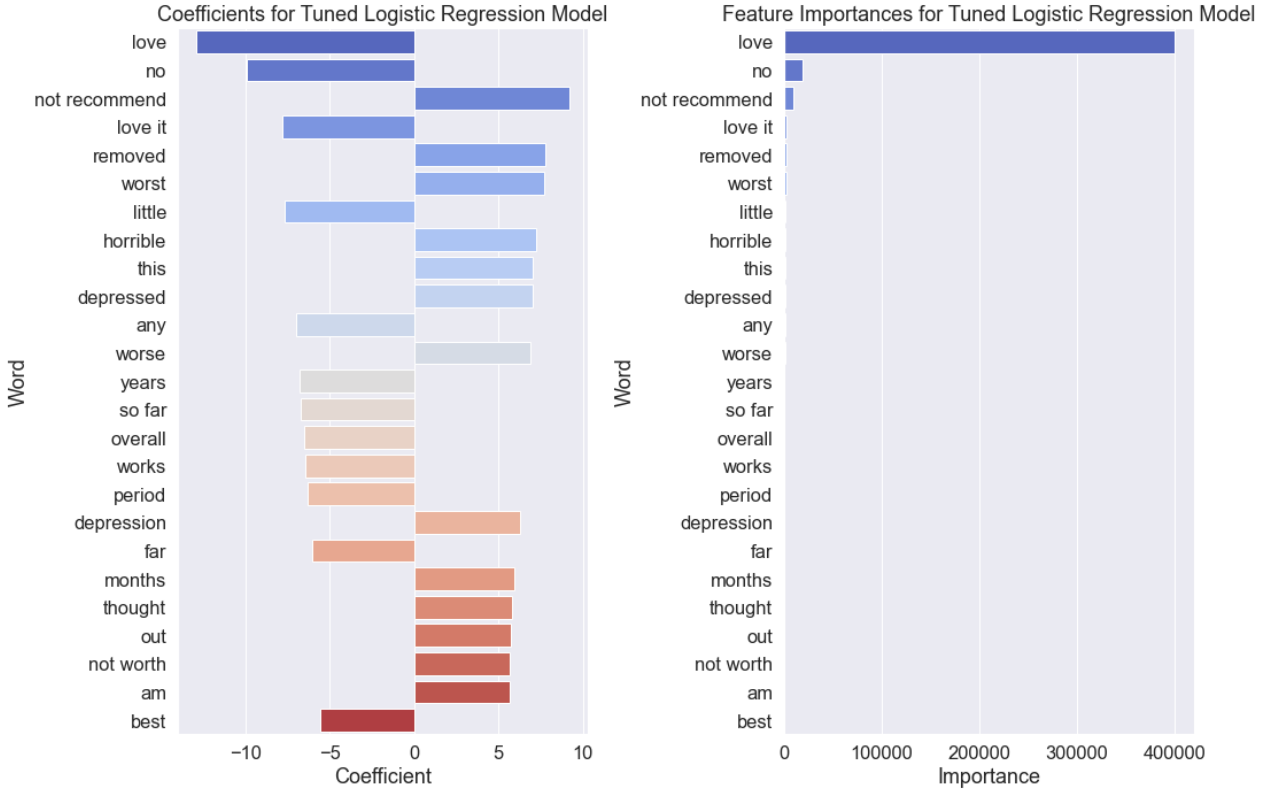
In [250]:
```
%%time

run_model_2(LogisticRegression(
    random_state=SEED,
    class_weight=class_weight,
    max_iter=1000,
    C=10,
    penalty='l2',
    solver='liblinear'
),
            'Tuned Logistic Regression'
            )
```

```
Training Score: 0.9763
Test Score:     0.8631

Accuracy Score: 0.8631
Recall Score: 0.9129
F1 Score: 0.828
ROC-AUC Score: 0.9456
```






```
CPU times: user 48.2 s, sys: 10.6 s, total: 58.8 s
Wall time: 30.3 s
```



This has the best ROC-AUC (0.95) and accuracy (86%) scores of all. Recall decreased a little bit from the untuned logistic regression model (96% -> 91%), but the tradeoff is worth it.

Sentiment prediction model evaluation

Our final model is again the tuned logistic regression model. It delivers the highest ROC-AUC score, which was our primary scoring metric. The overall accuracy of 86% is quite impressive given the class imbalance.

We also have the benefit of feature importances here. Although none of our engineered features appeared in the top 25 most important features for our final model, The "Nuanced Words Count" and the readability scores appeared in multiple other models' top 25 most important features. This suggests that certain composition styles can be indicative of the sentiment expressed. Unfortunately, the coefficients for these features were only captured in models that had all positive coefficients, so we are unable at this point to see which sentiment they were indicative of!

Recommendations

1. Consider developing and marketing a patch as an alternative to the pill

Users rate the patch quite highly, but it is used much less than the pill. There is room for growth here.

2. Emphasize favorable weight and body image effects of patch methods

Our word clouds and term importance analysis shows that users appreciate these aspects of the patch.

3. Abandon injectable product (Depo-Provera)

This method is not commonly used and not highly rated.

4. Apply our prediction models in online spaces

Look in spaces such as Reddit and Quora to gather updated data on public sentiment toward the various birth control methods.

Further Inquiry

- Incorporate upvotes more into analysis and modeling
- Explore the possibility of a recommendation algorithm
- More feature engineering possibilities
  - whether words are in English (spelled correctly)
  - use of emoticons
  - whether it uses slang sex terms versus technical language
- Analyze evidence of reviewers switching methods or brands
- Understand what went wrong with XGB reproducibility in the multiclass model
- Scrape new text from Reddit, Quora and apply modeling to it

Aaron Galbraith

https://www.linkedin.com/in/aarongalbraith (https://www.linkedin.com/in/aarongalbraith)
https://github.com/aarongalbraith (https://github.com/aarongalbraith)