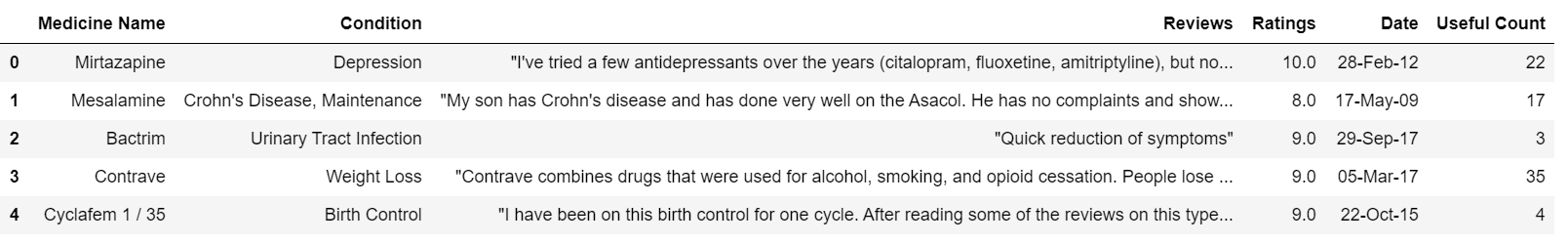
**Project Report**

Collecting and pre-processing the data:

**Scraping the reviews:**

For developing a decisive system, a subset of the extensive review data available on the Internet( Kaggle, Github, Drugs.com, WebMD) is taken into consideration. The reviews for neurological drugs for the treatment of epilepsy, seizures and bipolar disorders were scraped using scrapy, a Python library for developing custom web crawlers.

The final dataset consisted of an 214262 Rows and 6 columns .



**Cleaning the reviews:**

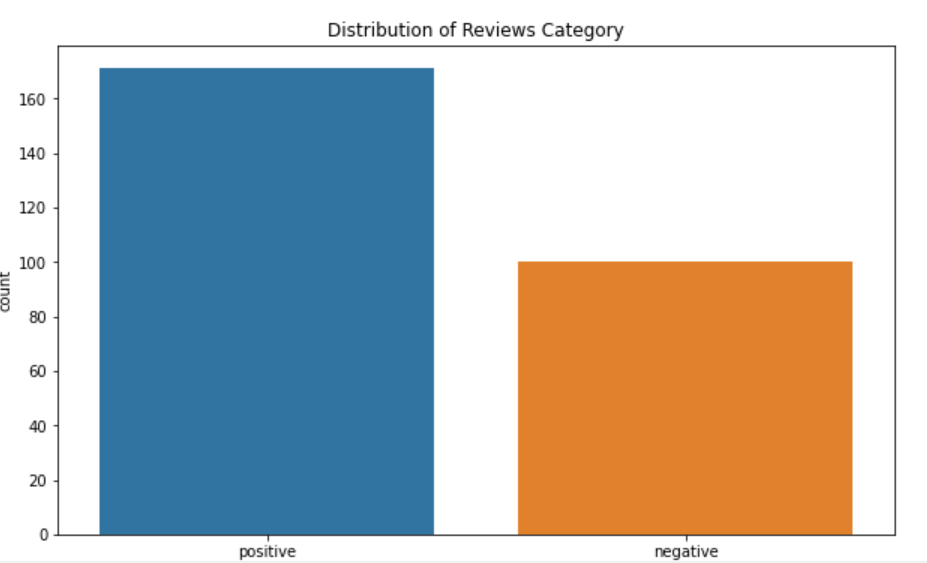
* Tokenizing the reviews into sentences  from Natural Language Toolkit(nltk).
* Standardizing of text which involved lowercase conversion, splitting of conjugate words, and correcting misspelled words, remove whitespaces, remove numbers.
* Lemmatization to get the root word form of the words using nltk.
* The stopwords, negation, and punctuation are retained in this step to preserve the information contained in the reviews as best as possible. At the end of this step, the cleaned sentences are ready to be labeled into appropriate categories.

**Distribution of reviews Category:**

The sentence can be classified into one of the Two categories:

positive: The reviews in which the improvement of patient’s health is implied after use of the drug.

Negative: The reviews which contain explicit mentions of adverse reactions to the patient after use of the drug.



The POS(parts-of-speech) tags of individual words of the sentence, generated using the nltk library.

**Choosing the right approach:**

Count Vectorizer approach: The CountVectorizer approach (creation of a sparse matrix of the size of words \* reviews) or the Term Frequency-Inverse Document Frequency(TF-IDF) approach (measures the frequency of a word along with the rareness of the word in the collection).

**Creation of bi-grams and tri-grams:**

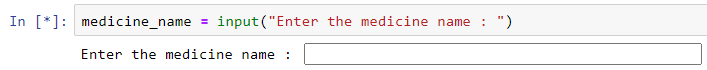
In NLP, each word in the text document is referred to as a ‘gram’. Thus, a combination of co-occurring words is known as an n-gram, where n is the length of the combination considered.

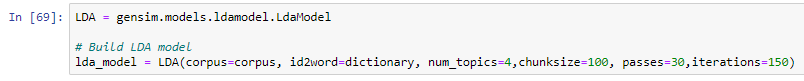
The bi-grams or tri-grams may be obtained as features independently using Gensim or by using scikit-learn’s feature extraction module to automatically generate them during vectorization.

**Building the Topic Model:**

LDA’s approach to topic modeling is it considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords, again, in a certain proportion.

The two main inputs to the LDA topic model are the dictionary(id2word) and the corpus. Gensim creates a unique id for each word in the document.





## **View the topics in LDA model:**

**Topic 0 is a represented as  \_'0.006\*" blood pressure" + 0.005\*"\'experienced depression" + 0.005\*" level headed"**

**It means the top 3 keywords that contribute to this topic are: " blood pressure" ,"experienced depression" ," level headed".. and so on and the weight of ‘blood pressure’ on topic 0 is 0.006.**

**Compute Model Perplexity and Coherence Score:**

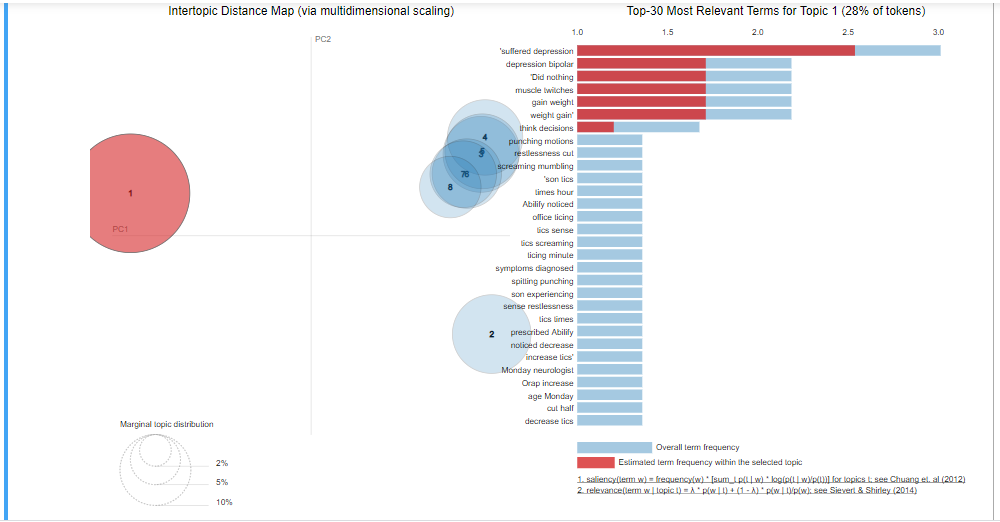
It provide a convenient measure to judge how good a given topic model is.

Perplexity: -6.562929451880235

Coherence Score: 0.8520706871504424

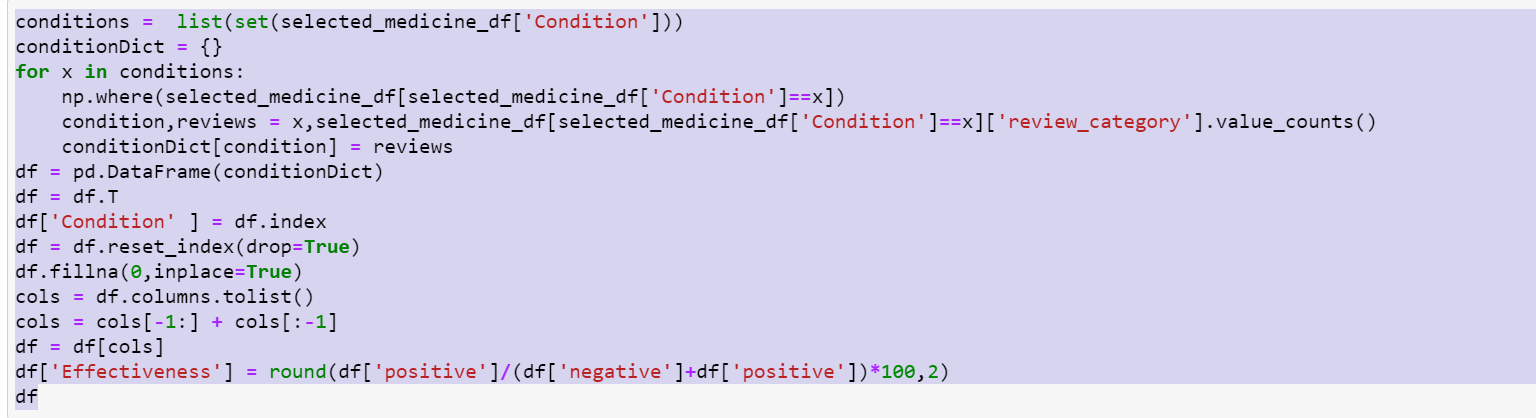
## **Visualize the topics-keywords:**

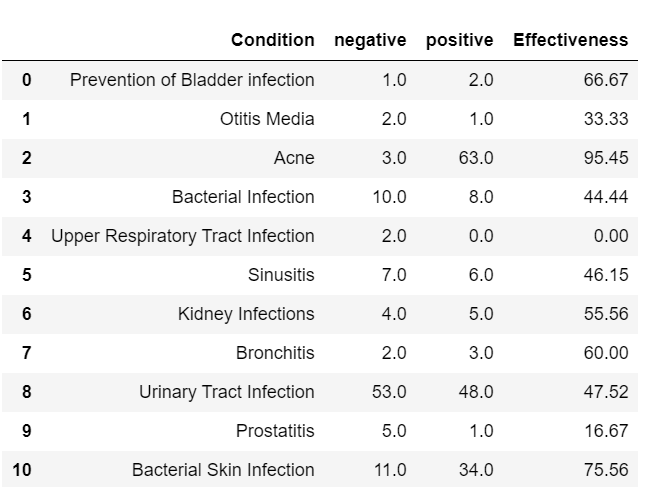
pyLDAvis package’s interactive chart and is designed to work well with jupyter notebooks. Each bubble on the left-hand side plot represents a topic. The larger the bubble, the more prevalent is that topic. A good topic model will have fairly big, non-overlapping bubbles scattered throughout the chart instead of being clustered in one quadrant. A model with too many topics, will typically have many overlaps, small sized bubbles clustered in one region of the chart.



## **Find the most effectiveness of the drug:**

Effectiveness is taken from Based on Rating and Number of Reviews ,condition wise.it is showing that each drug effectiveness percentage.





# Deployment: Deployment involves packaging up your web application and putting it in a production environment that can run the app. Here we are using streamlit for deployment

# 