TECHNICAL REPORT DRUG RECOMMENDER SYSTEM

Hardianti Ekaputri[1]

[1]Departement Mathematics, Faculty of Mathematics and Science, Universitas Indonesia

I. Introduction

Drug recommender systems, which give patients individualized suggestions for the choice and administration of medications, are crucial in the healthcare and pharmaceutical industries. Recommender systems have become important instruments in aiding healthcare providers in making knowledgeable judgments about drug therapy because to the rapid growth of technology and the accessibility of large volumes of medical data. These systems assess patient-specific data, including medical history, symptoms, genetic variables, and drug interactions, and produce personalised prescription recommendations by utilizing sophisticated algorithms and machine learning techniques.

The main goals of drug recommendation systems are to improve patient care and maximize therapeutic results. As a result of their subjectivity and susceptibility to human error, healthcare professionals' knowledge and experience are frequently relied upon in traditional methods of drug prescription. However, by utilizing data-driven methodologies, recommender systems are able to offer recommendations that are based on solid evidence and customized to meet the needs of each individual patient while taking into account a number of variables that affect the efficacy and safety of a treatment.

Medication recommender systems use underlying technologies that combine data from a variety of sources, including electronic health records (EHRs), clinical databases, academic publications, and medication interaction databases. These systems mine these enormous stores of medical data for useful patterns and correlations using modern data processing techniques including natural language processing (NLP), data mining, and predictive modeling. Recommender systems can efficiently identify the drugs that are most likely to be helpful for a certain patient by using algorithms including collaborative filtering, content-based filtering, and hybrid techniques.

Systems that recommend drugs have several potential advantages. First and foremost, they make it possible for medical professionals to acquire thorough and current data on drug interactions, contraindications, and side effects, enabling better prescribing procedures. Second, by taking into consideration patient-specific elements like age, gender, comorbidities, and genetic predispositions, these systems can help identify individualized therapy alternatives. Additionally, by lowering medication errors, decreasing trial-and-error methods, and refining treatment regimens, drug recommender systems have the potential to increase the efficiency of healthcare delivery, leading to better patient outcomes and lower healthcare costs.

Drug recommender system development and implementation, however, present numerous difficulties. It is essential to ensure the precision and dependability of recommendations because mistakes in medicine selection or dosage might have negative effects on a patient's health. To preserve patient data and adhere to legal and ethical obligations, privacy and security issues must also be addressed. Additional challenges to

be overcome include the incorporation of recommender systems into current healthcare workflows and the acceptance and adoption of these technologies by healthcare personnel.

II. Method

III. Result and Discussion

Data collecting and preprocessing

Import library

```
import numpy as np
import pandas as pd
import os
import spacy
import en_core_web_sm
from spacy.lang.en import English
from spacy.lang.en.stop_words import STOP_WORDS
import string
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.feature_extraction import DictVectorizer
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer,HashingVectorizer
from sklearn.base import TransformerMixin
from sklearn.pipeline import Pipeline
!pip install vaderSentiment
import vaderSentiment
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

Uploading files and read the files

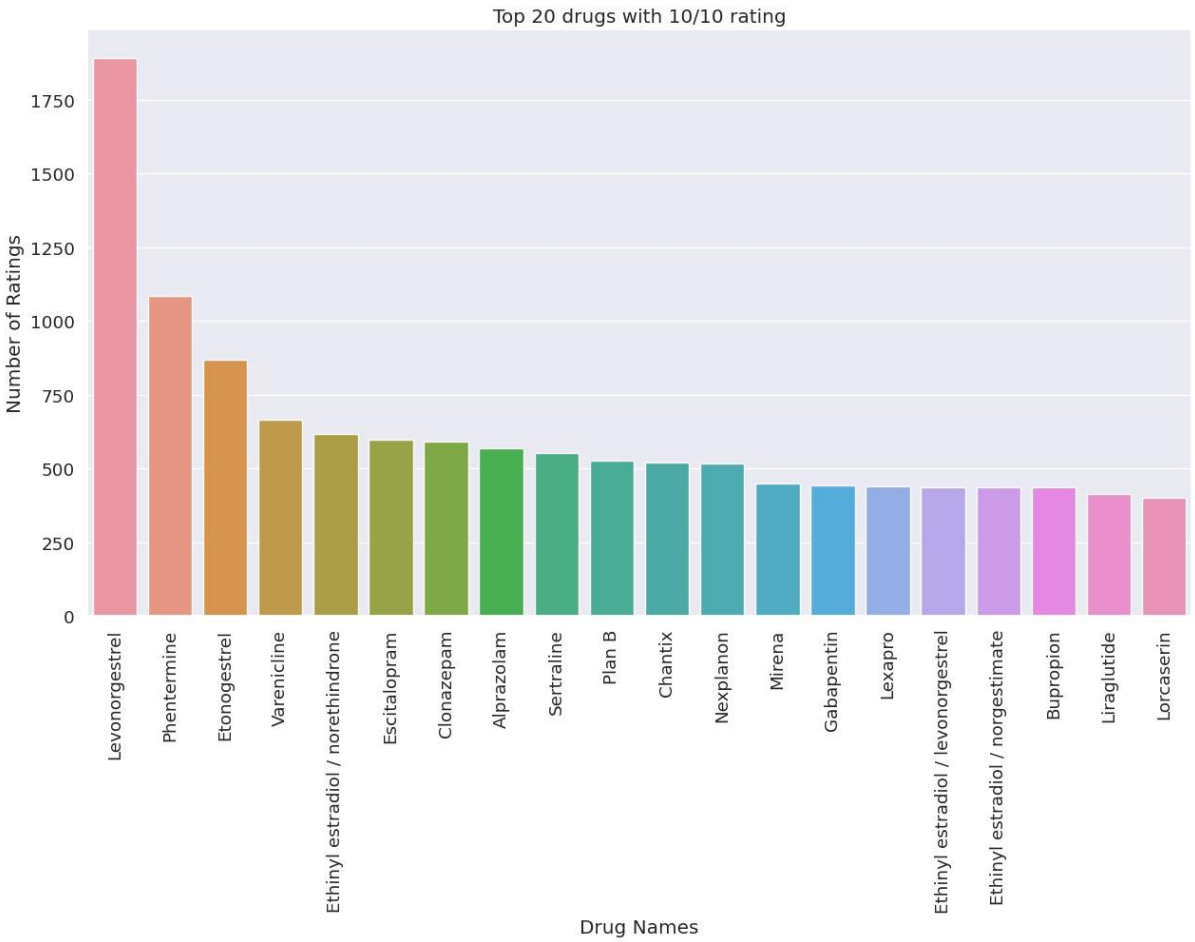
```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)


from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
# preprocessing the data file
#  read the data
df1 = pd.read_csv("/content/gdrive/MyDrive/Colab Notebooks/drugsComTrain_raw.csv")
df2 = pd.read_csv("/content/gdrive/MyDrive/Colab Notebooks/drugsComTest_raw.csv")
# combine two file
df = pd.concat([df1, df2])
df
# rename the cols
df.columns = ['ID','drug name','condition','review','rating','date','useful count']
```

```
import seaborn as sns
#!pip install matplotlib
import matplotlib.pyplot as plt

# Setting the Parameter
sns.set(font_scale = 1.2, style = 'darkgrid')
plt.rcParams['figure.figsize'] = [15, 8]

rating = dict(df.loc[df.rating == 10, "drug name"].value_counts())
drugname = list(rating.keys())
drug_rating = list(rating.values())

sns_rating = sns.barplot(x = drugname[0:20], y = drug_rating[0:20])

sns_rating.set_title('Top 20 drugs with 10/10 rating')
sns_rating.set_ylabel("Number of Ratings")
sns_rating.set_xlabel("Drug Names")
plt.setp(sns_rating.get_xticklabels(), rotation=90);
```

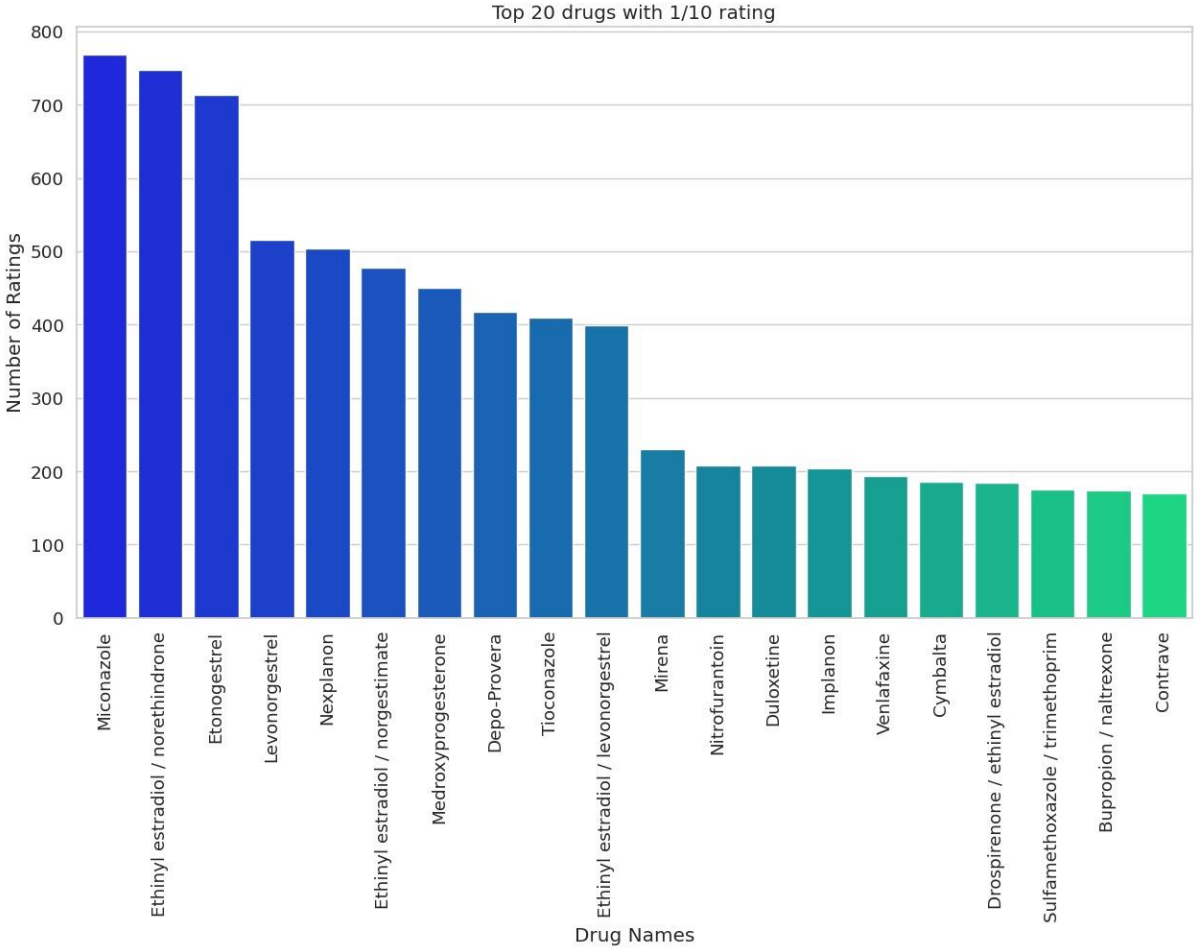
To view the data, import the seaborn and matplotlib packages. Decide on the parameters first, then tally the number of ratings that are 10. Next, make a bar plot with drug_rating on the y-axis and drugname on the x-axis.



This is the bar plot that was previously made. Here, the top 20 medications with the highest overall rating of 10 may be shown. Levonorgestrel, followed by Phentermine and other drugs, has the highest overall rating value of all of them at over 1750.

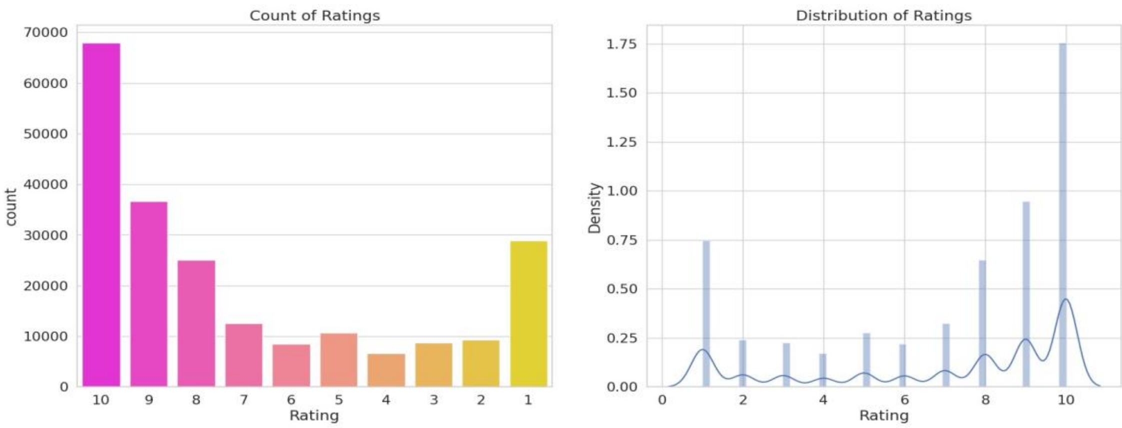
```
# Setting the Parameter
sns.set(font_scale = 1.2, style = 'whitegrid')
plt.rcParams['figure.figsize'] = [15, 8]

rating = dict(df.loc[df.rating == 1, "drug name"].value_counts())
drugname = list(rating.keys())
drug_rating = list(rating.values())

sns_rating = sns.barplot(x = drugname[0:20], y = drug_rating[0:20], palette = 'winter')

sns_rating.set_title('Top 20 drugs with 1/10 rating')
sns_rating.set_ylabel("Number of Ratings")
sns_rating.set_xlabel("Drug Names")
plt.setp(sns_rating.get_xticklabels(), rotation=90);
```

To view the data, import the seaborn and matplotlib packages. Decide on the parameters first, then tally the number of ratings that are 1. Next, make a bar plot using drug_rating on the y-axis and drug_name on the x-axis.



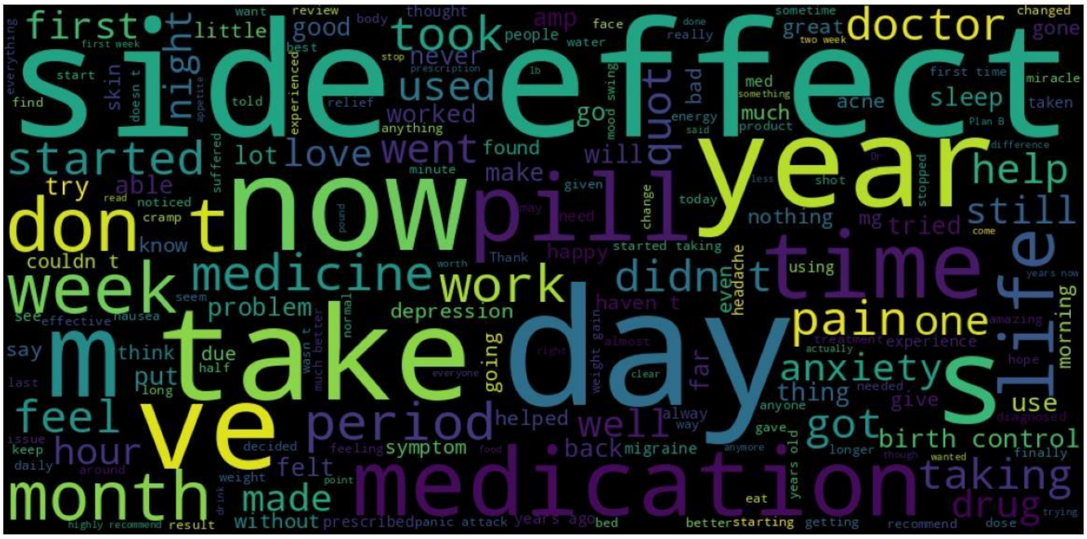
This is the previously created bar plot. Here, it can be observed that there are the top 20 drugs with the highest total rating of 1. Among them, the drug Miconacole has the highest total rating value of over 750, followed by the drug Ethinyl estradiol and others.

```python
# A countplot of the ratings so we can see the distribution of the ratings
plt.rcParams['figure.figsize'] = [20,8]
sns.set(font_scale = 1.4, style = 'whitegrid')
fig, ax = plt.subplots(1, 2)
sns_1 = sns.countplot(x='rating', data=df, palette='spring', order=list(range(10, 0, -1)), ax=ax[0])
sns_2 = sns.distplot(df['rating'], ax = ax[1])
sns_1.set_title('Count of Ratings')
sns_1.set_xlabel("Rating")

sns_2.set_title('Distribution of Ratings')
sns_2.set_xlabel("Rating")
```

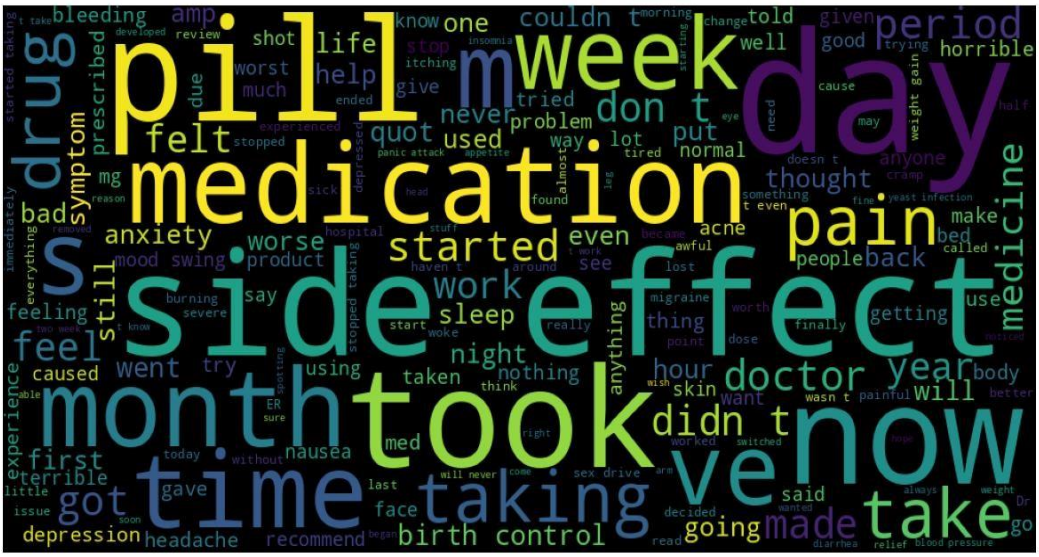


Display the total rating and rating distribution using a countplot.

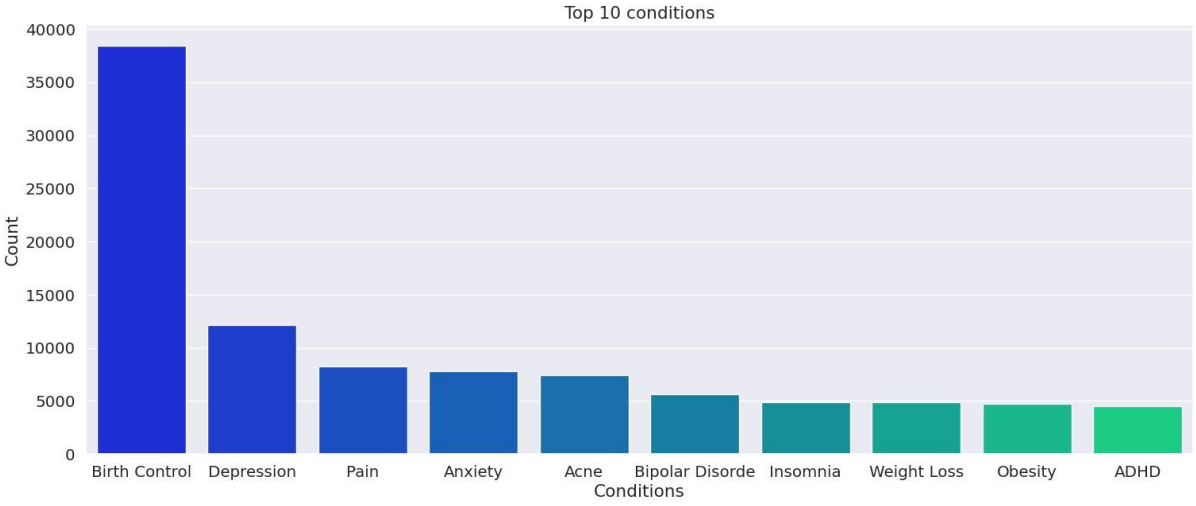
```python
#!pip install wordcloud
from wordcloud import WordCloud# Word cloud of the reviews with rating equal to 10
df_rate_ten = df.loc[df.rating == 10, 'review']
k = (' '.join(df_rate_ten))

wordcloud = WordCloud(width = 1000, height = 500).generate(k)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud)
plt.axis('off');
```



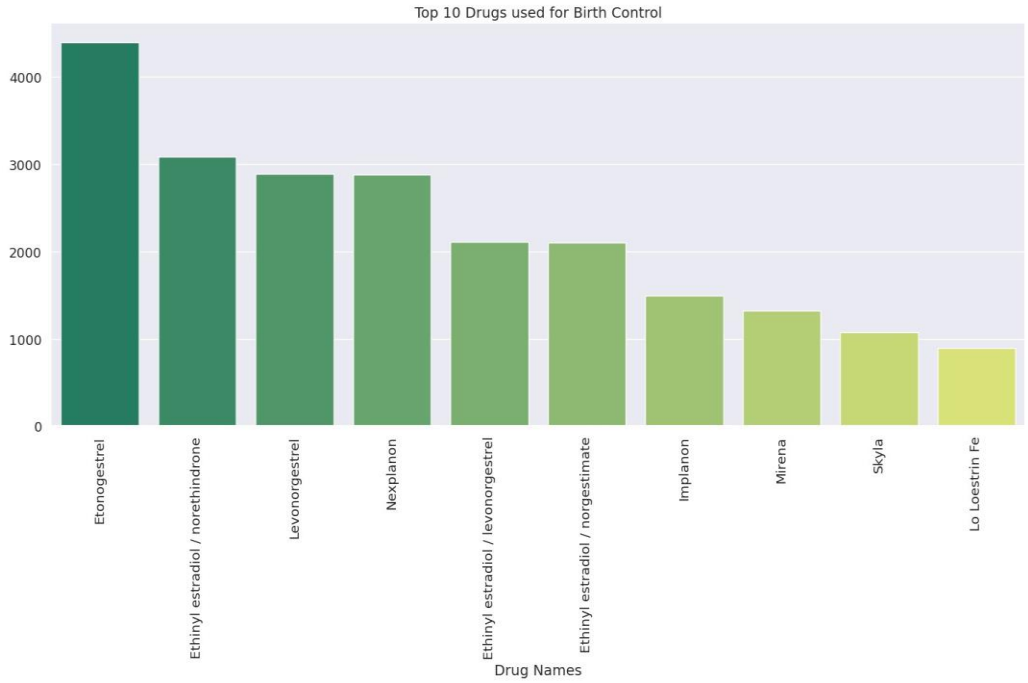
```
from wordcloud import WordCloud# Word cloud of the reviews with rating equal to 1
df_rate_ten = df.loc[df.rating == 1, 'review']
k = (' '.join(df_rate_ten))

wordcloud = WordCloud(width = 1000, height = 500).generate(k)
plt.figure(figsize=(15, 10))
plt.imshow(wordcloud)
plt.axis('off');
```




```
# This barplot show the top 10 conditions the people are suffering.
cond = dict(df['condition'].value_counts())
top_condition = list(cond.keys())[0:10]
values = list(cond.values())[0:10]
sns.set(style = 'darkgrid', font_scale = 1.3)
plt.rcParams['figure.figsize'] = [18, 7]

sns_ = sns.barplot(x = top_condition, y = values, palette = 'winter')
sns_.set_title("Top 10 conditions")
sns_.set_xlabel("Conditions")
sns_.set_ylabel("Count");
```



```
# Top 10 drugs which are used for the top condition, that is Birth Control
df_cond_birth = df[df['condition'] == 'Birth Control']['drug name'].value_counts()[0: 10]
sns.set(font_scale = 1.2, style = 'darkgrid')

sns_ = sns.barplot(x = df_cond_birth.index, y = df_cond_birth.values, palette = 'summer')
sns_.set_xlabel('Drug Names')
sns_.set_title("Top 10 Drugs used for Birth Control")
plt.setp(sns_.get_xticklabels(), rotation = 90);
```



This bar plot displays the top 10 drugs that are used for birth control.

```
df2 = df[df['useful count'] > 10]
```

```
df_condition = df2.groupby(['condition'])['drug name'].nunique().sort_values(ascending=False)
df_condition = pd.DataFrame(df_condition).reset_index()
df_condition.tail(20)
```

	condition	drug name
706	64 users found this comment helpful.	1
707	92 users found this comment helpful.	1
708	Gastritis/Duodenitis	1
709	Esophageal Variceal Hemorrhage Prophylaxis	1
710	98 users found this comment helpful.	1
711	Severe Mood Dysregulation	1
712	Short Stature	1
713	Short Stature for Age	1
714	Meningitis	1
715	Skin Disinfection, Preoperative	1
716	Melanoma	1

Calculate the total "useful count" that has a value greater than 10 based on the conditions and drugs used.

```
df_condition_1 = df_condition[df_condition['drug name'] == 1].reset_index()

all_list = set(df.index)

# deleting them
condition_list = []
for i,j in enumerate(df['condition']):
    for c in list(df_condition_1['condition']):
        if j == c:
            condition_list.append(i)

new_idx = all_list.difference(set(condition_list))
df = df.iloc[list(new_idx)].reset_index()
del df['index']
```

Remove the "condition" values that have only one drug associated with them, and then reset the index to ensure that the index is sorted after removing some rows of data.

```
# removing the conditions with  in it.

all_list = set(df.index)
span_list = []
for i,j in enumerate(df['condition']):
    if "" in str(j):
        span_list.append(i)
new_idx = all_list.difference(set(span_list))
df = df.iloc[list(new_idx)].reset_index()
del df['index']
```

Remove the "condition" values that contain "", and then reset the index to ensure that the index is sorted after removing some rows of data.

Import the nltk library, which includes the stopwords dictionary used for text-based data processing.

```
import re
from bs4 import BeautifulSoup
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Removing some stopwords from the list of stopwords that are important for drug recommendations.

```
# removing some stopwords from the list of stopwords as they are important for drug recommendation

stops = set(stopwords.words('english'))

not_stop = ["aren't","couldn't","didn't","doesn't","don't","hadn't","hasn't","haven't","isn't","mightn't",
            "mustn't","needn't","no","nor","not","shan't","shouldn't","wasn't","weren't","wouldn't"]
for i in not_stop:
    stops.remove(i)
```

```
stemmer = SnowballStemmer('english')

def review_to_words(raw_review):
    # 1. Delete HTML
    review_text = BeautifulSoup(raw_review, 'html.parser').get_text()
    # 2. Make a space
    letters_only = re.sub('[^a-zA-Z]', ' ', review_text)
    # 3. lower letters
    words = letters_only.lower().split()
    # 5. Stopwords
    meaningful_words = [w for w in words if not w in stops]
    # 6. Stemming
    stemming_words = [stemmer.stem(w) for w in meaningful_words]
    # 7. space join words
    return( ' '.join(stemming_words))

# create a list of stopwords
nlp = spacy.load('en_core_web_sm')
stop_words = spacy.lang.en.stop_words.STOP_WORDS
parser = English()
punctuations = string.punctuation
# Creating our tokenizer function
def spacy_tokenizer(sentence):
    # Creating our token object, which is used to create documents with linguistic annotations.
    mytokens = parser(sentence)

    # Lemmatizing each token and converting each token into lowercase
    mytokens = [ word.lemma_.lower().strip() if word.lemma_ != "-PRON-" else word.lower_ for word in mytokens ]

    # Removing stop words
    mytokens = [ word for word in mytokens if word not in stop_words and word not in punctuations ]

    # return preprocessed list of tokens
    return mytokens
```