# CSE 676/B Deep Learning, Spring 2024
# Deep Learning Analysis of Drug Reviews for Enhanced Health Care
# Final Project – Checkpoint


**Team Members**

**Name:** Bhavesh Tharlapally
**UBIT:** bhavesht
**UB Person Number:** 50541076

**Name:** Sreya Sirivella
**UBIT:** sreyasir
**UB Person Number:** 50537841



**Short Summary of our project:**

This project aims to develop a sentiment analysis model for drug reviews utilizing deep learning technology. Developing a model which has the ability to understand the sentiment expressed in patient reviews of pharmaceutical drugs is of paramount importance as it allows us to identify the side effects, assess the treatments effectiveness, helps to understand the patients experience and enables early detection of issues. This meaningful information helps healthcare professionals make informed decisions, improve the patients care and pay attention to the drug monitoring process. This thorough analysis also helps public make smart and informed decisions about medicines and health care.


**Dataset:**

The dataset we plan to use is retrieved from the well-known UCI Machine Learning Repository.
**UCI Machine Learning Repository:** https://archive.ics.uci.edu/
**Drug Review Dataset:**
https://archive.ics.uci.edu/dataset/462/drug+review+dataset+drugs+com

The dataset has two .tsv files namely, "drugsComTrain_raw", "drugsComTest_raw". These files have primarily 7 columns, namely, ID, drugName, condition, review, rating, date, usefulCount. ID holds values of random integers, drugName, condition, review hold values of type string, rating and usefulCount hold integer values again and date holds values in date format.

**Data Pre-processing steps:**

**Importing necessary libraries in our notebook:**

```python
# Importing necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
```

**Uploading the drug review training and testing dataset:**

```
# Uploading the drug review training dataset

from google.colab import files

uploaded_train = files.upload()
```

```
Choose Files drugsComTrain_raw.tsv
• drugsComTrain_raw.tsv(n/a) - 84289175 bytes, last modified: 10/2/2018 - 100% done
Saving drugsComTrain_raw.tsv to drugsComTrain_raw.tsv
```

```
# Uploading the drug review testing dataset

uploaded_test = files.upload()
```

```
Choose Files drugsComTest_raw.tsv
• drugsComTest_raw.tsv(n/a) - 28071166 bytes, last modified: 10/2/2018 - 100% done
Saving drugsComTest_raw.tsv to drugsComTest_raw.tsv
```

**Iterating through the above uploaded files to convert them into .csv format:**

```
# Iterating through the above uploaded files to convert them into .csv format

with open("train_data.tsv", "wb") as f:
    f.write(uploaded_train[next(iter(uploaded_train))])

with open("test_data.tsv", "wb") as f:
    f.write(uploaded_test[next(iter(uploaded_test))])
```

```
# Loading the dataset into a dataframe

train_data = pd.read_csv("train_data.tsv", delimiter='\t')
test_data = pd.read_csv("test_data.tsv", delimiter='\t')
```

```
# Saving the files in .csv format

train_data.to_csv("train_data.csv", index=False)
test_data.to_csv("test_data.csv", index=False)
```

Here we convert the file type from. tsv to .csv for easier data pre-processing.

**First few rows of training datasets:**

```
# First few rows of loaded datasets

print("Train Data Head:\n", train_data.head())
```

```
Train Data Head:
    Unnamed: 0                  drugName                     condition  \
0      206461                 Valsartan  Left Ventricular Dysfunction
1       95260                Guanfacine                          ADHD
2       92703                    Lybrel                 Birth Control
3      138000                Ortho Evra                 Birth Control
4       35696  Buprenorphine / naloxone             Opiate Dependence

                                              review  rating  \
0  "It has no side effect, I take it in combinati...     9.0
1  "My son is halfway through his fourth week of ...     8.0
2  "I used to take another oral contraceptive, wh...     5.0
3  "This is my first time using any form of birth...     8.0
4  "Suboxone has completely turned my life around...     9.0

                date  usefulCount
0       May 20, 2012           27
1     April 27, 2010          192
2  December 14, 2009           17
3   November 3, 2015           10
4  November 27, 2016           37
```

**Testing Dataset:**

```
print("Test Data Head:\n", test_data.head())
```

```
Test Data Head:
    Unnamed: 0         drugName                     condition  \
0      163740      Mirtazapine                    Depression
1      206473       Mesalamine  Crohn's Disease, Maintenance
2      159672          Bactrim       Urinary Tract Infection
3       39293         Contrave                   Weight Loss
4       97768  Cyclafem 1 / 35                 Birth Control

                                              review  rating  \
0  "I&#039;ve tried a few antidepressants over th...    10.0
1  "My son has Crohn&#039;s disease and has done ...     8.0
2                      "Quick reduction of symptoms"     9.0
3  "Contrave combines drugs that were used for al...     9.0
4  "I have been on this birth control for one cyc...     9.0

                 date  usefulCount
0   February 28, 2012           22
1        May 17, 2009           17
2  September 29, 2017            3
3       March 5, 2017           35
4    October 22, 2015            4
```

**Dataset Shape:**

```
# Shape of the datasets
print(train_data.shape)
print(test_data.shape)
```

```
(161297, 7)
(53766, 7)
```

**Describing the training dataset:**

```
Description of train dataset:
            Unnamed: 0         rating    usefulCount
count  161297.000000  161297.000000  161297.000000
mean   115923.585305       6.994377      28.004755
std     67004.445170       3.272329      36.403742
min         2.000000       1.000000       0.000000
25%     58063.000000       5.000000       6.000000
50%    115744.000000       8.000000      16.000000
75%    173776.000000      10.000000      36.000000
max    232291.000000      10.000000    1291.000000
```

**Testing Dataset:**

```
Description of testing dataset:
           Unnamed: 0        rating   usefulCount
count    53766.000000  53766.000000  53766.000000
mean    116386.701187      6.976900     27.989752
std      67017.739881      3.285207     36.172833
min          0.000000      1.000000      0.000000
25%      58272.500000      4.000000      6.000000
50%     116248.500000      8.000000     16.000000
75%     174586.750000     10.000000     36.000000
max     232284.000000     10.000000    949.000000
```

**Information of the datasets and its datatypes:**

```
# Information on the dataset

print(train_data.info())
print("\n",test_data.info())
```

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

**Testing Dataset:**

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

 None
```

As described earlier, the data types of feature columns are as expected and hence we do not need to change the datatype for any existing feature.

**Displaying unique entries in Unnamed:0**

```
# Data Preprocessing
# Examining "Unnamed: 0"

print(train_data['Unnamed: 0'].unique())
print(test_data['Unnamed: 0'].unique())
```

```
[206461  95260  92703 ... 187382  47128 215220]
[163740 206473 159672 ... 130945  47656 113712]
```

Here, unnamed:0 is just being used as an identifier for every row / entry in the dataset and hence if does add any significant value / meaning to our dataset as we can see below the value counts for all entries in unnamed: 0 is exactly "1".

**Value Counts:**

```
print(train_data['Unnamed: 0'].value_counts())
print(test_data['Unnamed: 0'].value_counts())
```

```
Unnamed: 0
206461    1
115685    1
78842     1
151214    1
225627    1
         ..
140483    1
29358     1
65306     1
26066     1
215220    1
Name: count, Length: 161297, dtype: int64
Unnamed: 0
163740    1
99146     1
195954    1
121131    1
88774     1
         ..
139402    1
167880    1
83507     1
37059     1
113712    1
Name: count, Length: 53766, dtype: int64
```

Dropping column unnamed:0

```
# Each column has a unique identifer represented with random integer values in column "Unnamed: 0"
# Dropping column as a part of pre-processing

train_data.drop(columns=['Unnamed: 0'], inplace=True)
test_data.drop(columns=['Unnamed: 0'], inplace=True)
```

### Checking for missing values in the dataset:

```
missing_values_train = train_data.isnull().sum()
print("Missing Values in Train Data:")
print(missing_values_train)

missing_values_test = test_data.isnull().sum()
print("\nMissing Values in Test Data:")
print(missing_values_test)
```

```
Missing Values in Train Data:
drugName         0
condition      899
review           0
rating           0
date             0
usefulCount      0
dtype: int64

Missing Values in Test Data:
drugName         0
condition      295
review           0
rating           0
date             0
usefulCount      0
dtype: int64
```

Percentage of missing values in Training and Testing Dataset is 0.55 and 0.54% respectively. Hence, dropping rows with missing values.

```
Percentage of Missing Values in Train Data:
drugName       0.000000
condition      0.557357
review         0.000000
rating         0.000000
date           0.000000
usefulCount    0.000000
dtype: float64

Percentage of Missing Values in Test Data:
drugName       0.000000
condition      0.548674
review         0.000000
rating         0.000000
date           0.000000
usefulCount    0.000000
dtype: float64
```

### Dropping rows with missing values:

```
# We drop rows with missing values

train_data.dropna(inplace=True)
test_data.dropna(inplace=True)
```

## Frequency of drugs in drugName:

```
# Frequency of drugs in "drugName"

print(train_data['drugName'].value_counts())
```

```
drugName
Levonorgestrel                        3657
Etonogestrel                          3336
Ethinyl estradiol / norethindrone     2850
Nexplanon                             2156
Ethinyl estradiol / norgestimate      2117
                                      ...
Omnipaque 350                            1
Vontrol                                  1
Ivabradine                               1
Neo-Poly-Dex                             1
Grifulvin V                              1
Name: count, Length: 3436, dtype: int64
```

## Post data cleaning:

```
Missing Values in Train Data:
drugName       0
condition      0
review         0
rating         0
date           0
usefulCount    0
dtype: int64

Missing Values in Test Data:
drugName       0
condition      0
review         0
rating         0
date           0
usefulCount    0
dtype: int64
```

## Normalizing rating and usefulCount columns:

```
# Normalizing rating and usefulCount columns

# Training Dataset
train_data['rating'] = (train_data['rating'] - train_data['rating'].min()) / (train_data['rating'].max() - train_data['rating'].min())
train_data['usefulCount'] = (train_data['usefulCount'] - train_data['usefulCount'].min()) / (train_data['usefulCount'].max() - train_data['usefulCount'].min())

# Testing Dataset
test_data['rating'] = (test_data['rating'] - test_data['rating'].min()) / (test_data['rating'].max() - test_data['rating'].min())
test_data['usefulCount'] = (test_data['usefulCount'] - test_data['usefulCount'].min()) / (test_data['usefulCount'].max() - test_data['usefulCount'].min())
```

## Output:

```
     rating  usefulCount
0  0.888889     0.020914
1  0.777778     0.148722
2  0.444444     0.013168
3  0.777778     0.007746
4  0.888889     0.028660
     rating  usefulCount
0  0.888889     0.020914
1  0.777778     0.148722
2  0.444444     0.013168
3  0.777778     0.007746
4  0.888889     0.028660
```

## Pre-processing date column in train and test datasets:

```
# Pre-processing date column in train and test datasets

train_data['date'] = pd.to_datetime(train_data['date'])
test_data['date'] = pd.to_datetime(test_data['date'])
```

```
train_data.head()
```

| | drugName | condition | review | rating | date | usefulCount |
|---|---|---|---|---|---|---|
| 0 | Valsartan | Left Ventricular Dysfunction | "It has no side effect, I take it in combinati... | 0.888889 | 2012-05-20 | 0.020914 |
| 1 | Guanfacine | ADHD | "My son is halfway through his fourth week of ... | 0.777778 | 2010-04-27 | 0.148722 |
| 2 | Lybrel | Birth Control | "I used to take another oral contraceptive, wh... | 0.444444 | 2009-12-14 | 0.013168 |
| 3 | Ortho Evra | Birth Control | "This is my first time using any form of birth... | 0.777778 | 2015-11-03 | 0.007746 |
| 4 | Buprenorphine / naloxone | Opiate Dependence | "Suboxone has completely turned my life around... | 0.888889 | 2016-11-27 | 0.028660 |

## Testing Dataset:

```
test_data.head()
```

| | drugName | condition | review | rating | date | usefulCount |
|---|---|---|---|---|---|---|
| 0 | Valsartan | Left Ventricular Dysfunction | "It has no side effect, I take it in combinati... | 0.888889 | 2012-05-20 | 0.020914 |
| 1 | Guanfacine | ADHD | "My son is halfway through his fourth week of ... | 0.777778 | 2010-04-27 | 0.148722 |
| 2 | Lybrel | Birth Control | "I used to take another oral contraceptive, wh... | 0.444444 | 2009-12-14 | 0.013168 |
| 3 | Ortho Evra | Birth Control | "This is my first time using any form of birth... | 0.777778 | 2015-11-03 | 0.007746 |
| 4 | Buprenorphine / naloxone | Opiate Dependence | "Suboxone has completely turned my life around... | 0.888889 | 2016-11-27 | 0.028660 |

## Converting text to lowercase notation:

```
# Pre-processing review column
# Converting text to lowercase notation

train_data['review'] = train_data['review'].str.lower()
test_data['review'] = test_data['review'].str.lower()
```

```
# Printing the first few words

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    "it has no side effect, i take it in combinati...
1    "my son is halfway through his fourth week of ...
2    "i used to take another oral contraceptive, wh...
3    "this is my first time using any form of birth...
4    "suboxone has completely turned my life around...
Name: review, dtype: object
0    "it has no side effect, i take it in combinati...
1    "my son is halfway through his fourth week of ...
2    "i used to take another oral contraceptive, wh...
3    "this is my first time using any form of birth...
4    "suboxone has completely turned my life around...
Name: review, dtype: object
```

## Function used for tokenization:

```
# Text tokenization for further processing of review feature
# Defining tokenize_text function

from nltk.tokenize import word_tokenize

def tokenize_text(text):
    tokens = word_tokenize(text)
    return tokens
```

**Applying tokenization and reviewing pre-processed data:**

```
train_data['review'] = train_data['review'].apply(tokenize_text)
test_data['review'] = test_data['review'].apply(tokenize_text)
```

```
# Review tokens

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    [``, it, has, no, side, effect, ,, i, take, it...
1    [``, my, son, is, halfway, through, his, fourt...
2    [``, i, used, to, take, another, oral, contrac...
3    [``, this, is, my, first, time, using, any, fo...
4    [``, suboxone, has, completely, turned, my, li...
Name: review, dtype: object
0    [``, it, has, no, side, effect, ,, i, take, it...
1    [``, my, son, is, halfway, through, his, fourt...
2    [``, i, used, to, take, another, oral, contrac...
3    [``, this, is, my, first, time, using, any, fo...
4    [``, suboxone, has, completely, turned, my, li...
Name: review, dtype: object
```

**Removing punctuations from the above generated tokens:**

```
# Removing punctuations from the above generated tokens
# Defining function remove_punctuation
import string

def remove_punctuation(tokens):
    punctuations = string.punctuation
    tokens_without_punctuations = [token for token in tokens if token not in punctuations]
    return tokens_without_punctuations
```

**Reviewing tokens post modification:**

```
# Removing punctuations

train_data['review'] = train_data['review'].apply(remove_punctuation)
test_data['review'] = test_data['review'].apply(remove_punctuation)
```

```
# Review Tokens

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    [``, it, has, no, side, effect, i, take, it, i...
1    [``, my, son, is, halfway, through, his, fourt...
2    [``, i, used, to, take, another, oral, contrac...
3    [``, this, is, my, first, time, using, any, fo...
4    [``, suboxone, has, completely, turned, my, li...
Name: review, dtype: object
0    [``, it, has, no, side, effect, i, take, it, i...
1    [``, my, son, is, halfway, through, his, fourt...
2    [``, i, used, to, take, another, oral, contrac...
3    [``, this, is, my, first, time, using, any, fo...
4    [``, suboxone, has, completely, turned, my, li...
Name: review, dtype: object
```

**Removing Stop words:**

```
# Getting stopwords from English

stop_words = set(stopwords.words('english'))
```

```
# Defining function remove_stopwords

def remove_stopwords(tokens):
    tokens_without_stopwords = [token for token in tokens if token not in stop_words]
    return tokens_without_stopwords
```

**Token after removing stop words:**

```
# Remove stopwords

train_data['review'] = train_data['review'].apply(remove_stopwords)
test_data['review'] = test_data['review'].apply(remove_stopwords)
```

```
# Review modifications

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    [``, side, effect, take, combination, bystolic...
1    [``, son, halfway, fourth, week, intuniv, beca...
2    [``, used, take, another, oral, contraceptive,...
3    [``, first, time, using, form, birth, control,...
4    [``, suboxone, completely, turned, life, aroun...
Name: review, dtype: object
0    [``, side, effect, take, combination, bystolic...
1    [``, son, halfway, fourth, week, intuniv, beca...
2    [``, used, take, another, oral, contraceptive,...
3    [``, first, time, using, form, birth, control,...
4    [``, suboxone, completely, turned, life, aroun...
Name: review, dtype: object
```

**Lemmatization**

This helps in normalizing the text data and reducing the dimensionality of the feature space

```
# Lemmatization
# This helps in normalizing the text data and reducing the dimensionality of the feature space

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def lemmatize_tokens(tokens):
    lemmatized_tokens = [lemmatizer.lemmatize(token) for token in tokens]
    return lemmatized_tokens
```

### Applying lemmatization and reviewing tokens:

```
# Applying lemmatization to the tokens

train_data['review'] = train_data['review'].apply(lemmatize_tokens)
test_data['review'] = test_data['review'].apply(lemmatize_tokens)
```

```
# Review changes

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    [``, side, effect, take, combination, bystolic...
1    [``, son, halfway, fourth, week, intuniv, beca...
2    [``, used, take, another, oral, contraceptive,...
3    [``, first, time, using, form, birth, control,...
4    [``, suboxone, completely, turned, life, aroun...
Name: review, dtype: object
0    [``, side, effect, take, combination, bystolic...
1    [``, son, halfway, fourth, week, intuniv, beca...
2    [``, used, take, another, oral, contraceptive,...
3    [``, first, time, using, form, birth, control,...
4    [``, suboxone, completely, turned, life, aroun...
Name: review, dtype: object
```

### Stemming:

Stemming is a more aggressive approach where words are reduced to their root form by removing suffixes

```
# Stemming

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

# Defining function for stemming

def stem_tokens(tokens):
    stemmed_tokens = [stemmer.stem(token) for token in tokens]
    return stemmed_tokens
```

```
# Apply stemming
# Stemming is a more aggressive approach where words are reduced to their root form by removing suffixes

train_data['review'] = train_data['review'].apply(stem_tokens)
test_data['review'] = test_data['review'].apply(stem_tokens)
```

### Tokens post applying stemming:

```
# Review changes

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    [``, side, effect, take, combin, bystol, 5, mg...
1    [``, son, halfway, fourth, week, intuniv, beca...
2    [``, use, take, anoth, oral, contracept, 21, p...
3    [``, first, time, use, form, birth, control, 0...
4    [``, suboxon, complet, turn, life, around, fee...
Name: review, dtype: object
0    [``, side, effect, take, combin, bystol, 5, mg...
1    [``, son, halfway, fourth, week, intuniv, beca...
2    [``, use, take, anoth, oral, contracept, 21, p...
3    [``, first, time, use, form, birth, control, 0...
4    [``, suboxon, complet, turn, life, around, fee...
Name: review, dtype: object
```

**Removing HTML tags and special characters if any from the tokens:**

```
# Removing HTML tags and special characters if any from the tokens

import re

# Function for removing html tokens

def remove_html(text):
    text_without_html = re.sub(r'<.*?>', '', text)
    text_cleaned = re.sub(r'[^a-zA-Z0-9\s]', '', text_without_html)
    return text_cleaned

def remove_html_from_tokens(tokens):
    text = ' '.join(tokens)
    text_cleaned = remove_html(text)
    tokens_cleaned = text_cleaned.split()
    return tokens_cleaned
```

```
# Applying the above function

train_data['review'] = train_data['review'].apply(remove_html_from_tokens)
test_data['review'] = test_data['review'].apply(remove_html_from_tokens)
```

**Tokens post removal of HTML tags and special characters:**

```
# Review modifications

print(train_data['review'].head())
print(test_data['review'].head())
```

```
0    [side, effect, take, combin, bystol, 5, mg, fi...
1    [son, halfway, fourth, week, intuniv, becam, c...
2    [use, take, anoth, oral, contracept, 21, pill,...
3    [first, time, use, form, birth, control, 039, ...
4    [suboxon, complet, turn, life, around, feel, h...
Name: review, dtype: object
0    [side, effect, take, combin, bystol, 5, mg, fi...
1    [son, halfway, fourth, week, intuniv, becam, c...
2    [use, take, anoth, oral, contracept, 21, pill,...
3    [first, time, use, form, birth, control, 039, ...
4    [suboxon, complet, turn, life, around, feel, h...
Name: review, dtype: object
```

## Understanding frequency of words:

```
# Understanding frequency of words
# Word frequency analysis

from collections import Counter

# Function for word frequency
def word_frequency(text_data):
    # Flatten the tokens
    flattened_text = [word for sublist in text_data for word in sublist]
    # Counting frequency of each word
    word_freq = Counter(flattened_text)
    return word_freq
```

```
# Computing word frequency analysis

train_word_freq = word_frequency(train_data['review'])
test_word_freq = word_frequency(test_data['review'])
```

## Most common words and their frequencies:

```
# Print the most common words and their frequencies

print("Common words in training data:")
print(train_word_freq.most_common(10))
```

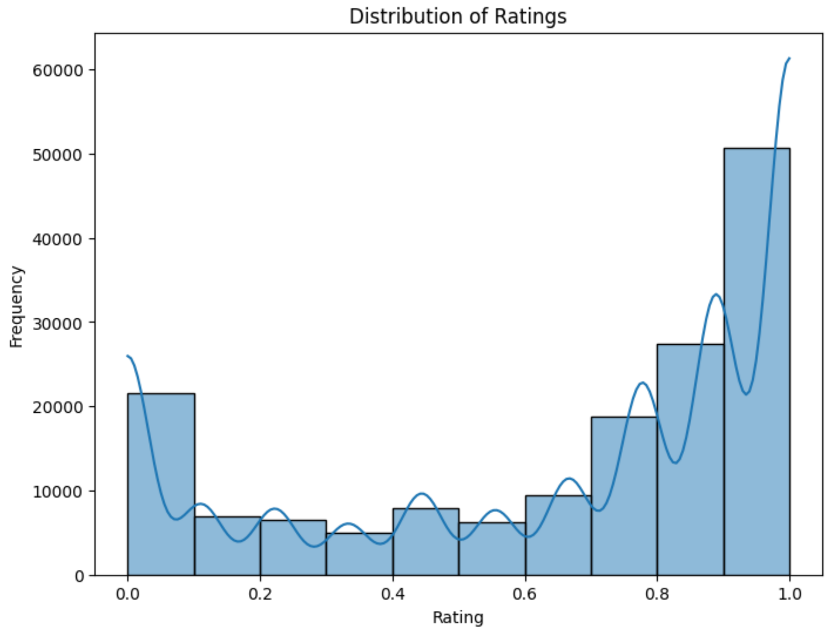
```
Common words in training data:
[('039', 260971), ('take', 97823), ('day', 95556), ('month', 68440), ('year', 65495), ('effect', 62818), ('work', 61997), ('get', 57942), ('week', 57611), ('start', 57014)]
```

```
print("Common words in testing data:")
print(test_word_freq.most_common(10))
```

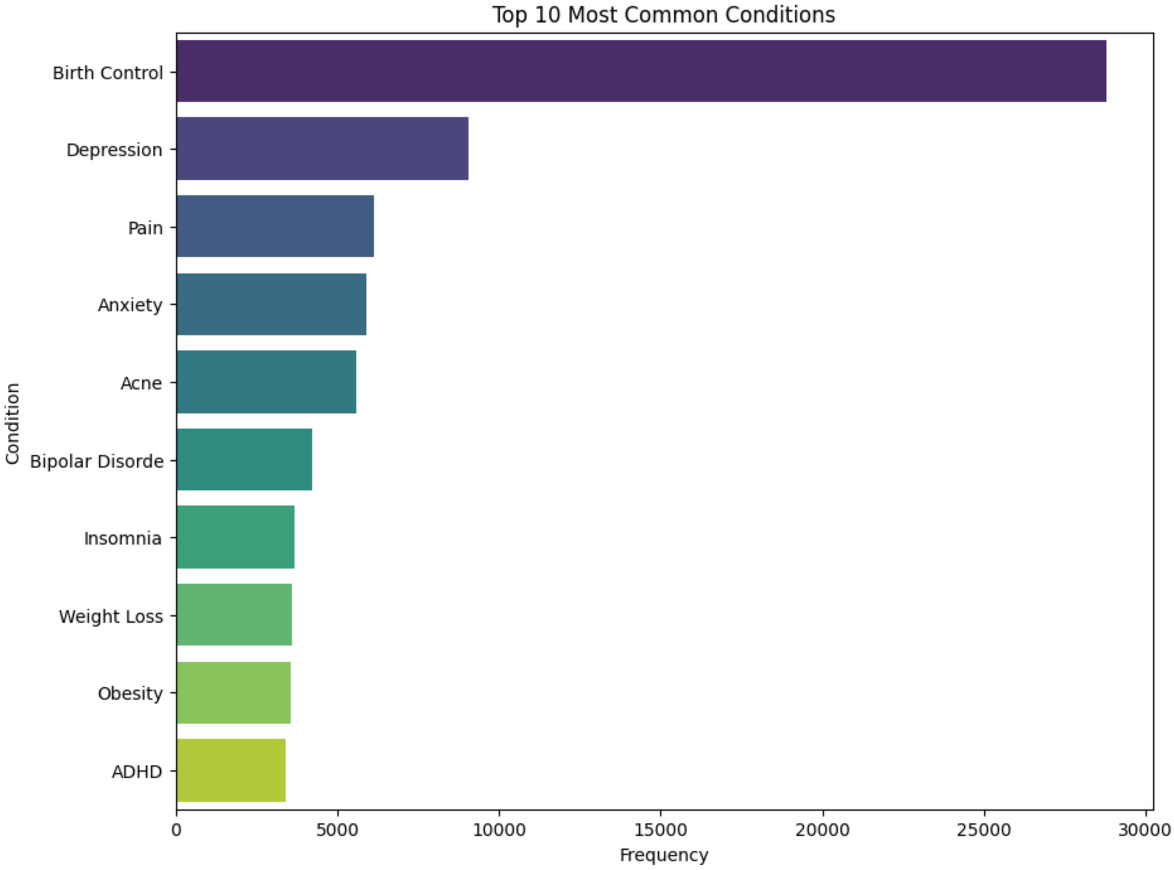
```
Common words in testing data:
[('039', 260971), ('take', 97823), ('day', 95556), ('month', 68440), ('year', 65495), ('effect', 62818), ('work', 61997), ('get', 57942), ('week', 57611), ('start', 57014)]
```

## Data Visualization:

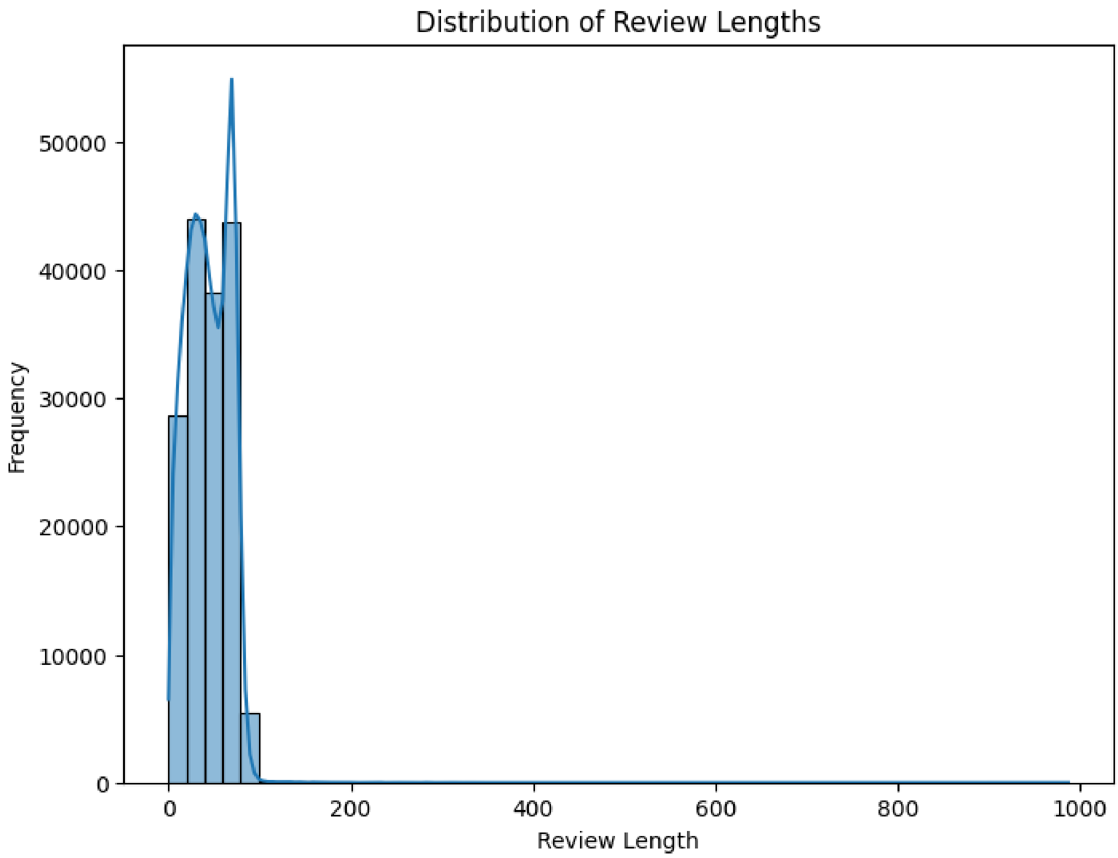
Distribution of Ratings:



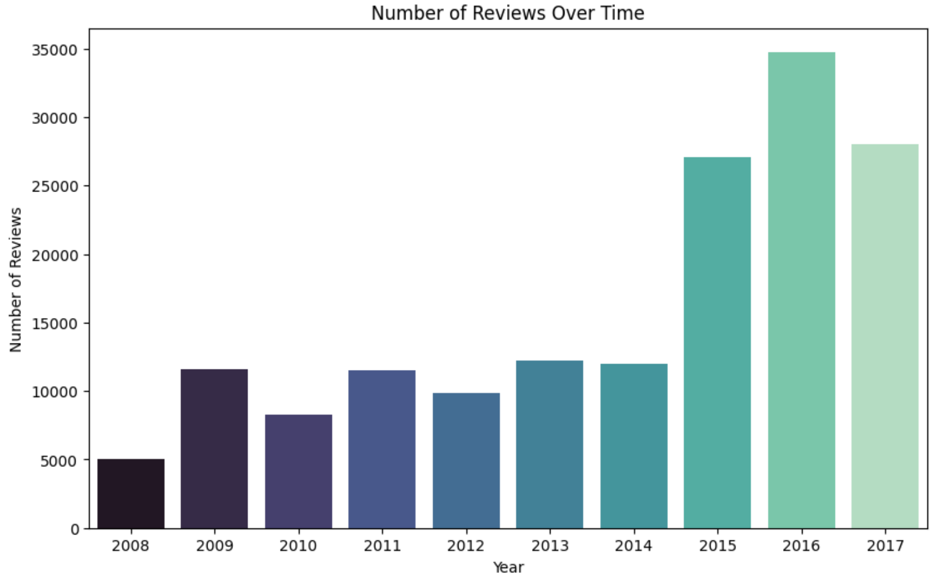
**Most common conditions mentioned in the reviews:**



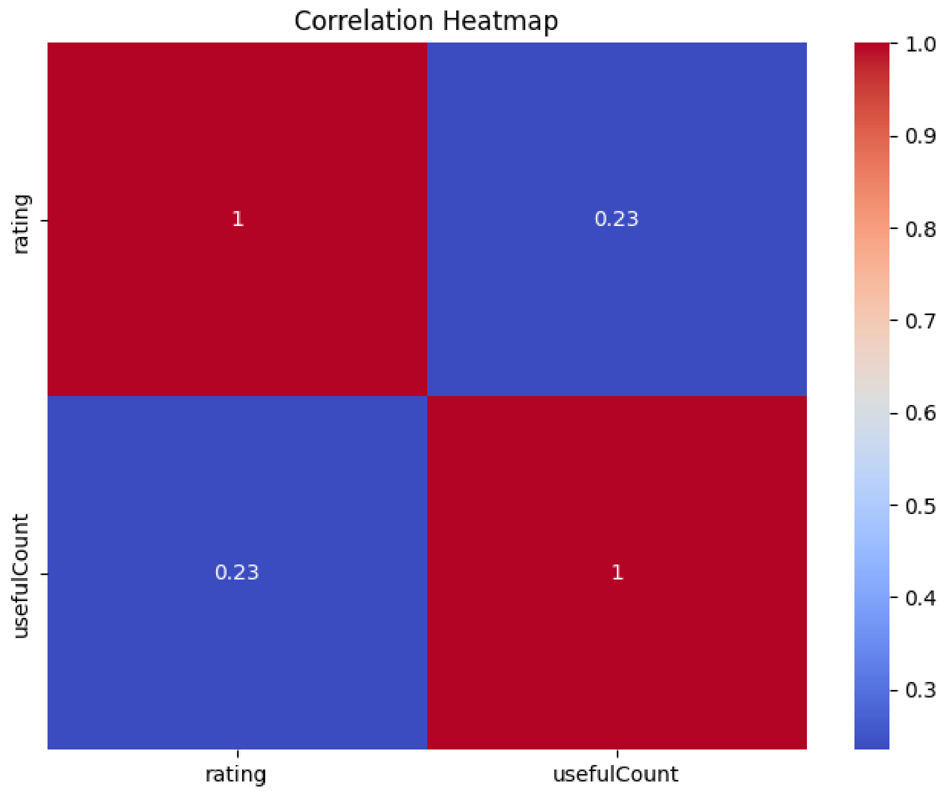

**Length of reviews:**



**Number of reviews over time:**




**Correlation Heat Map:**




Implementing analyze_sentiment function that performs sentiment analysis on each review and returns the sentiment category i.e. positive, negative or neutral

```
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

def analyze_sentiment(review):
    sentiment_scores = sid.polarity_scores(review)
    if sentiment_scores['compound'] >= 0.05:
        return 'positive'
    elif sentiment_scores['compound'] <= -0.05:
        return 'negative'
    else:
        return 'neutral'
```

### Apply sentiment analysis and creating a sentiment column:

```
# Apply sentiment analysis and creating a sentiment column

train_data['sentiment'] = train_data['review'].apply(lambda x: analyze_sentiment(' '.join(x)))
```

```
# On testing data
test_data['sentiment'] = test_data['review'].apply(lambda x: analyze_sentiment(' '.join(x)))
```

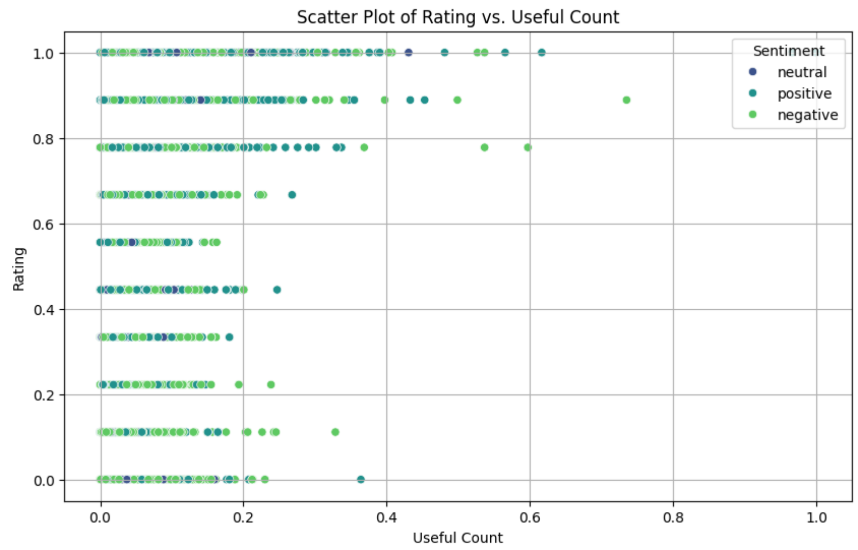
### Train Dataset:

```
                         drugName                     condition  \
0                       Valsartan  Left Ventricular Dysfunction
1                      Guanfacine                          ADHD
2                          Lybrel                 Birth Control
3                      Ortho Evra                 Birth Control
4        Buprenorphine / naloxone            Opiate Dependence
...                           ...                           ...
161292                    Campral             Alcohol Dependence
161293               Metoclopramide               Nausea/Vomiting
161294                    Orencia          Rheumatoid Arthritis
161295          Thyroid desiccated            Underactive Thyroid
161296                Lubiprostone          Constipation, Chronic

                                                   review    rating  \
0       [side, effect, take, combin, bystol, 5, mg, fi...  0.888889
1       [son, halfway, fourth, week, intuniv, becam, c...  0.777778
2       [use, take, anoth, oral, contracept, 21, pill,...  0.444444
3       [first, time, use, form, birth, control, 039, ...  0.777778
4       [suboxon, complet, turn, life, around, feel, h...  0.888889
...                                                   ...       ...
161292  [wrote, first, report, midoctob, 2014, alcohol...  1.000000
161293  [given, iv, surgey, immedi, becam, anxiou, cou...  0.000000
161294  [limit, improv, 4, month, develop, bad, rash, ...  0.111111
161295  [039, thyroid, medic, 49, year, spent, first, ...  1.000000
161296  [039, chronic, constip, adult, life, tri, linz...  0.888889

             date  usefulCount  review_length  year sentiment
0       2012-05-20     0.020914              9  2012   neutral
1       2010-04-27     0.148722             66  2010  positive
2       2009-12-14     0.013168             73  2009  positive
3       2015-11-03     0.007746             43  2015  positive
4       2016-11-27     0.028660             62  2016  positive
...            ...          ...            ...   ...       ...
161292 2015-05-31     0.096824             60  2015  positive
161293 2011-11-01     0.026336             25  2011  negative
161294 2014-03-15     0.027111             11  2014  negative
161295 2015-09-19     0.061193             78  2015  positive
161296 2014-12-13     0.089853             35  2014  negative

[160398 rows x 9 columns]
```

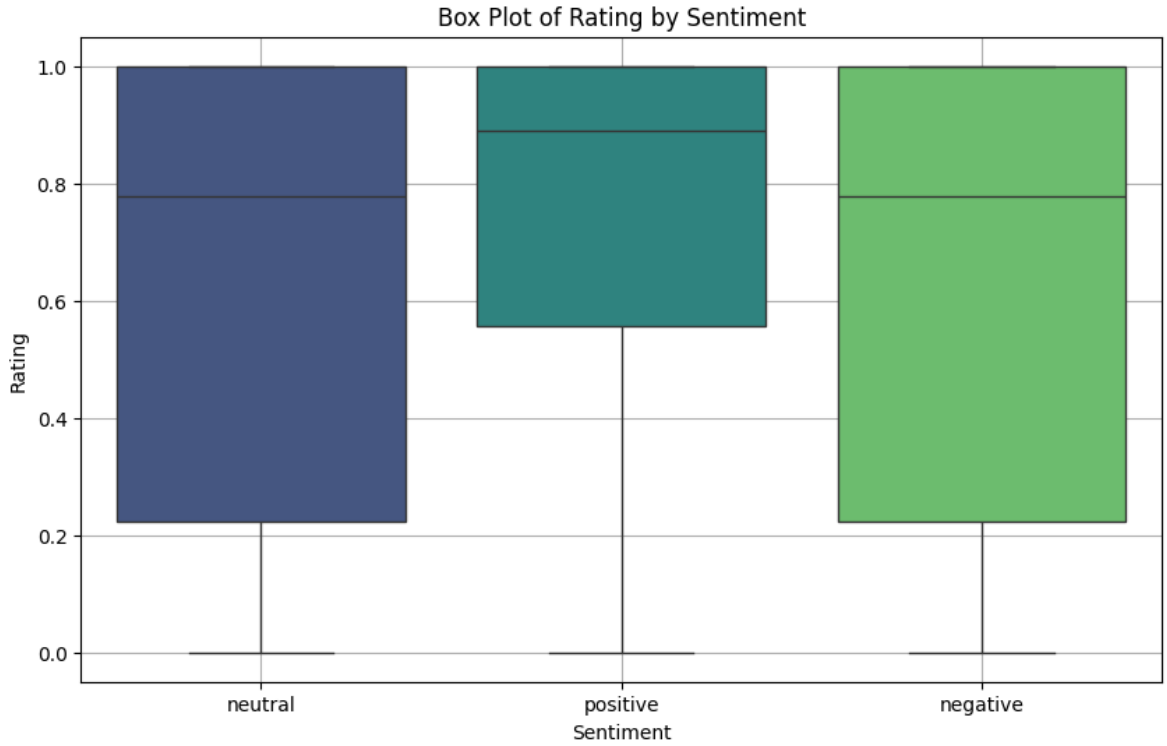
## Test Dataset:

```
                        drugName                     condition  \
0                      Valsartan  Left Ventricular Dysfunction
1                     Guanfacine                          ADHD
2                         Lybrel                 Birth Control
3                     Ortho Evra                 Birth Control
4       Buprenorphine / naloxone            Opiate Dependence
...                          ...                           ...
161292                   Campral             Alcohol Dependence
161293            Metoclopramide               Nausea/Vomiting
161294                   Orencia          Rheumatoid Arthritis
161295        Thyroid desiccated           Underactive Thyroid
161296               Lubiprostone          Constipation, Chronic

                                                    review    rating  \
0       [side, effect, take, combin, bystol, 5, mg, fi...  0.888889
1       [son, halfway, fourth, week, intuniv, becam, c...  0.777778
2       [use, take, anoth, oral, contracept, 21, pill,...  0.444444
3       [first, time, use, form, birth, control, 039, ...  0.777778
4       [suboxon, complet, turn, life, around, feel, h...  0.888889
...                                                   ...       ...
161292  [wrote, first, report, midoctob, 2014, alcohol...  1.000000
161293  [given, iv, surgey, immedi, becam, anxiou, cou...  0.000000
161294  [limit, improv, 4, month, develop, bad, rash, ...  0.111111
161295  [039, thyroid, medic, 49, year, spent, first, ...  1.000000
161296  [039, chronic, constip, adult, life, tri, linz...  0.888889

              date  usefulCount sentiment
0       2012-05-20     0.020914   neutral
1       2010-04-27     0.148722  positive
2       2009-12-14     0.013168  positive
3       2015-11-03     0.007746  positive
4       2016-11-27     0.028660  positive
...            ...          ...       ...
161292 2015-05-31     0.096824  positive
161293 2011-11-01     0.026336  negative
161294 2014-03-15     0.027111  negative
161295 2015-09-19     0.061193  positive
161296 2014-12-13     0.089853  negative

[160398 rows x 7 columns]
```

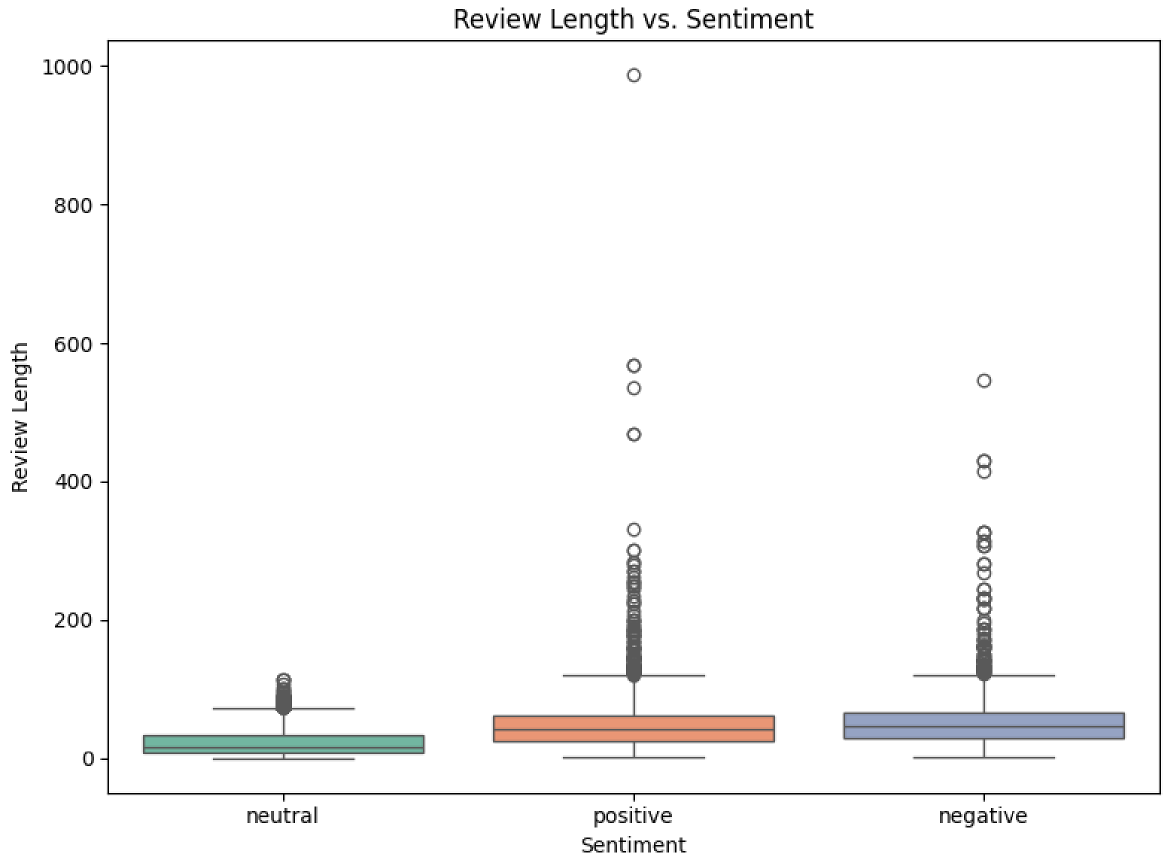
## Scatter Plot of Rating vs. Useful Count:



**Box Plot of Rating by Sentiment:**

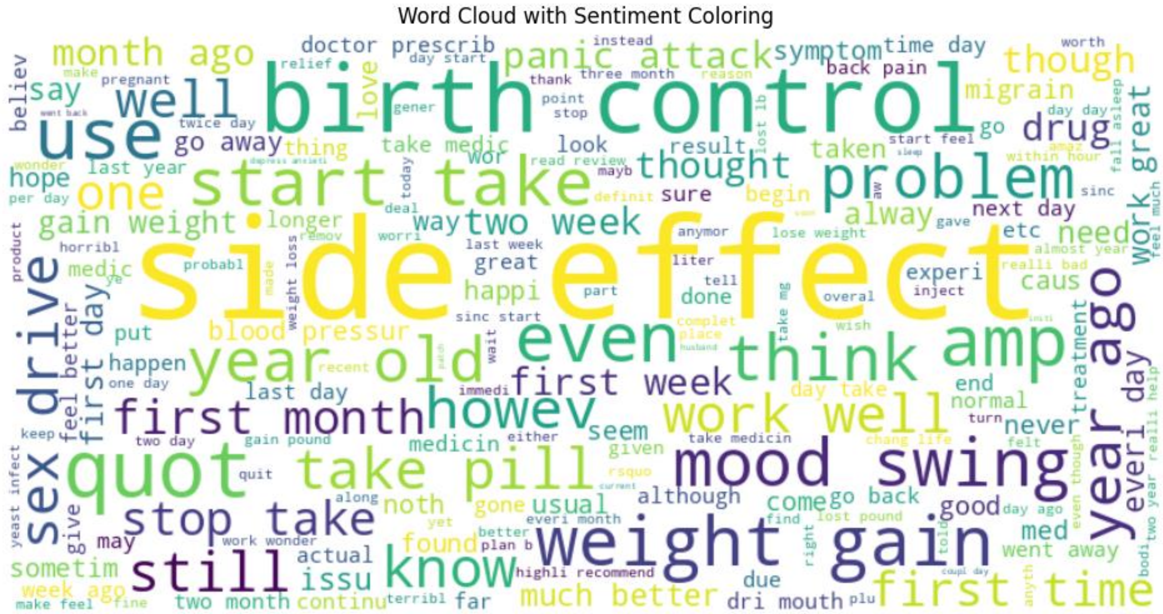



**Review Length vs. Sentiment:**



**Word frequency v/s sentiment:**


**Word Cloud with Sentiment Coloring:**




**Vectorization:**

Process of converting text data into numerical vectors that can be used as input for deep learning models

```
# Vectorization
# Process of converting text data into numerical vectors that can be used as input for deep learning models
from sklearn.feature_extraction.text import TfidfVectorizer

# Vectorization using TF-IDF method
tfidf_vectorizer = TfidfVectorizer()

# Converting tokens into a string for reviews
train_reviews = train_data['review'].apply(lambda x: ' '.join(x))
test_reviews = test_data['review'].apply(lambda x: ' '.join(x))

# Fit TF-IDF vectorizer on training data for transformation
train_tfidf = tfidf_vectorizer.fit_transform(train_reviews)
test_tfidf = tfidf_vectorizer.fit_transform(test_reviews)
```

**Shape of the TF-IDF matrices:**

```
# Shape of the TF-IDF matrices
print("Shape of TF-IDF matrix for training data:", train_tfidf.shape)
print("Shape of TF-IDF matrix for testing data:", test_tfidf.shape)
```

```
Shape of TF-IDF matrix for training data: (160398, 52896)
Shape of TF-IDF matrix for testing data: (160398, 52896)
```

**Truncate Sequence:**

```
# Truncate Sequences

import tensorflow as tf

max_sequence_length = 100

# Truncate sequences to a maximum length of 100
def truncate_sequences(sequences, max_length):
    truncated_sequences = []
    for sequence in sequences:
        truncated_sequence = sequence[:max_length]
        truncated_sequences.append(truncated_sequence)
    return truncated_sequences
```

```
X_train_truncated = truncate_sequences(train_tfidf.toarray(), max_sequence_length)
X_test_truncated = truncate_sequences(test_tfidf.toarray(), max_sequence_length)
```

**Encode Labels:**

```
# Function to encode sentiment labels

from sklearn.preprocessing import LabelEncoder

def encode_sentiment_labels(labels):
    label_encoder = LabelEncoder()
    encoded_labels = label_encoder.fit_transform(labels)
    return encoded_labels, label_encoder.classes_
```

**Train Test Validation Split and truncate sequences into tensors:**

```
# Split the data into training and testing sets
from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(X_train_truncated, train_labels_encoded, test_size=0.2, random_state=42)
```

```
# Converting truncated sequences into tensors
import torch

X_train_tensor = torch.tensor(X_train)
X_val_tensor = torch.tensor(X_val)
X_test_tensor = torch.tensor(X_test_truncated)

y_train_tensor = torch.tensor(y_train)
y_val_tensor = torch.tensor(y_val)
y_test_tensor = torch.tensor(test_labels_encoded)
```

**Tensor Shape:**

```
# Shapes of the tensors
print("Shape of X_train_tensor:", X_train_tensor.shape)
print("Shape of X_val_tensor:", X_val_tensor.shape)
print("Shape of X_test_tensor:", X_test_tensor.shape)
print("Shape of y_train_tensor:", y_train_tensor.shape)
print("Shape of y_val_tensor:", y_val_tensor.shape)
print("Shape of y_test_tensor:", y_test_tensor.shape)
```

```
Shape of X_train_tensor: torch.Size([128318, 100])
Shape of X_val_tensor: torch.Size([32080, 100])
Shape of X_test_tensor: torch.Size([53471, 100])
Shape of y_train_tensor: torch.Size([128318])
Shape of y_val_tensor: torch.Size([32080])
Shape of y_test_tensor: torch.Size([53471])
```

**Model Architecture:**

```
import torch
import torch.nn as nn
import torch.optim as optim

# Model Architecture
class SentimentRNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(SentimentRNN, self).__init__()
        self.hidden_size = hidden_size
        self.rnn = nn.RNN(input_size, hidden_size, batch_first=True)
        self.fc = nn.Linear(hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, x):
        out, _ = self.rnn(x)
        out = self.fc(out[:, -1, :])
        out = self.softmax(out)
        return out
```

**Parameters, loss function, optimizer and model initialization:**

```
# Values for the model
input_size = len(tfidf_vectorizer.vocabulary_)
hidden_size = 128
output_size = 3
```

```
# Model Initiaization

sentiment_model = SentimentRNN(input_size, hidden_size, output_size)
```

```
# Loss function
criterion = nn.CrossEntropyLoss()

# Optimizer
optimizer = optim.Adam(sentiment_model.parameters(), lr=0.001)
```

**Train Data loaders and model training:**

```
import torch
from torch.utils.data import DataLoader, TensorDataset

X_train_tensor = torch.tensor(X_train_truncated, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train, dtype=torch.long)

train_dataset = TensorDataset(X_train_tensor, y_train_tensor)

batch_size = 64

train_loader = DataLoader(dataset=train_dataset, batch_size=batch_size, shuffle=True)
```

```
# Training the model
def train_model(model, criterion, optimizer, train_loader, num_epochs=10):
    model.train()
    for epoch in range(num_epochs):
        running_loss = 0.0
        for inputs, labels in train_loader:
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()
            running_loss += loss.item()
        print(f"Epoch {epoch+1}, Loss: {running_loss/len(train_loader)}")
```