Text Analytics and Natural Language Processing
Project Report

Team Members
Anuj Anand
Ashish Juneja

Link to GitHub: https://github.com/anujanand6/Sentiment-Analysis-of-Drug-Reviews

Problem Statement:

Patients give review for a drug based on the efficacy and their experience using the drug

The objective is two folds:

1. Looking at drugs that are being used in the market for the condition, identify the most favorable and least favorable drug. Also understand what works for patients who rate highly about a drug
2. Assign a sentiment (positive, negative) to an unlabeled review using a labelled corpus

Dataset description

Drugs Review Dataset (link to dataset)

- The data is a collection of patients' reviews of drugs.

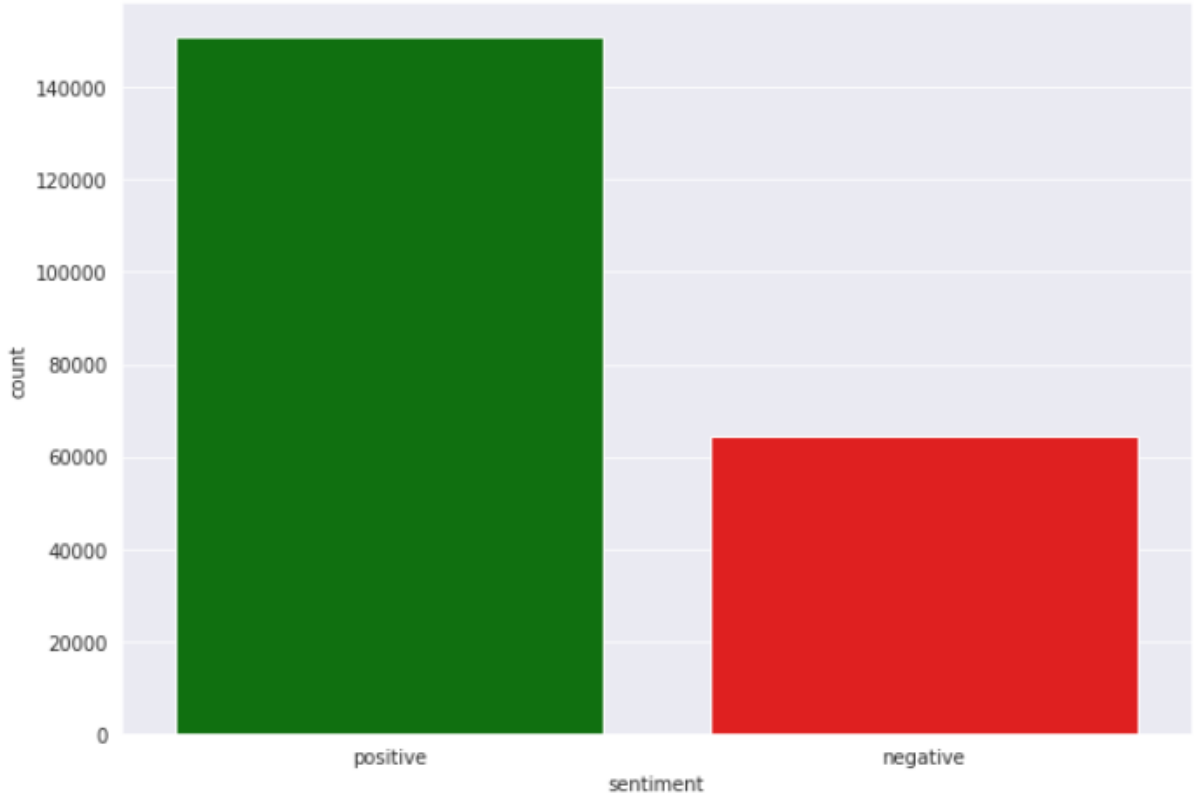
- Features:
    - Unique ID: a unique ID for each drug
    - Drug Name: name of the drug
    - Condition: medical condition that the drug is used to treat
    - Review: patients' review of the drug
    - Rating: overall scoring of the drug on a scale of 1 to 10
    - Date: the date on which the review was recorded
    - usefulCount : this is a count of how many people found this review to be useful

- The dataset has already been split into train and test, each with 161k and 53k reviews, respectively.

- The rating will be used to label each review as positive, negative

Exploratory Data Analysis

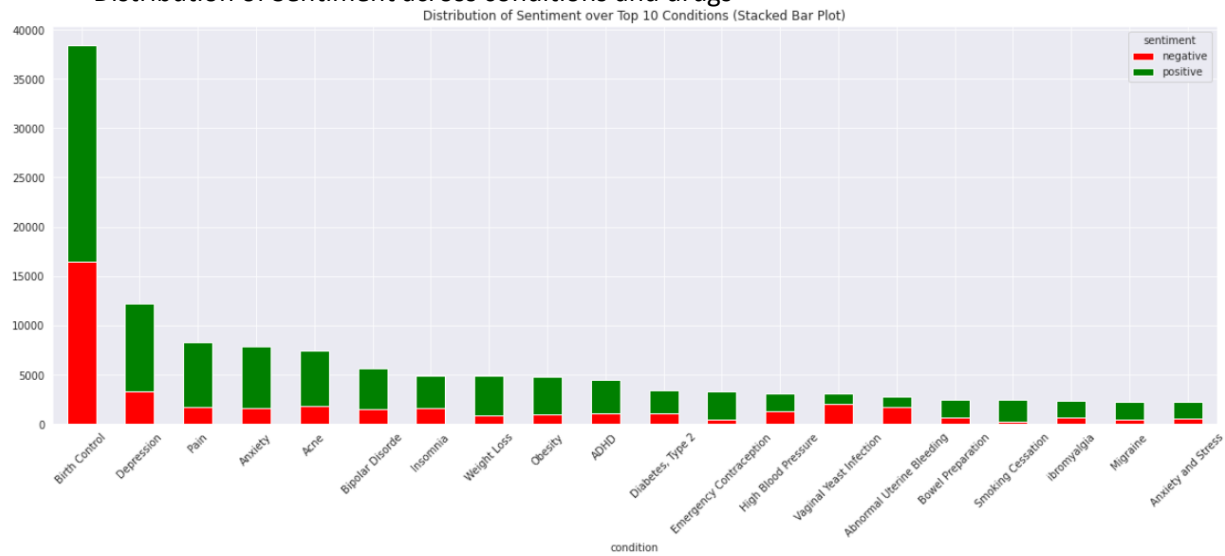
- Size of corpus - ~215k
- Records with missing values - ~1200
- # of conditions – 917
- Distribution of Sentiment Label



This is a clear case of class imbalance.

At this point, under sampling the positive reviews is an option, however we want to move ahead with first selecting the top 'N' conditions for the purpose of this analysis (you'll see further in the code) and then again check for the pos/neg distribution.

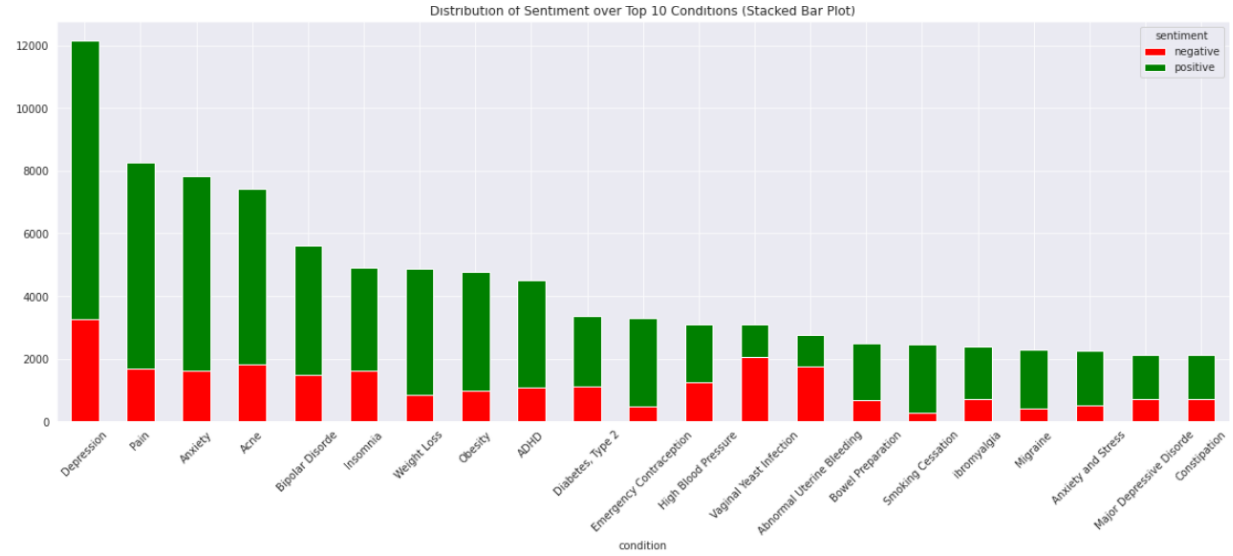
- Distribution of Sentiment across conditions and drugs



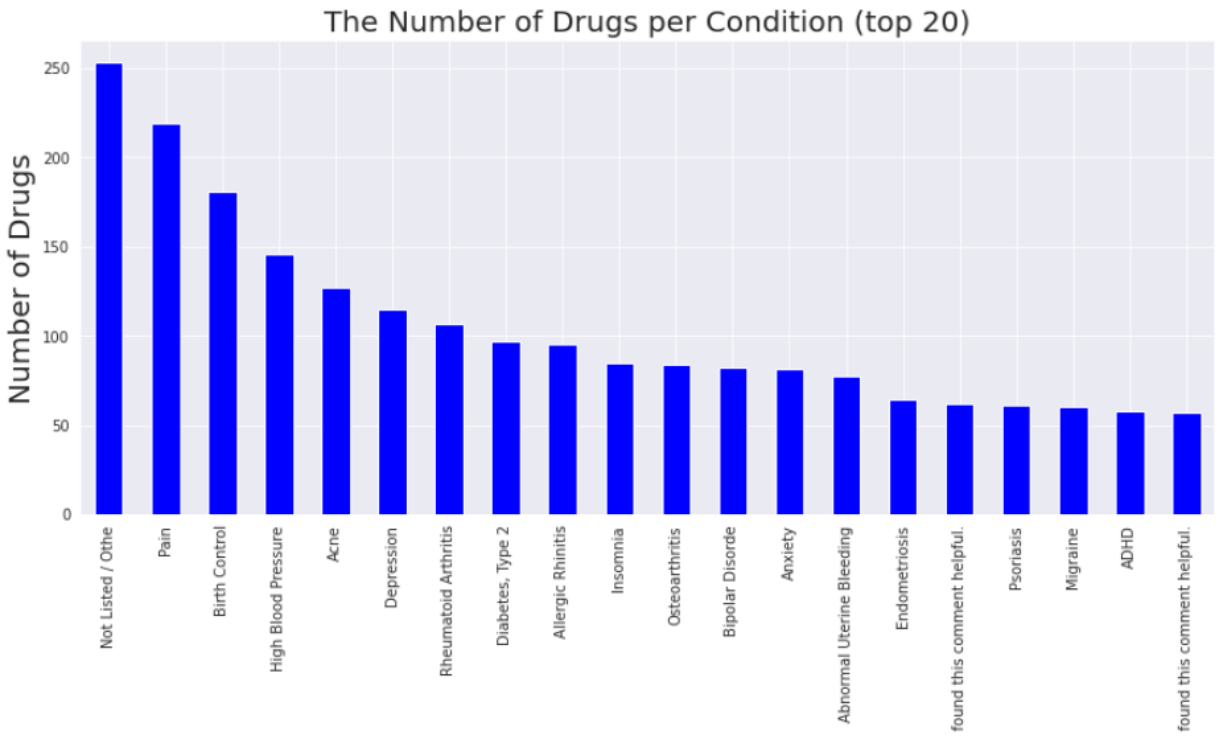
In almost all conditions there seem to be more positive reviews than negative ones.

Birth control seems to have the highest number of reviews followed by depression. Including these conditions to the dataset might induce some bias which we will tackle later on.

Let's visualize the above after taking out birth control.



- # of drugs per condition
  Understanding the distribution of the drugs among the conditions is also important for our use case so that we do not end up taking conditions with too less drugs
  As we would expect, generic conditions like pain and birth control seem to have a high number of reviews.



Data Preprocessing

- Split the data into positive and negative
- Remove default stop words
- Dropping reviews containing `</span>` and `"Not Listed / Othe"`
- Dropping 'Date' and 'Usefulcount' column

Data Reduction

There are three ways we could do this:

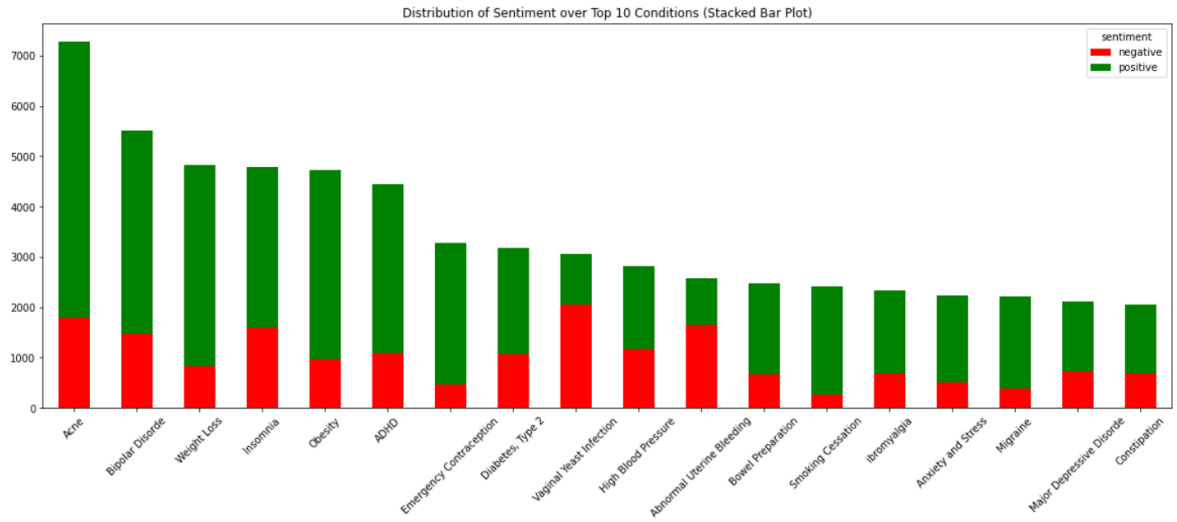
- Random sampling
- K means clustering and then choosing a subset of data from each clusters
- Reduce the number of conditions

Unfortuntely due to the huge size of our data, we could not implement K-
Means clustering and with random sampling we may lose some important information. So, we decided t o go ahead with the third option of reducing the number of conditions.

Looking at the top conditons with most reviews, the most intuitive choice would be to go with top 'N' co nditions.
However, we made a few decisions to support the choice of going with the below and here's why -

- Removed conditions which are very generic (Ex: Cough, Fever) even though they had more reviw es than the below 4
- To ensure each condition we select has almost comparable number of reviews. (Ex: removed bir th control since it had too high number of reviews)

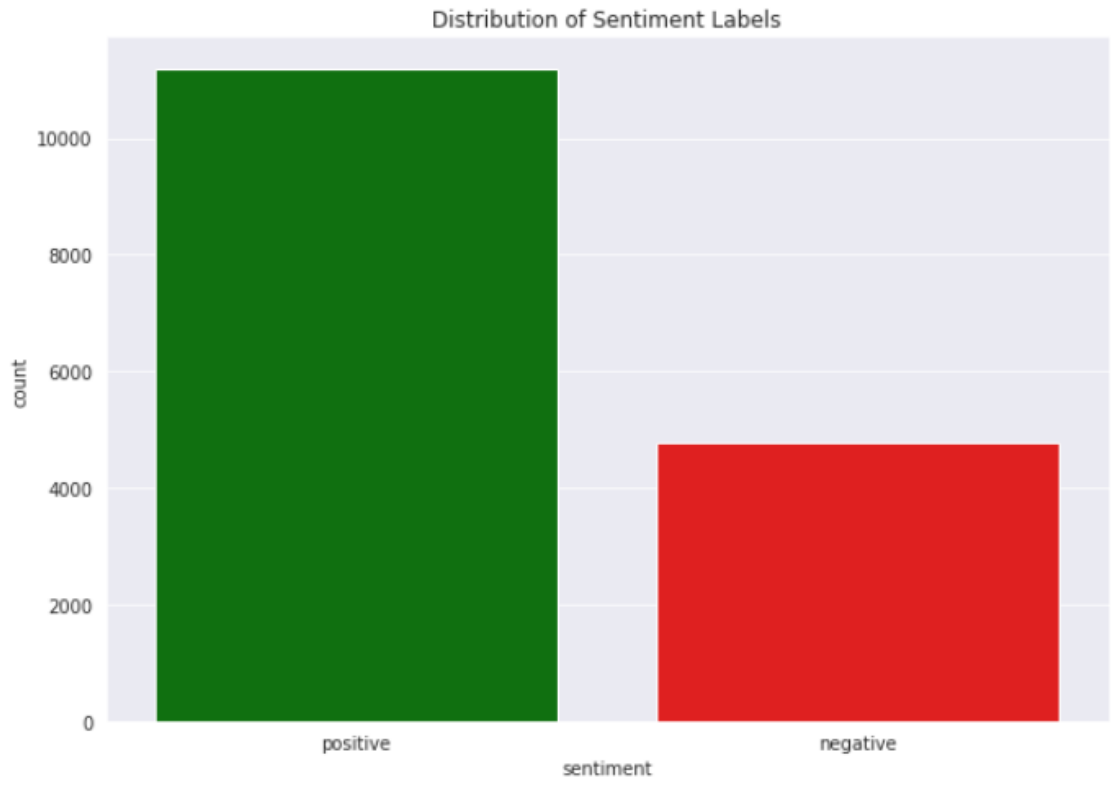
	# Reviews	# Conditions
Total reviews	~210k	917
Removing generic conditions	~150k	912
Removing conditions with <2k reviews	~63k	18
Removing drugs < 10 reviews	~62k	18
Keeping top 4 that make intuitive sense	~15k	4

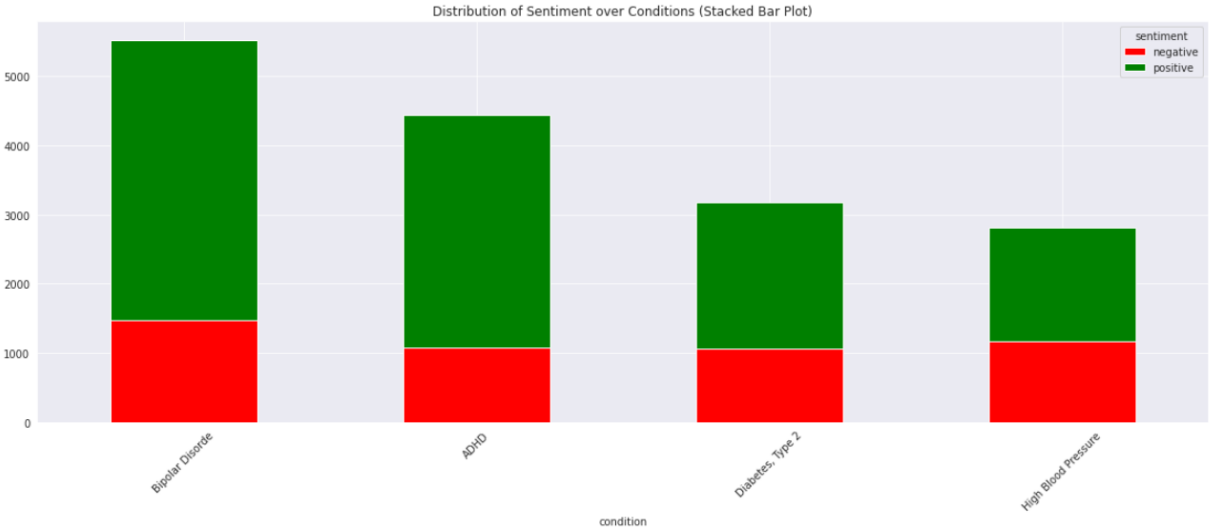


Conditions selected on intuitive sense from the above plot:

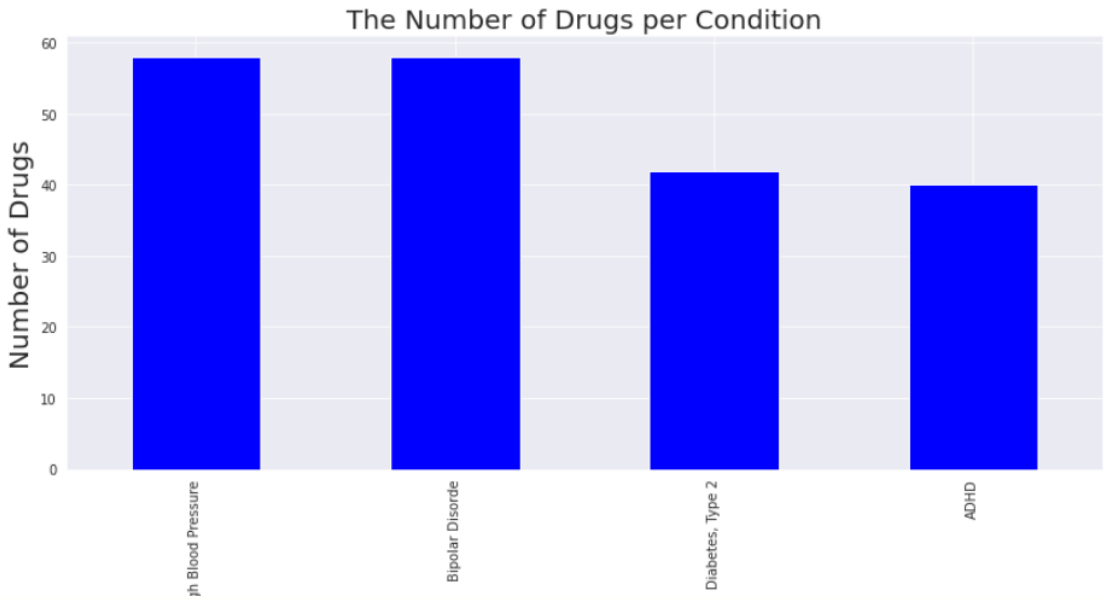
- Bipolar Disorder
- ADHD
- Diabetes, Type 2
- High Blood Pressure

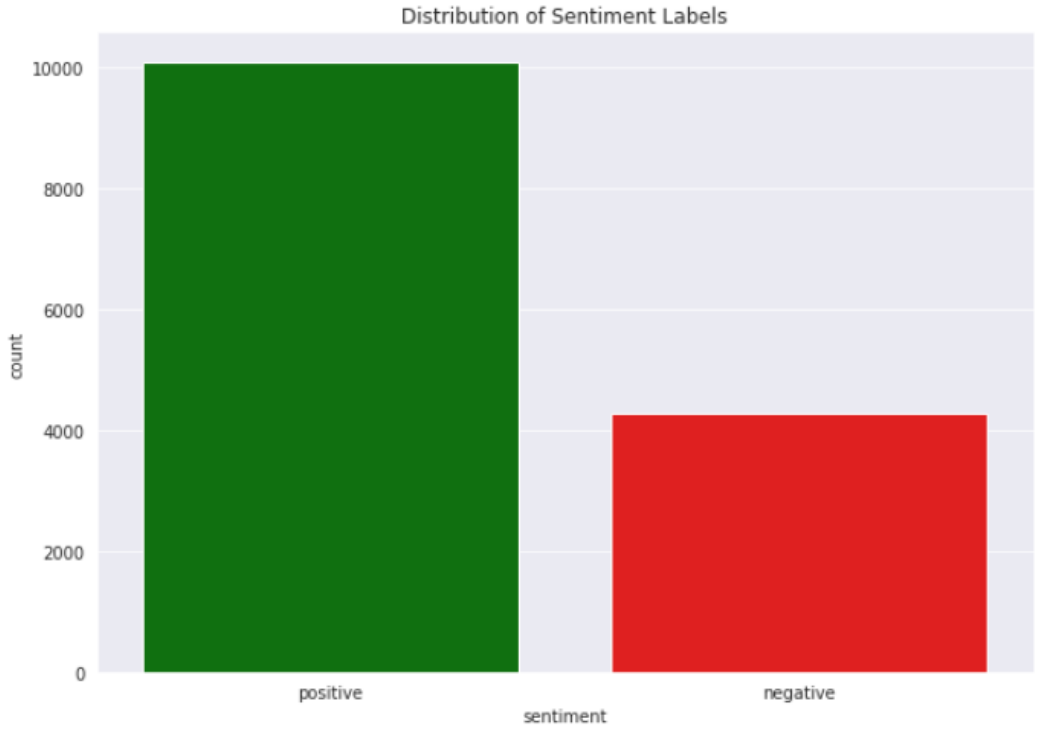
After reducing the dataset, the sentiment class has a distribution like the larger corpus





All the conditions have almost the same number of reviews and unique drugs.





Design Tokenization Strategy and Word Count utilities

- Tokenization
- Add stop words to the default list/remove stop words from the default list

- Though removing the default stop words, the top words do not really convey much information. Hence, let's add more stopwords to the model like mg, feel, take, day, years, etc.
- Also, some words like "aren't", "couldn't", "no", "not", etc. convey information related to the negative sentiment. So, it's best if we do not remove those to help the classification perform better.

- Count Vectorization & TF-IDF

```
Top words present in the whole reviews corpus:
[('weight', 3065), ('life', 2911), ('better', 2431), ('drug', 2418), ('dose', 2233), ('good', 2231), ('great', 2205), ('night', 2082), ('sleep', 2077), ('adhd', 2052)]

Top words present in the positive review corpus:
[('life', 2544), ('weight', 2328), ('better', 1963), ('great', 1892), ('good', 1780), ('adhd', 1667), ('dose', 1580), ('depression', 1496), ('sleep', 1466), ('night', 1449)]

Top words present in the negative review corpus:
[('drug', 982), ('felt', 773), ('weight', 737), ('pain', 678), ('went', 657), ('dose', 653), ('bad', 649), ('night', 633), ('stopped', 616), ('sleep', 611)]
```

The distinction in words seems a bit more obvious like:

great, good, better, life in the positive corpus.

pain, bad, stopped, severe, etc. in the negative corpus.

Most favorable/Least Favorable drug within each Condition

The intent is to identify the most favorable product and least favorable product in each condition.

For the purpose of this analysis, we designed a 'favorabilty' score i.e. defined as the difference of positive reviews and negative reviews for each drug within each condition.

Why we chose to identify most favorable and least favorable drug for each conditions Vs combined for all conditions? - Because each drug works differently for different conditions. The same drug could use to treat a condition but could have side effects which cause symptoms of a different condition. So the scores of a drug treated for one condition cannot be compared with those of a different condition.

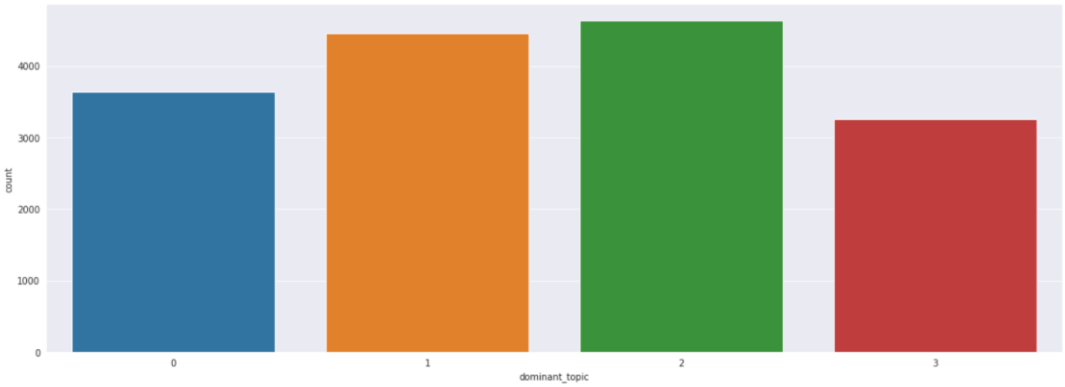
Condition	Most favorable product	Least favorable product	**Positive characteristic of the drug**
Bipolar Disorde	Lamotrigine	Celexa	Looks like the positive aspect mood alleviator
Diabetes, Type 2	Liraglutide	Tradjenta	Helps with weight loss
High Blood Pressure	Olmesartan	Amlodipine	No clear trend
ADHD	Lisdexamfetamine	Mydayis	Changed Life of the patient

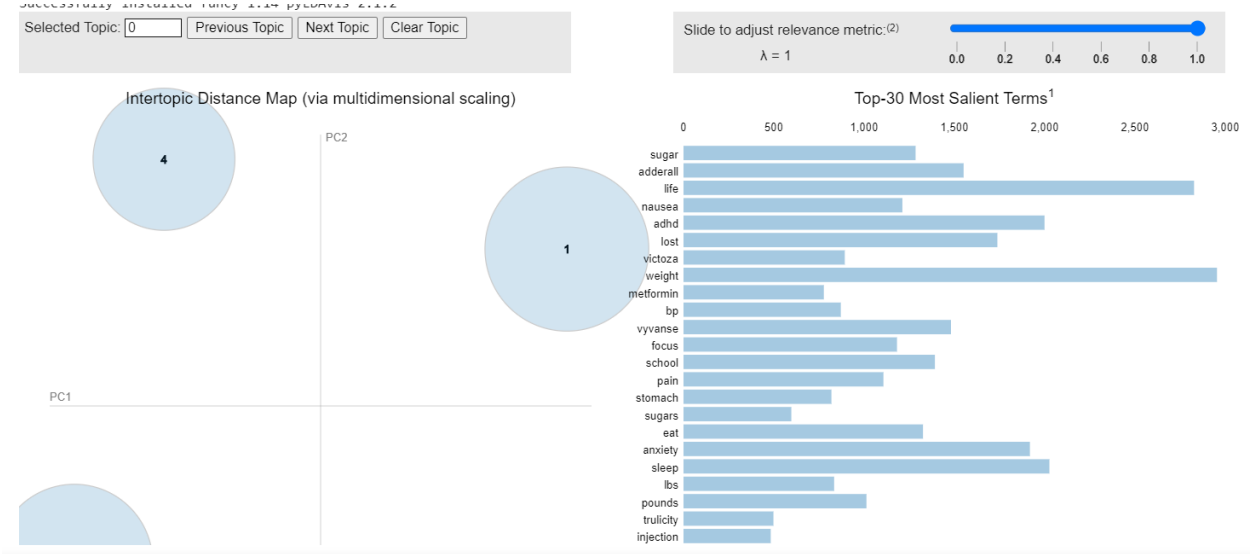
Topic Modeling – LDA and NMF

Since we have 4 conditions (which in a sense are the main themes) in our dataset, we thought it would be interesting to see the topic model results with number of components as 4.

Visualize topics from LDA - Using pyLDAvis

We see topics 1 and 2 are more dominant





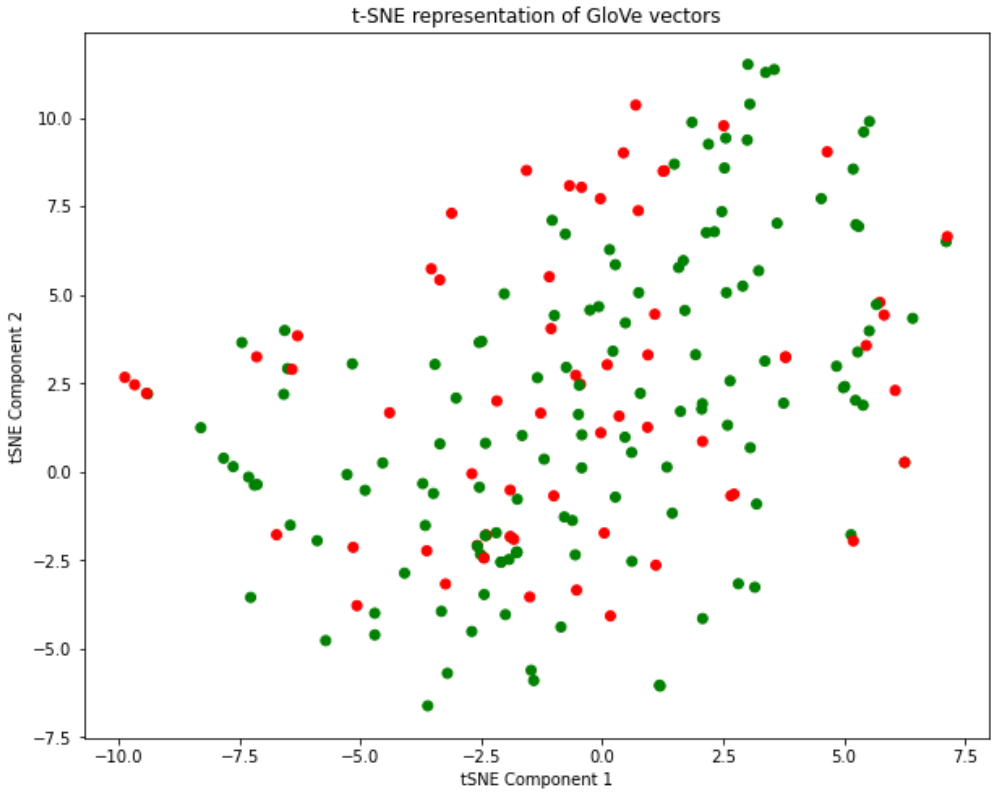
Cosine Similarity

Counts and TF-IDF do a better job at separating reviews from the two classes

Embedding

GloVe Embedding

Plotting the GloVe vectors to see if there are any visible clusters.
Unfortunately, this did not provide us with any good information.



ElMo

We tried implementing ELMo and it worked fine on small data but when we ran it on our reduced datatset, it did not stop running and we ran it for more than 2 hours. Hence, we omitted this from our project due to lack of computational power.

Unfortunately, we could not code it in such a way it uses the GPU.

DistilBERT

Due to the domain-specific nature of our dataset, it does not really make sense to use embedding methods like GloVe which has the vectors for only a general corpus, with neither context nor domain knowledge.

We wanted to tackle the issue of 'no context', using the famous DistilBERT model.
The GPU kept crashing even when ran BERT on our corpus

So we had to reduce two things to ensure it runs smoothly: Batch size, and Maximum length of each review.

After running multiple iterations and trying different combinations, this is the only one that worked. And anything above this, crashed the system.

1. Batch size = 5
2. Maximum length of each review = 200

We do lose some information when we truncate the reviews to 200, but since majority of the reviews have around 250 tokens, the loss is not too much.

BioBERT

Now that we tried the BERT model which is great at including the context of the word while embedding it, we wanted to go a step further and include some domain knowledge as well i.e. biomedical terms.

It is very similar to BERT, but as you might have guessed, this is used specifically for biomedical data which is exactly what we were looking for. BioBERT is the first domain-specific BERT based model pre-trained on biomedical corpora for 23 days on eight NVIDIAV100 GPUs. Though it's mainly used for NER, Relation extraction and Question & Answering, it can be used for word embeddings as well.

The specific model that we are using is 'biobert_v1.1_pubmed' which has been train on Wikipedia, Books, and PubMed articles.
For more information, check out the BioBERT paper written by the researchers of Korea University & Clova AI research group based in Korea.
Note: Though the code is exactly the same as BERT, it took quite sometime to get hold of the models and understand how they work. This is mainly because BioBERT is not very widely used.

Similar to BERT, BioBERT also crashed when we ran it on our dataset. Unfortunately, BioBERT did not run with the same configurations as BERT. We had to reduce the batch size and maximum length even further.

This is the configuration that worked,

1. Batch Size = 1
2. Maximum length = 125

We end up losing more information than we did with BERT. (This will be seen in the classification section.)

Classification Models

Note: We implemented the above steps for all the types in embedding methods that we have done so far. Finally, we run **only** the best model on the test data set to obtain the final accuracy.

Like we saw earlier in the EDA, the sentiment labels are highly imbalanced and this is something that should be taken care of to avoid any bias and give a more realistic accuracy. Also, it's more vital that the model classifies a negative review as negative, than the vice versa.

We surely do not want someone taking a drug based on a review labelled as positive when in fact it's negative. There could be a lot of bad consequences.
We took just one example to show you how we handled this issue

Linear SVC

Steps involved in classification are as follows:

1. Adding and Removing stopwords from the default stopwords list:
2. Handle the imbalance in the class labels.
3. Choose between spaCy's base English model and scispaCy's model.
4. Classification using Linear SVC, RBF SVC and Gaussian Naive Bayes (with train and validation data only)

Class Imbalance

Like we saw earlier in the EDA, the sentiment labels are highly imbalanced and this is something that should be taken care of to avoid any bias and give a more realistic accuracy. Also, it's more vital that the model classifies a negative review as negative, than the vice versa.

We surely do not want someone taking a drug based on a review labelled as positive when in fact it's negative. There could be a lot of bad consequences.
We took just one example to show you how we handled this issue.

```
Accuracy using Counts with Linear SVC model:
Train accuracy :              0.988
Validation accuracy :         0.862
Sensitivity :                 0.757
Specificity :                 0.908


Confusion Matrix :
 [[ 879  282]
 [ 247 2433]]

Precision :                   0.781
Negative Predictive Value :   0.896
False Positive Rate :         0.092
False Negative Rate :         0.243
False Discovery Rate :        0.219
Overall Accuracy :            0.862
```

As you can see, though the overall accuracy is high, the sensitivity/true negative rate is much lower than the specificity/true positive rate with a difference of 15%. This is obviously because of the imbalance.
Like we mentioned earlier, this is not good! But fortunately, the SVC model has a parameter called ***'class weights'*** which penalizes the majority class more than the minority class. And setting this parameter to 'balanced', the weights are chosen by the model automatically.

Let's run the above model again, but with 'balanced' class weights.

```
Accuracy using Counts with Linear SVC model:
Train accuracy :              0.985
Validation accuracy :         0.863
Sensitivity :                 0.81
Specificity :                 0.885


Confusion Matrix :
 [[ 940  221]
 [ 307 2373]]

Precision :                   0.754
Negative Predictive Value :   0.915
False Positive Rate :         0.115
False Negative Rate :         0.19
False Discovery Rate :        0.246
Overall Accuracy :            0.863
```

Now this is much better! The true negative rate seems to still be lower than the true positive rate, but at least we can be sure that it's not because of the imbalance in the dataset. For rest of the modelling, we decided to go ahead with the class weights as balanced.

Choosing Spacy Model

We had to choose from three choices:

1. spaCy's Base English model
2. sciSpacy's Disease-Chemical interaction model without replacing entities.
3. sciSpacy's Disease-Chemical interaction model after replacing entities.

All three performed almost the same, the third one performing just a tab bit worse. But we still decided to go head with the sciSpacy's Disease-Chemical interaction model after replacing entities because replacing the entities significantly reduced the feature space to about 8000 from about 13,000 with the base english model to 8350 with not much of a drop in accuracy.

Below are the results for the same running them on 4 embedding types:
Using Spacy's Base English model and Linear SVC:

```
Accuracy using Counts with Base English model:
Train accuracy: 0.985145739910314
Validation accuracy: 0.8498978897208985


Accuracy using TF-IDF with Base English model:
Train accuracy: 0.9331558295964125
Validation accuracy: 0.8464942137508509


Accuracy using NMF with Base English model:
Train accuracy: 0.703335201793722
Validation accuracy: 0.7066031313818925


Accuracy using LDA with Base English model:
Train accuracy: 0.703335201793722
Validation accuracy: 0.7066031313818925
```

Using Scispacy model with entities:

```
Accuracy using Counts with ScispaCy model without replacing entities:
Train accuracy: 0.976457399103139
Validation accuracy: 0.8505786249149081


Accuracy using TF-IDF with ScispaCy model without replacing entities:
Train accuracy: 0.9191423766816144
Validation accuracy: 0.8498978897208985


Accuracy using NMF with ScispaCy model without replacing entities:
Train accuracy: 0.703335201793722
Validation accuracy: 0.7066031313818925


Accuracy using LDA with ScispaCy model without replacing entities:
Train accuracy: 0.7550448430493274
Validation accuracy: 0.7620830496936691
```

Using Scispacy model replacing with entity type

```
Accuracy using Counts with ScispaCy model after replacing entities:
Train accuracy: 0.9677690582959642
Validation accuracy: 0.8434309053778081


Accuracy using TF-IDF with ScispaCy model after replacing entities:
Train accuracy: 0.9119955156950673
Validation accuracy: 0.8519400953029271


Accuracy using NMF with ScispaCy model after replacing entities:
Train accuracy: 0.703335201793722
Validation accuracy: 0.7066031313818925


Accuracy using LDA with ScispaCy model after replacing entities:
Train accuracy: 0.703335201793722
Validation accuracy: 0.7066031313818925
```

Model Results

```
Accuracy using Counts with Linear SVC model:
Train accuracy :              0.964
Validation accuracy :         0.845
Sensitivity :                 0.803
Specificity :                 0.863


Accuracy using TF-IDF with Linear SVC model:
Train accuracy :              0.904
Validation accuracy :         0.816
Sensitivity :                 0.778
Specificity :                 0.833


Accuracy using NMF with Linear SVC model:
Train accuracy :              0.633
Validation accuracy :         0.628
Sensitivity :                 0.596
Specificity :                 0.641


Accuracy using LDA with Linear SVC model:
Train accuracy :              0.496
Validation accuracy :         0.487
Sensitivity :                 0.773
Specificity :                 0.363


Accuracy using GloVe with Linear SVC model:
Train accuracy :              0.704
Validation accuracy :         0.702
Sensitivity :                 0.025
Specificity :                 0.996


Accuracy using DistlBERT with Linear SVC model:
Train accuracy :              0.854
Validation accuracy :         0.829
Sensitivity :                 0.825
Specificity :                 0.831


Accuracy using BioBERT with Linear SVC model:
Train accuracy :              0.839
Validation accuracy :         0.807
Sensitivity :                 0.798
Specificity :                 0.811
```

Counts and TF-IDF have performed well with Counts having a validation accuracy of 86% but there seems to be quite a bit of over fitting, which we could expect because of the high dimensions.

Both NMF and LDA perform poorly with LDA having a validation accuracy of 48%!

GloVe seems to have a decent accuracy, but that's because it just labels everything as positive. Clearly GloVe does not seem to be able to distinguish between positive and negative reviews.

DistilBERT and BioBERT seem to have comparable accuracies, with DistilBERT's being a little higher. We expected it to be the other way around but this could be becaus of the higher truncation we had to do for BioBERT. If they were trained on the same configurations, BioBERT could have actually performed better!

Also, BERT and BioBERT seem to have overfit very minimally!!

RBF SVC

Model Results

```
Train accuracy :              0.934
Validation accuracy :         0.852
Sensitivity :                 0.82
Specificity :                 0.866


Accuracy using TF-IDF with RBF SVC model:
Train accuracy :              0.986
Validation accuracy :         0.88
Sensitivity :                 0.801
Specificity :                 0.915


Accuracy using NMF with RBF SVC model:
Train accuracy :              0.646
Validation accuracy :         0.637
Sensitivity :                 0.594
Specificity :                 0.656


Accuracy using LDA with RBF SVC model:
Train accuracy :              0.618
Validation accuracy :         0.609
Sensitivity :                 0.592
Specificity :                 0.616


Accuracy using GloVe with RBF SVC model:
Train accuracy :              0.733
Validation accuracy :         0.692
Sensitivity :                 0.439
Specificity :                 0.802


Accuracy using DistlBERT with RBF SVC model:
Train accuracy :              0.81
Validation accuracy :         0.806
Sensitivity :                 0.78
Specificity :                 0.817


Accuracy using BioBERT with RBF SVC model:
Train accuracy :              0.798
Validation accuracy :         0.788
Sensitivity :                 0.773
Specificity :                 0.794
```

The results are almost similar except that TF-IDF and GloVe seem to have done a better job than last time!

But TF-IDF is still overfitting and GloVe still has a low sensitivity.

The accuracy of DistilBERT and BioBERT seem to have decreased a bit, suggesting that the relationship between the independent and dependent variables is more linear. Also, there is literally no overfitting!

Naïve Bayes

Model Results

The results are far worse than the above two models. This could be attributed to the imbalance in classes and Naive Bayes is not the best suited when you have such issues.

One thing to note is that in this case, NMF, LDA and GLoVe have performed almost the same with a very low sensitivity.

```
Accuracy using Counts with Gaussian NB model:
Train accuracy :              0.699
Validation accuracy :         0.582
Sensitivity :                 0.848
Specificity :                 0.466


Accuracy using TF-IDF with Gaussian NB model:
Train accuracy :              0.699
Validation accuracy :         0.583
Sensitivity :                 0.843
Specificity :                 0.471


Accuracy using NMF with Gaussian NB model:
Train accuracy :              0.704
Validation accuracy :         0.69
Sensitivity :                 0.197
Specificity :                 0.903


Accuracy using LDA with Gaussian NB model:
Train accuracy :              0.668
Validation accuracy :         0.666
Sensitivity :                 0.188
Specificity :                 0.874


Accuracy using GloVe with Gaussian NB model:
Train accuracy :              0.683
Validation accuracy :         0.681
Sensitivity :                 0.156
Specificity :                 0.908


Accuracy using DistlBERT with Gaussian NB model:
Train accuracy :              0.749
Validation accuracy :         0.756
Sensitivity :                 0.684
Specificity :                 0.787


Accuracy using BioBERT with Gaussian NB model:
Train accuracy :              0.747
Validation accuracy :         0.741
Sensitivity :                 0.663
Specificity :                 0.775
```

Test accuracy of the best model

```
Accuracy using DistlBERT with Linear SVC model:
Train accuracy :              0.854
Validation accuracy :         0.829
Sensitivity :                 0.825
Specificity :                 0.831
```

Based on the above results, it mainly comes down to:

1. Counts with Linear SVC
2. TFIDF with RBF SVC
3. DistilBERT with Linear SVC
4. DistlBERT with RBF SVC model

The first two perform well on the train set but seem to overfit because of the lower validation accuracy.

DistilBERT with the RBF SVC model does not produce great results but does not overfit.

But, we decided to go with the DistilBERT model with Linear SVC because it gives pretty good results with very little overfitting.

Future steps given more time

1. Try using more advanced embedding techniques like RoBERTa or XLNet .
2. At first, we wanted to try the LSTM model for classification so we could give that a run.
3. Use different strategies for extracting important characteristics of each drug.