Prescription of Medicine using Knowledge Graph and Sentiment Analysis

Project report submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of the degree

Bachelor of Technology
In
Computer Science and Engineering

by

A Akhil Naik (BT16CSE001) **A Hrisheek Kumar (BT16CSE006)**
B Bhuvaneswar (BT16CSE011) **K Vamsee Krishna (BT16CSE036)**

under the guidance of

Dr Syed Taqi Ali

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

2020

Prescription of Medicine using Knowledge Graph and Sentiment Analysis

Project report submitted to
Visvesvaraya National Institute of Technology, Nagpur
in partial fulfillment of the requirements for the award of the degree

Bachelor of Technology
In
Computer Science and Engineering

by

A Akhil Naik (BT16CSE001) **A Hrisheek Kumar (BT16CSE006)**
B Bhuvaneswar (BT16CSE011) **K Vamsee Krishna (BT16CSE036)**

under the guidance of

Dr Syed Taqi Ali

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology
Nagpur 440 010 (India)

2020

Department of Computer Science and Engineering
Visvesvaraya National Institute of Technology, Nagpur

Declaration

We, A Akhil Naik, A Hrisheek Kumar, B Bhuvaneswar, K Vamsee Krishna, hereby declare that this project work titled “Prescription of Medicine using Knowledge Graph and Sentiment Analysis” is carried out by us in the Department of Computer Science and Engineering of Visvesvaraya National Institute of Technology, Nagpur.

The work is original and has not been submitted earlier whole or in part for the award of any degree at this or any other Institution.

Date:

Sr. No	Enrollment No	Name	Signature
1	BT16CSE001	A. Akhil Naik	
2	BT16CSE006	A. Hrisheek Kumar	
3	BT16CSE011	B. Bhuvaneswar	
4	BT16CSE036	K. Vamsee Krishna	

Certificate

This is to certify that the project titled “Prescription of Medicine Using Knowledge Graph and Sentiment Analysis”, submitted by Ajmeera Akhil Naik, Asanabada Hrisheek Kumar, Baditi Bhuvaneswar, Kakumanu Vamsee Krishna in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering**, VNIT Nagpur. The work is comprehensive, complete and fit for final evaluation.

Dr. Syed Taqi Ali
Assistant Professor, Computer Science and Engineering, VNIT, Nagpur.

Dr. U. A. Deshpande
Head, Department of Computer Science and Engineering, VNIT, Nagpur
Date:

ACKNOWLEDGEMENT

We wish to express our sincere thanks to our project guide Dr. Syed Taqi Ali, Assistant Professor, Department of Computer Science and Engineering, VNIT Nagpur for their invaluable guidance and continuous support till the end of this project.

We wish to pay our sincere thanks to Dr. U. A. Deshpande, Head of Department, Department of Computer Science and Engineering, VNIT Nagpur, for providing the requisite facilities needed to complete the project.

We owe our gratitude to the Department of Computer Science and Engineering, VNIT Nagpur for granting us the unrestricted access to the lab. Also, we are sincerely thankful to the entire faculty of Department of Computer Science and Engineering, VNIT Nagpur.

ABSTRACT

Finding a correct medicine for a person suffering from a particular disease is still a challenging aspect. In this thesis, a software for Implementing Knowledge Graph and Sentiment Analysis to find out most suitable medicine for a certain disease to help medical personals in health domain; this system deals with a query statement that will be given as a input and find relation between entities like person and disease by knowledge graph, and medicine found through sentiment analysis of review statements given by consumers in the dataset.

MOTIVATION

In present days many medical practitioners are relying on computer-based applications for medical treatment and nowadays people often order medicines online or refer to various medicines online. This software helps many ordinary people and people related to medical industry to predict most suitable medicine for a disease through the application of Knowledge Graph and Sentiment Analysis. Thus, making people to choose the right medicine for the disease they are suffering from. The Dataset we will be working consists diseases which are faced by present world. Thus this system helps to find solution for some real-world problems.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 IDEA BEHIND THE PROJECT

In recent years usage of computer science and its applications in medical field is increasing drastically. And people are using internet for finding solutions for their health problems and end up with multiple choices of medicines which leads to confusion, to help with this problem of finding a correct medicine which is deduced from the previous experiences of users of those particular medicines.

1.2 BASIC OVERVIEW OF THE SYSTEM

The system comprises of the following major parts

- Creation of knowledge graph
- Obtaining Entities (person and disease)
- Exploratory Data Analysis
- Pre-processing of review statements
- Model Building of Deep Learning model, Light GBM classifier and Dictionary Sentiment Analysis
- Sentiment Analysis of review given by patient

This system takes the input query like 'somebody is suffering from some disease'. System finds out the person's name and disease he is suffering from that statement and medicine for that disease found Sentiment analysis of review statements given by patient who has used them earlier and is stored in a dataset, we analyse the dataset and find out its properties and then we do the pre-processing of the review statements like removing rows with null values, cleaning the required review statements like removing unwanted symbols etc., and converting it into vectorized form. Building the models for deep learning, to improve performance we build machine learning model using light gbm classifiers and for further improvement we perform dictionary sentiment analysis and take the added normalized scores of three models to detect the best medicine available. Hence the correct medicine can be founded as per the limitations of the dataset.

CHAPTER 2
LITERATURE SURVEY

2.1 NATURAL LANGUAGE PROCESSING:

Natural Language Processing is an area in computer science [1] that consists techniques for analysing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language text processing for [2] application purpose.

2.1.1 MORPHOLOGY

Morphology is the study of word formation – like how words are built up from smaller pieces [3]. It is about the identification, analysis and description of the structure of a given language's Morphemes and other linguistic units, such as root words, affixes, parts of speech, intonations and stresses or implied context [4]
Example: Washing = wash + ing.

2.1.2 POS TAGGING AND TECHNIQUES

Part-of-speech tagging (POS tagging or POST) is the process of tagging up a word in a text as corresponding to a particular part of speech, which is performed based on its definition and context i.e., its relationship with the adjacent and related words in a sentence [5].
Different types of POS tagging are:

1. **LEXICAL BASED METHODS:** This method assigns the tag based on the most frequently occurring one with a word in the training corpus.
2. **RULE-BASED METHODS:** This method assigns POS tag, which is based on the set of rules for a statement. For example, we can have a rule that says, words ending with "ed" should be assigned to a verb.
3. **PROBABILISTIC METHODS:** Probabilistic method assigns the POS tags based on the probability of a particular tag sequence occurring.

2.2 KNOWLEDGE GRAPH

The Knowledge Graph which is introduced by Google is a "knowledge base" used by Google and its services to enhance its search engine's results with information gathered from [6] various resources. The information obtained from knowledge graph is presented to users in an information-box to the right side of search results [7] of Google Search.

A knowledge base is a kind of technology which is used to store complex structured and unstructured information used by a computer system [8].
Knowledge Graphs capture facts related to people, events, processes, applications, data and various things, and the relationships among them.
Knowledge Graphs provide more relevant results to searches using semantic-search techniques. Knowledge Graph of Google uses the relationships between words and concepts to understand the context of a query, to assign specific meaning to user intents [9] and show similar results accordingly.

2.3 SENTIMENT ANALYSIS

Sentiment analysis is the classification and interpretation of opinion and emotions (positive, negative and neutral) within text data using text-analysis techniques. Sentiment analysis allows concerned persons to identify a person's sentiment(opinion) toward products, brands or services in feedback.
Sentiment analysis models focuses on polarity of opinion (positive, negative, neutral) but also on feelings and emotions (angry, happy, sad, etc), and even on intentions (e.g. interested v. not interested) [10].
Different types of sentiment analysis are:

2.3.1 Fine-grained Sentiment Analysis

Polarity precision is important, one might consider expanding their polarity categories to include: Very positive, Positive, Neutral, Negative, Very negative [10].
The above is usually referred to as fine-grained sentiment analysis, and can be like 5-star ratings in a review, for example:
Too Positive = 5 stars, Too Negative = 1 star.

2.3.2 Emotion detection

This kind of sentiment analysis is used for detecting emotions, like happiness, sadness, anger, frustration and so on. Emotion detection sentiment analysis use lexicons (i.e. lists of words and the emotions they convey) or complex machine learning algorithms.
One of the downsides of using lexicons is that people can express emotions in different ways [10] with different kinds of words (like which can mean in the other way).

2.3.3 Aspect-based Sentiment Analysis

Aspect-based sentiment analysis helps us to detect whether the review is positive or negative. Usually, when analysing sentiments of texts, say medicine reviews, one wants to know which particular aspects or features people are mentioning in a positive, neutral, or negative way [10]. For example, in this text: "There were rashes after I used this ointment", an aspect-based classifier will show the result that the given sentence expresses a negative opinion about the ointment.

CHAPTER 3
SOFTWARE AND TOOLS

Note: No need of installation of the below mentioned tools because we are using Google Colab, If you are not using Google Colab then install the below mentioned tools as mentioned in their respective segments.

3.1 Python

Python is a programming language that has more efficient high-level data-structures and a simple but efficient approach to object-oriented programming. Python's syntax and dynamic typing and its interpreted nature, makes it an ideal language for scripting and rapid application development. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms. Python also has a standard-library-modules that provide access to system functionality. Some of these modules are designed to enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs [11].

Installation:

To download python, go to the official python download page
http://www.python.org/download/.
It is our choice to which version we want to download depending on our requirement. Once
python is downloaded installation can be done as normal executable package.
If the python 3.4.1 version is used then we do not need to follow this process
as the new update integrates this process in the installation phase and so we
no longer need to manually add the System path variable.
1. Open the start menu and type in "environment" and select option called "Edit the system environment variables".
2. When the "System Properties" window appears, click on "Environment variables"
3. Open the "Environment Variables" window opens. We can notice that it controls all the "System Variables" rather than just this associated with our user. Now click on "New" to create new variable for Python.
4. Now simply enter a name for our path and press "OK".

3.2 Anaconda

Anaconda is a distribution of the Python and R data science package. Formerly known as Continuum Analytics. More than 100 new packages are there in Anaconda. This Anaconda work environment is used for scientific computing, machine learning and data science [12].

Follow the below mentioned steps to install the Anaconda distribution of Python on Windows.

Installation Steps:
1. Visit the Anaconda downloads page
Visit the following link: Anaconda.com/downloads
The download page of Anaconda distribution of Python

2. Select Windows
Choose Windows in list of the three operating systems.

3. Download
Download the most recent python 3 release. At the time of writing, the most recent release was the python 3.6 version. For problem solvers, select the python 3.6 version. Select 64-bit or 32-bit according to your system configuration of windows. Select Python 3.6 version or higher. Python 2 is legacy Python and start downloading the exe.

4. Open and run the installer
After the download completes, open and run the .exe installer
Click Next to confirm the installation.
Click next to proceed with the installation.
Then agree to the license. Anaconda End User License Agreement.
Then Click I Agree to proceed
In Anaconda installation options. When installing Anaconda on Windows, do not check
Add Anaconda to my PATH environment variable

5. Opening the Anaconda Prompt from the Windows start menu
After the installation of Anaconda is complete, go to the Windows start menu and select the Anaconda Prompt [13].

3.3 Anaconda Navigator

Anaconda Navigator is a graphical user interface software (GUI) included in Anaconda distribution that allows us to launch applications and manage conda packages, environments, and channels without the usage of command-line commands.
The command-line program conda is a package manager and also an environment manager.
Navigator is a point-and-click way to work with packages and environments without the need to type conda commands in a terminal window. You can use it to install packages in an environment, run the packages, and update them – all inside Navigator [14].

Some applications available by default in Navigator:
Jupyter Lab

Jupyter Notebook
Spyder
RStudio
Anaconda Prompt (Windows only)
Anaconda PowerShell (Windows only)

3.4 Jupyter Notebook

The Jupyter Notebook is an open-source web application that helps us to create and share documents that consists of live code, visualizations and narrative text [15]. Can be used for data cleaning and transformation, statistical modelling, numerical simulation, data visualization and much more. Jupyter Notebooks combine your code, descriptive text, output, images, and interactive interfaces into a single notebook file(.ipynb) that is edited, viewed, and is used in a web browser[14].

3.5 Google Colab

Colab notebooks runs code on Google's cloud servers, that means you can leverage the power of Google hardware including GPUs and TPUs, regardless of the system RAM and GPU configuration.

The advantages of Google Colab over Jupyter notebook
Google Colab helps us to use Tesla K80 GPU it also gives us a total of 12 GB of RAM, and can be used up to 12 hours in row.
TPUs (Tensor Processing Units) developed by Google to accelerate operations.
Google Collaboratory helps us to import libraries without installing them, Saves to Google Drive which helps us to share and have multiple people at multiple places work on the same document at once, Collapsible section, Interactive widgets like sliders.
Scratch cell, which is a cell to run test code but which isn't saved in your notebook, Code snippets, etc.,
We will be working on Google Colab because our dataset which we will be working on in future is very huge like 160000 rows and it will take so much of time if we run it in individual PC, so we will run the code in Colab leveraging the GPU’s and TPU’s for better performance.

3.6 NumPy

NumPy: NumPy - Numerical Python is the fundamental package used for scientific computing with python. It is a highly specific library for performing numerical operations. It gives a

MATLAB-style syntax. Using this mathematical and logical operations on arrays can be performed [16].

Installation:
conda install -c anaconda numpy

Features:
NumPy offers comprehensive mathematical **functions**, random number generators, linear algebra routines, Fourier transforms etc [16].,

3.7 NLTK

The Natural Language Toolkit (**NLTK**) [17] is a library that helps us to build **Python** programs that work with human language data for the application of natural language processing (NLP). NLTK consists of text processing libraries for parsing, tokenization, stemming, tagging, classification and semantic reasoning.

Installing NLTK through Anaconda
Enter command conda install -c anaconda nltk.

Review the package upgrade, downgrade, install information and enter yes.

NLTK is downloaded and installed.

Features:

NLTK consists most of the algorithms such as tokenizing, part-of-speech tagging, stemming, sentiment analysis, topic segmentation, and named entity recognition.

NLTK helps the computer to analyse, pre-process, and understand the text in a file.

3.8 spaCy

spaCy is an open-source library for advanced Natural Language Processing (NLP) in Python. Plays a key role in the creation of Knowledge Graph. Neo4j, blazegraph and many other tools are available but spaCy is most well-known tool used around the globe. Of any NLP library released to date it provides the fast and accurate syntactic analysis. It also offers access to larger word vectors that are easier to customize.

Installation:
Install spacy.
Run the below command in Anaconda Prompt,

conda install -c conda-forge spacy.

Features:
Non-destructive tokenization
Named entity recognition
pretrained statistical models and word vectors
Part-of-speech tagging

3.9 Pandas

Pandas is one of the most widely used **python** libraries for importing and handling datasets of various formats. It provides high-performance, easy to use structures and a high-level data manipulation tool. The key data structure in Pandas is called the Data-Frame. Data-Frames helps us to store and manipulate tabular data in rows of observations and columns of variables.
Here Pandas is used for Pre-processing of the Dataset.

Installation:
Type pip install pandas in anaconda prompt.

Features:
Handling missing data.

Cleaning up data.

Input and output tools.

Multiple file formats supported.

Merging and joining of datasets.

Visualize.

Mask data.

Perform mathematical operations on the data.

3.10 Matplotlib

Matplotlib [18] is a library used for the plotting purpose in python and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI's like Jupyter Notebook, Google Colab, etc.

Installation:

For matplotlib: conda install -c conda-forge matplotlib.

Features:

Visualization of univariate and bivariate data
Fitting in and visualization linear regression models
Plotting of statistical time series data

3.11 KERAS - TENSORFLOW

The **Keras** is a powerful and easy-to-use free open source library **in Python**. It is one the most recommended library for developing and evaluating **deep learning** models [19]. Its minimalistic, modular approach makes it easy to get **deep neural networks** up and running. It wraps the efficient numerical computation libraries like Theano and TensorFlow and helps us to define and train neural network models in just a few lines of code. It is a wrapper library built on top of TensorFlow.

TensorFlow is a Python library which is used fast numerical computing. To create Deep Learning models this library is used as a foundation library. It is also an open source AI library, which uses data flow graphs to build models. It allows us to create large-scale neural networks with many layers. And it is a bit of time-taking procedure to code the neural network using TensorFlow. So, Keras is used for Deep Learning process in this project.

Installation:

KERAS

Step 1: Create a new conda environment where we will install our modules to build our models
Step 1.1: Open Anaconda Prompt (Run as Administrator)
conda create –name deeplearning
Step 1.2: Activate the conda environment that we just created use
activate deeplearning

Step 2: Install Keras
conda install -c anaconda keras

Important: Since this is a new environment you need to do few installations again, otherwise while importing keras it will give the following error:
ModuleNotFoundError: No module named 'keras'
Since we are working in Google Colab no need of writing these below commands
conda install jupyter #very important if you work on Jupyter Notebook
conda install spyder #very important if you work on spyder

Other Libraries if not present like
conda install matplotlib
conda install pandas

TensorFlow:
conda **install tensorflow**

Features:

It runs smoothly on both CPU and **GPU**.

Keras supports almost all kinds of models of a neural **network** – fully connected, recurrent, embedding, convolutional, pooling etc.

3.12 Light GBM

Light GBM [20] is a gradient boosting framework which uses a tree-based learning algorithm. Light GBM is given that name because of its high speed. It can handle large datasets. And it takes low memory to run. It is more famous because it is more accurate in results. It is widely used in data science application development because it supports GPU Learning. The tree grows leaf-wise in Light GBM whereas in other algorithms it grows level-wise. It will choose the max delta loss leaf to grow. while growing the same leaf, Leaf-wise algorithm reduces more loss compared to a level-wise algorithm.

LGBM on small datasets is not advisable. Light GBM was sensitive for overfitting and can overfit the small data. There is no threshold on the number of rows but our experience suggests me to use it only for data with 10,000+ rows.

Parameters involved in parameter tuning

IO parameters: categorical_feature, max_bin, save_binary, ignore_column.

Core parameters: multiclass, regression, binary, boosting, learning_rate, num_boost_round, device, num_leaves.

Metric parameter: Metric: mae, mse, binary_logloss, multi_logloss.

Control Parameters are

min_data_in_leaf, max_depth, feature_fraction, early_stopping_round, bagging_fraction, lambda, max_cat_group, min_gain_to_split.

Installation:
conda install -c conda-forge lightgbm

Features:
Labelled dependency parsing

Syntax-driven sentence segmentation
Built in visualizers for syntax and Name Entity Recognition

3.13 Sklearn

Scikit-learn (Sklearn) is an open-source machine learning library for **Python**. The algorithms which it provides are random forests, support vector machine and k-neighbours, and it also supports python scientific and numerical libraries like SciPy and NumPy. The library provides a lot of efficient tools for machine learning and statistical modelling including regression, classification, dimensionality reduction and clustering.

Installation:
conda install -c anaconda scikit-learn

3.14 TextBlob

TextBlob is a Python library that helps us for processing textual data. It helps the common natural language processing (NLP) by providing a simple API which performs the tasks like part-of-speech tagging, classification, sentiment analysis, translation, noun phrase extraction and more.

Installation:
conda install -c conda-forge textblob

3.15 TQDM

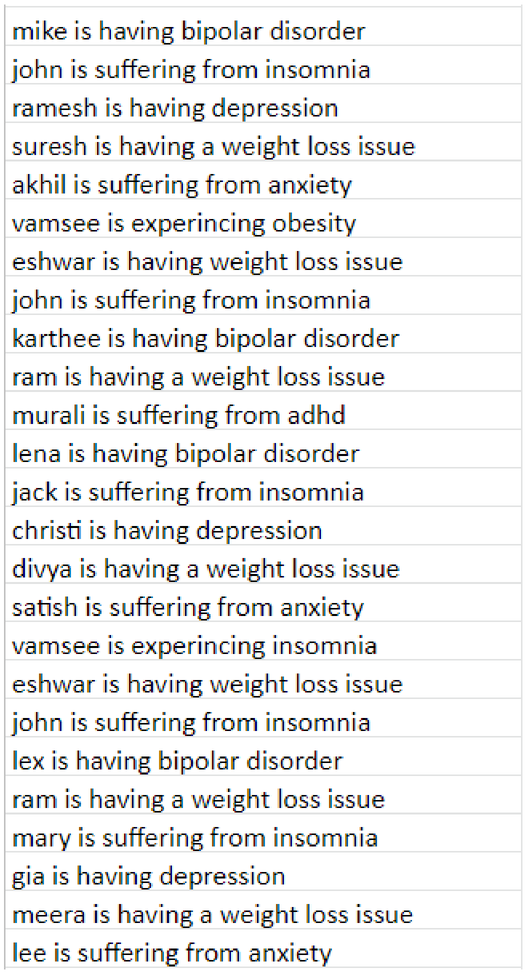
It is a package which is useful to show a smart progress meter when wrapped with any iterable like tqdm(iterable). sometimes it is executed as a module with pipes.

Installation:
conda install -c conda-forge tqdm

CHAPTER 4
METHODOLOGY

4.1 CREATION OF KNOWLEDGE GRAPH AND OBTAINING ENTITIES

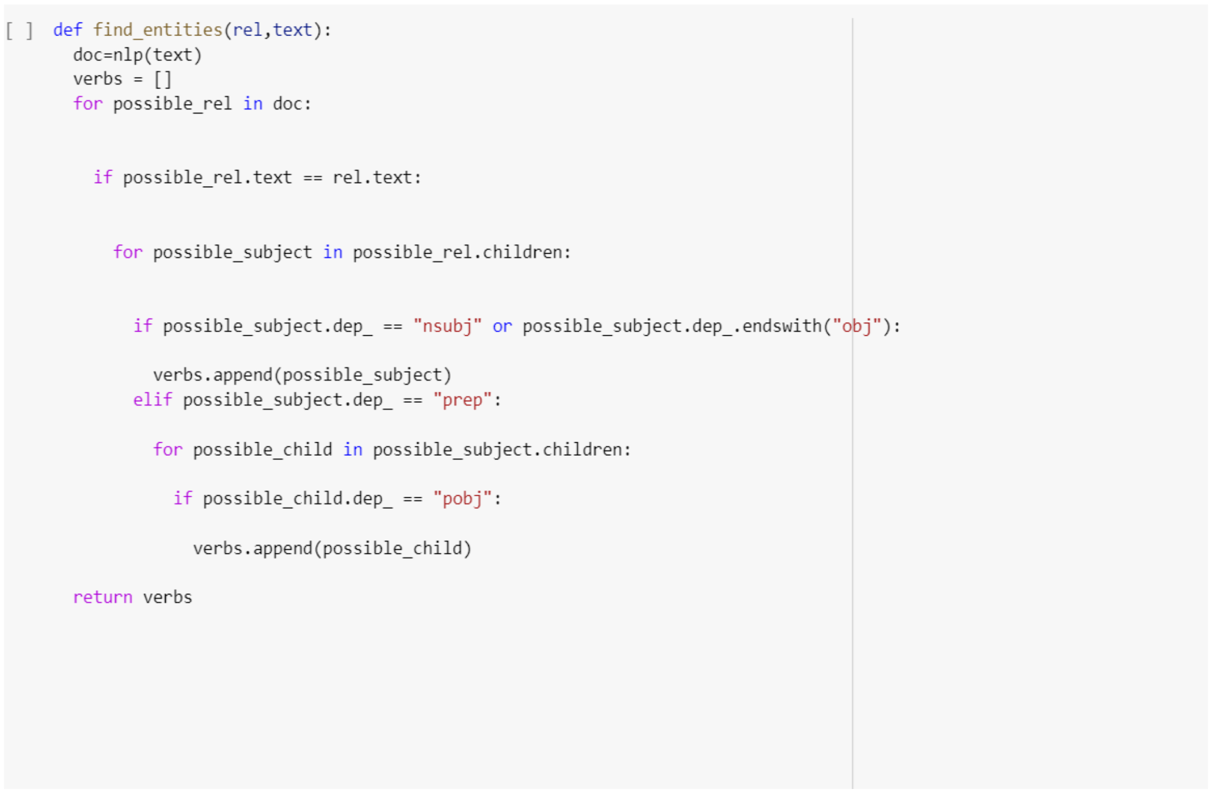
Our system takes the input statement as for example “Ram is suffering from fever”.
Like some in the image below



mike is having bipolar disorder
john is suffering from insomnia
ramesh is having depression
suresh is having a weight loss issue
akhil is suffering from anxiety
vamsee is experincing obesity
eshwar is having weight loss issue
john is suffering from insomnia
karthee is having bipolar disorder
ram is having a weight loss issue
murali is suffering from adhd
lena is having bipolar disorder
jack is suffering from insomnia
christi is having depression
divya is having a weight loss issue
satish is suffering from anxiety
vamsee is experincing insomnia
eshwar is having weight loss issue
john is suffering from insomnia
lex is having bipolar disorder
ram is having a weight loss issue
mary is suffering from insomnia
gia is having depression
meera is having a weight loss issue
lee is suffering from anxiety

Fig.1 Figure showing the sentences used for inputs.

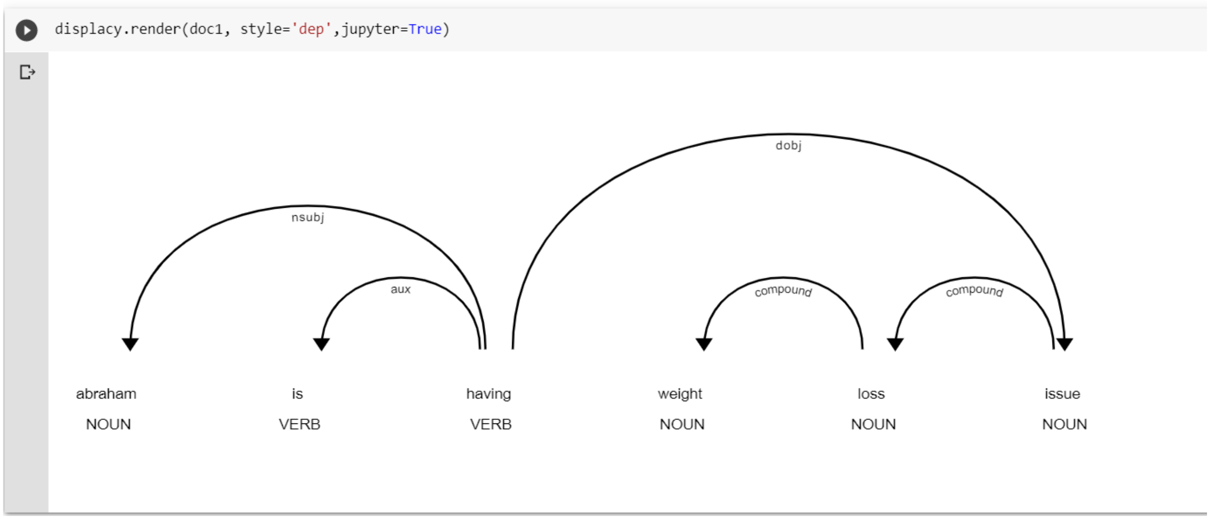
Then the system divides the input statement into nouns and verbs by using POS Tagging. Our system follows subject-verb-object structure first noun founded in sentence is subject and then after there will be verb and after that there will be another noun which is object, when the verb is found it is cross checked with the possible verbs which can mean suffering from a disease, it is like the essence of the sentence is decided on the verb like if the verb is suffering then he is suffering from some disease then the person and disease is found out or if the verb is like watching it doesn’t mean he is suffering from some kind of disease then the subject and object are not useful for us. Thus, we find the noun verb noun entity relation by using various methods in spaCy library taking some reference structures for a statement, which are pre-defined in the code, like “somebody is suffering from some disease” or “Somebody is having disease” etc.,



```
[ ]  def find_entities(rel,text):
       doc=nlp(text)
       verbs = []
       for possible_rel in doc:


         if possible_rel.text == rel.text:


           for possible_subject in possible_rel.children:


             if possible_subject.dep_ == "nsubj" or possible_subject.dep_.endswith("obj"):

               verbs.append(possible_subject)
             elif possible_subject.dep_ == "prep":

               for possible_child in possible_subject.children:

                 if possible_child.dep_ == "pobj":

                   verbs.append(possible_child)

       return verbs
```

Fig.2 Code for finding the entity pairs(i.e., person and disease) in the text.

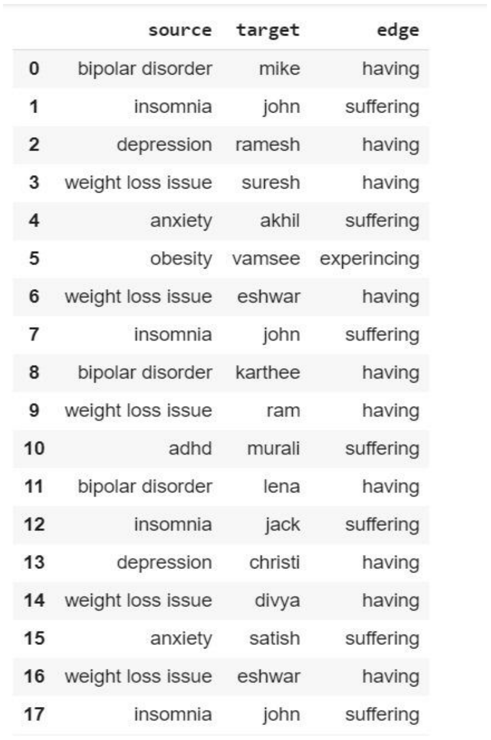
Creating a graph relation for some given statement.



```
displacy.render(doc1, style='dep',jupyter=True)
```

Fig.3 Dependency graph for a particular text input.

So here subject noun is ‘Abraham’, verb is ‘having’ and object noun is ‘weight-loss issue’, first we find the verb and then the next children nouns in the graph subject and object are found by traversing all the way from start to the end of the sentence. Thus we can find a relation between Abraham and weight-loss issue, Hence the two entities Abraham and weight-loss issue will be the outputs of this phase and the disease weight-loss issue is found and the disease is passed for Semantic Analysis of several review statements in the dataset for that specific disease.



	source	target	edge
0	bipolar disorder	mike	having
1	insomnia	john	suffering
2	depression	ramesh	having
3	weight loss issue	suresh	having
4	anxiety	akhil	suffering
5	obesity	vamsee	experincing
6	weight loss issue	eshwar	having
7	insomnia	john	suffering
8	bipolar disorder	karthee	having
9	weight loss issue	ram	having
10	adhd	murali	suffering
11	bipolar disorder	lena	having
12	insomnia	jack	suffering
13	depression	christi	having
14	weight loss issue	divya	having
15	anxiety	satish	suffering
16	weight loss issue	eshwar	having
17	insomnia	john	suffering

Fig.4 Entity pairs and their relation after applying knowledge graph.

4.2 SENTIMENTAL ANALYSIS OF REVIEW STATEMENTS

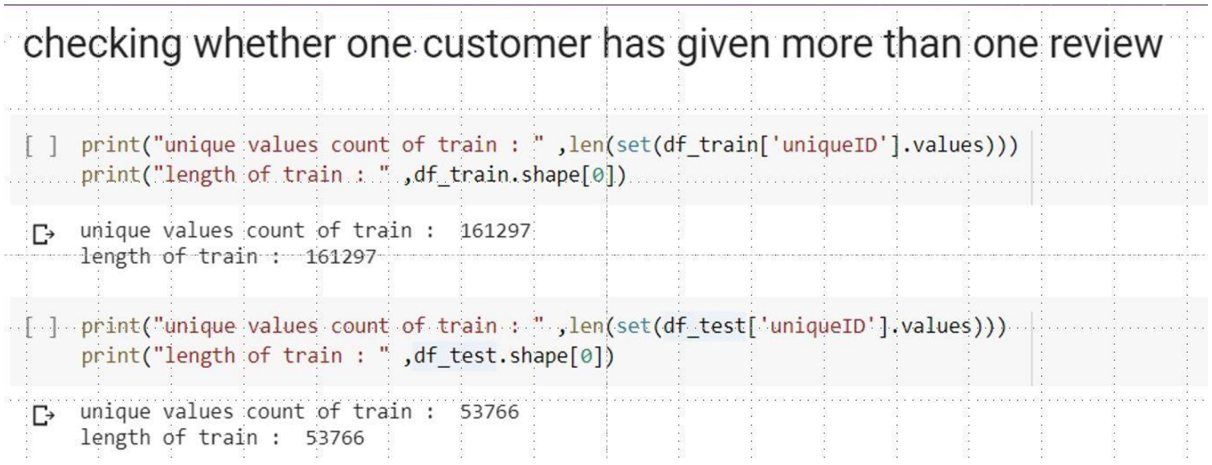
4.2.1 UNDERSTANDING THE DATASET

We have obtained the data set from the UCI Machine learning repository
It consists two datasets
1 Train data
2. Test data
Both these datasets consist of same column names
Following are column names
1 drugname: name of the drug. (i.e. medicine)
2 condition: name of the condition. (i.e. disease)
3 review: review from the patient.
4 rating: rating to that medicine given by patient. (max 10 star rating)
5 date: date on which the patient has given the review.
6 useful Count: no of users who found the review useful.

The flow of data set is that, when a patient purchases a drug for his particular condition (i.e. the disease he is suffering from) and writes a review and rating for that drug. The date of the review entered is also noted. If other users/patients see this as a useful review and when they hit the useful button the useful-count increases.

4.2.2 EXPLORATORY DATA ANALYSIS

In order to check the medicine to have better results doctors too test them on different persons. We prefer the data set which have reviews are from different persons.
We are checking if a person has given more than one review.


checking whether one customer has given more than one review

```
[ ]  print("unique values count of train : " ,len(set(df_train['uniqueID'].values)))
     print("length of train : " ,df_train.shape[0])

     unique values count of train :  161297
     length of train :  161297

[ ]  print("unique values count of train : " ,len(set(df_test['uniqueID'].values)))
     print("length of train : " ,df_test.shape[0])

     unique values count of train :  53766
     length of train :  53766
```

Fig.5 Checking whether one customer has given more than one review.

From the above fig we can say that the number of unique review and the length of the dataset (i.e total reviews) are same and hence all the reviews given are from different persons.

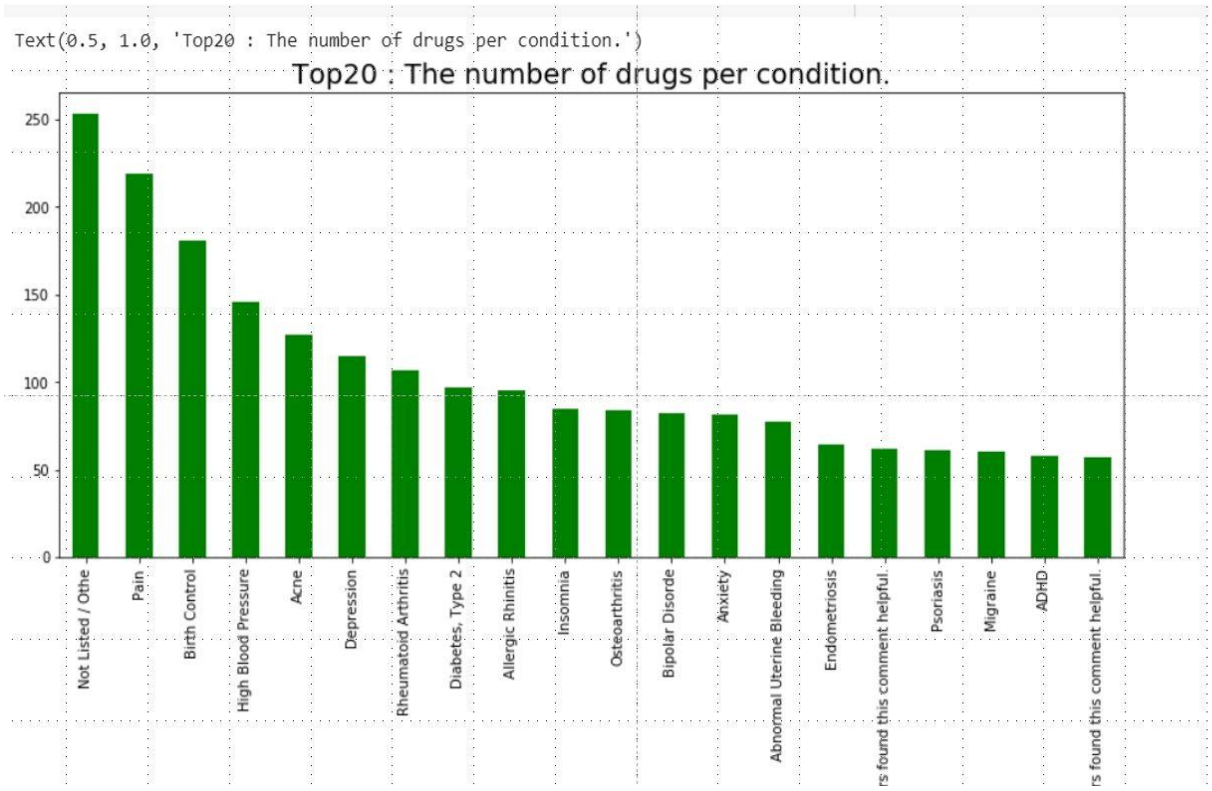

```
Text(0.5, 1.0, 'Top20 : The number of drugs per condition.')
```

Fig.6 Graph showing number of top 20 drugs per condition.

As we can see that top20 drugs per condition is greater than 100, whereas the bottom 20 drugs per condition is nearly 1 which is not recommendable to patient so we remove the diseases with 1 drug (i.e. medicine).

As we can see in above figure that the last column i.e. <\span> (users found this comment helpful) is not a condition and this might be an error while entering the review by the patients.
These types of errors will be cleaned up during data processing.

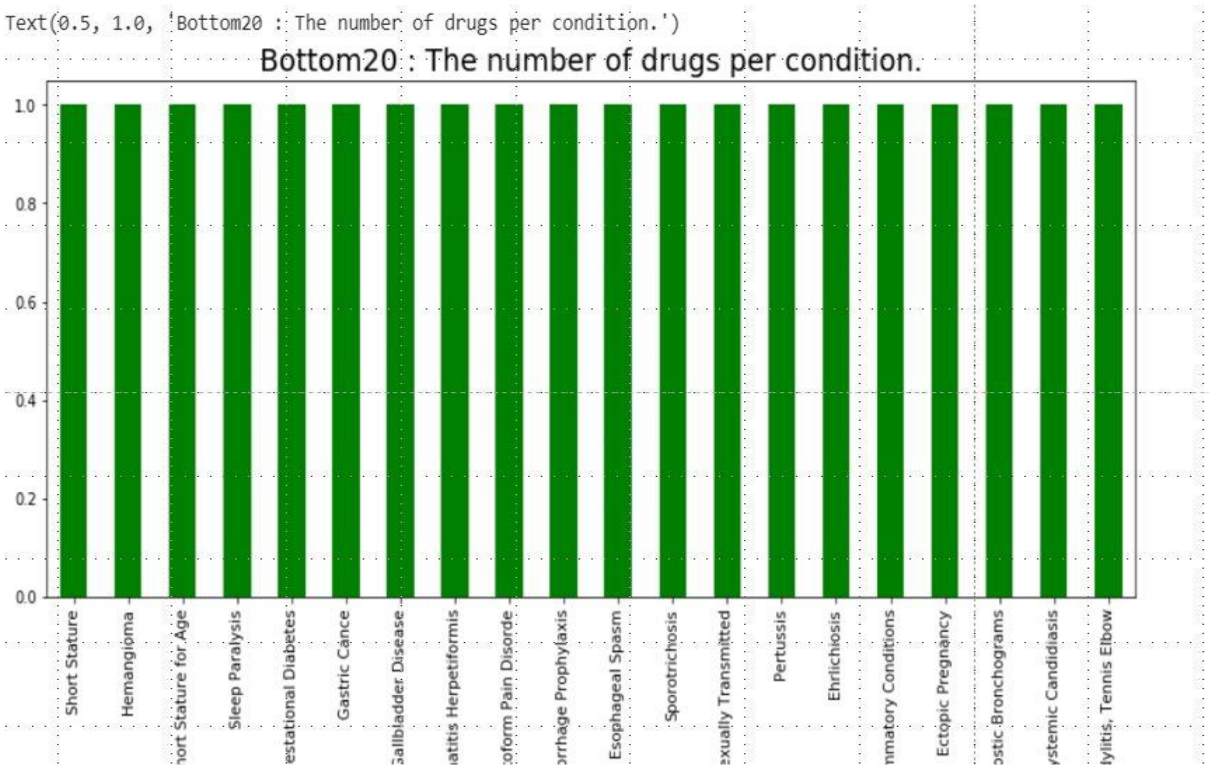
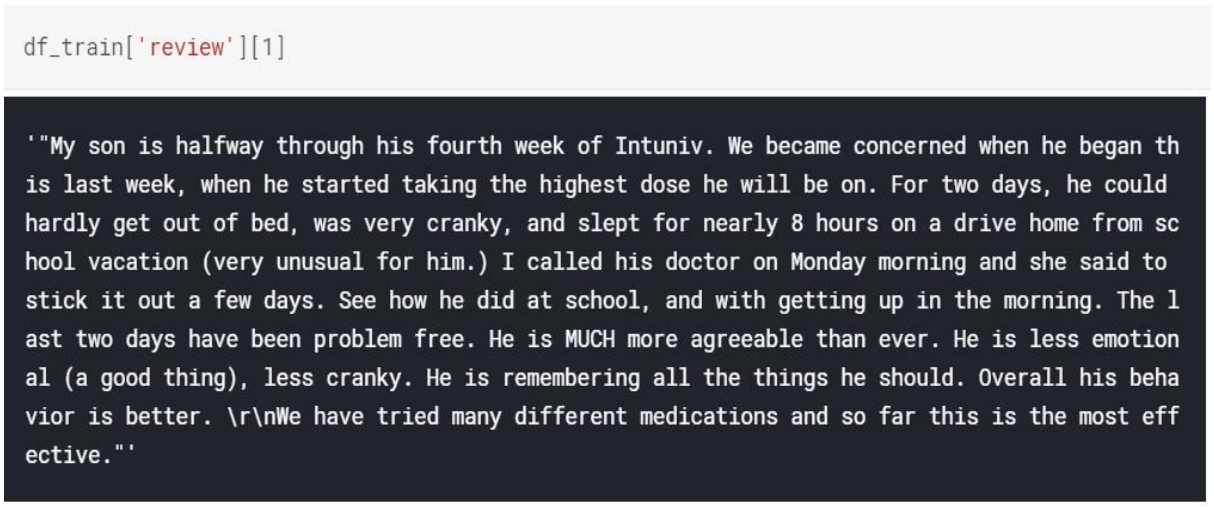


Fig.7 Graph showing number of bottom 20 drugs per condition.

Bottom 20 are those drugs with atmost 1 drug (medicine) for that particular condition (disease).
If there is only 1 medicine. It may or may not have a good effect on patient. It may or may not cure the patient. So, it is better to study the conditions with more than one drug.
Which will help the patient to choose good drug for the condition.



```
df_train['review'][1]
```

```
'"My son is halfway through his fourth week of Intuniv. We became concerned when he began th
is last week, when he started taking the highest dose he will be on. For two days, he could
hardly get out of bed, was very cranky, and slept for nearly 8 hours on a drive home from sc
hool vacation (very unusual for him.) I called his doctor on Monday morning and she said to
stick it out a few days. See how he did at school, and with getting up in the morning. The l
ast two days have been problem free. He is MUCH more agreeable than ever. He is less emotion
al (a good thing), less cranky. He is remembering all the things he should. Overall his beha
vior is better. \r\nWe have tried many different medications and so far this is the most eff
ective."'
```

Fig.8 Some random review statement given by a patient.

As we can that there are some reviews
1. With the html tags are present like \n or \r etc.

2. With the emotional words present within the parentheses.
3. With some words in uppercase letters.
We can remove these in data preprocessing for better results

Taking the ratings from 1-5 as negative and 6-10 as positive we will try to analyse the Text in the review column by considering the above classification.
We are using the n gram count plots to differentiate the reviews between 1-5 star-rating and 6-10 star-rating (i.e. we are counting the word sequences that occur in the reviews and then we are checking if we can differentiate between the negative and positive ratings).

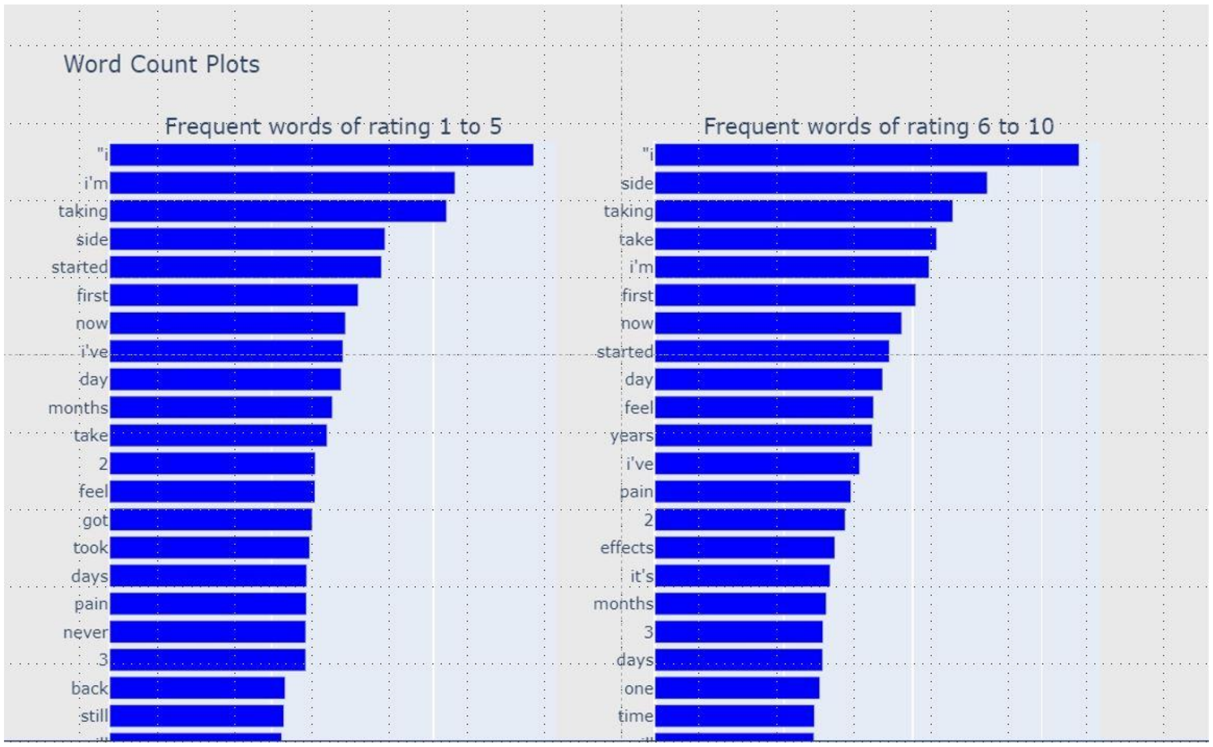


Fig.9 Comparison of rating using word count plots of Unigram.

As we can see that word count-plot we can't differentiate the reviews with single world
i.e. the probability of words in both the ratings are same in this word count plot.

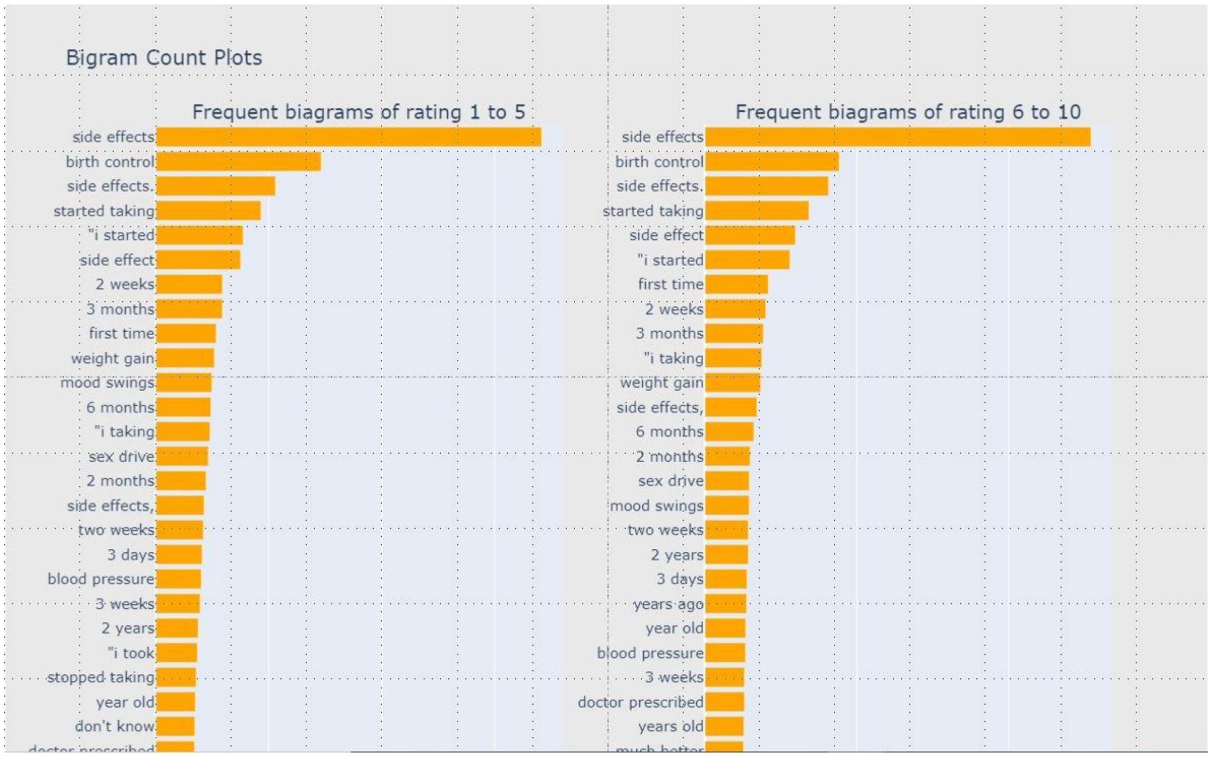


Fig.10 Comparison of rating using word count plots of Bigram.

And in this bigram count plot also the probability of the words is almost same in the both rating so we can't differentiate the reviews between positive and negative using bigram.

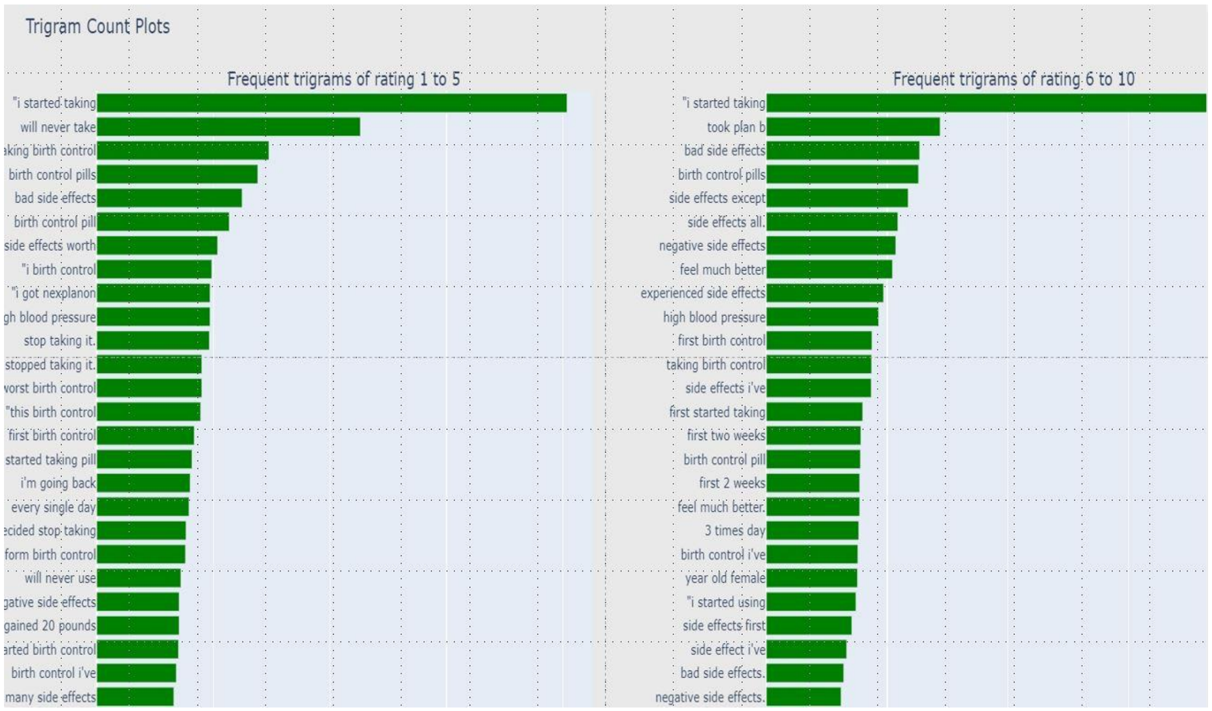


Fig.11 Comparison of rating using word count plots of Trigram.

From this Trigram count plot we can't differentiate between the positive review and negative review.

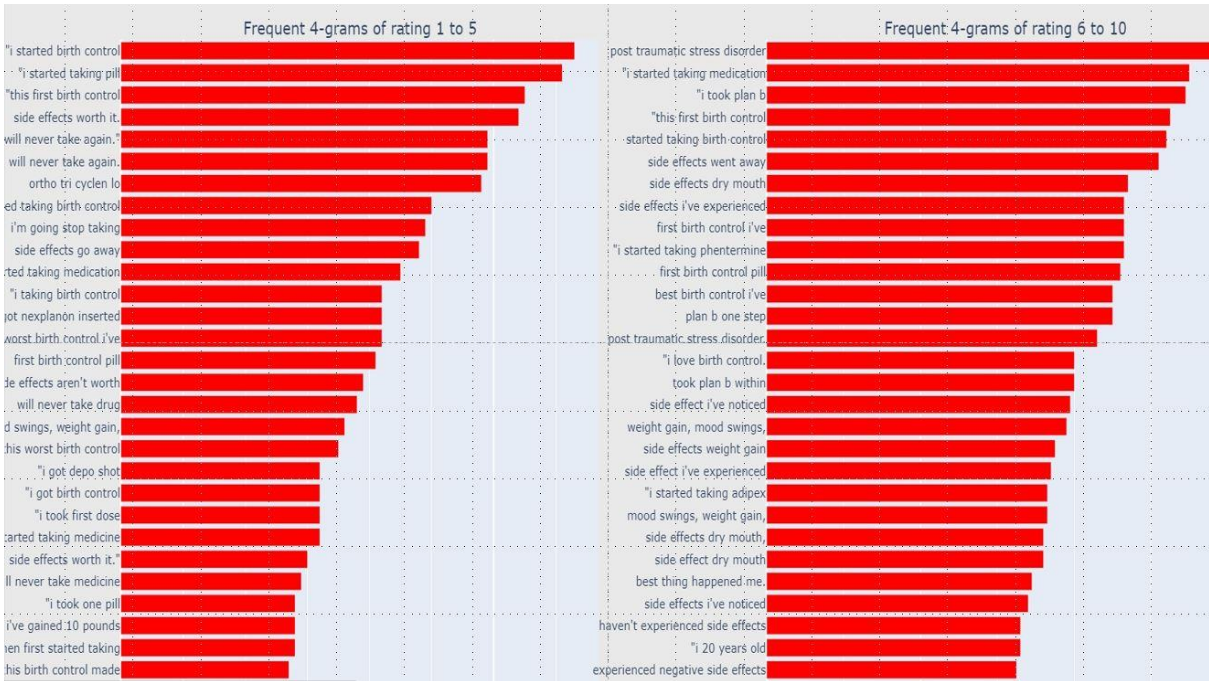


Fig.12 Comparison of rating using word count plots of 4-gram.

Using the 4-gram count plot we can see that the probability of sequence of words are not same in the both the rating ranges and we can clearly differentiate the reviews using this 4-gram count plot if it is positive review or negative review.

Count of Rating Values: We can see that most of the ratings for the drugs are very high or very low (i.e. 10,9,1) which shows that reviews given by the patients extreme.

```
rating = df_all['rating'].value_counts().sort_values(ascending=False)
rating.plot(kind="bar", figsize = (14,6), fontsize = 10,color="green")
plt.xlabel("", fontsize = 20)
plt.ylabel("", fontsize = 20)
plt.title("Count of rating values", fontsize = 20)
```

```
Text(0.5, 1.0, 'Count of rating values')
```

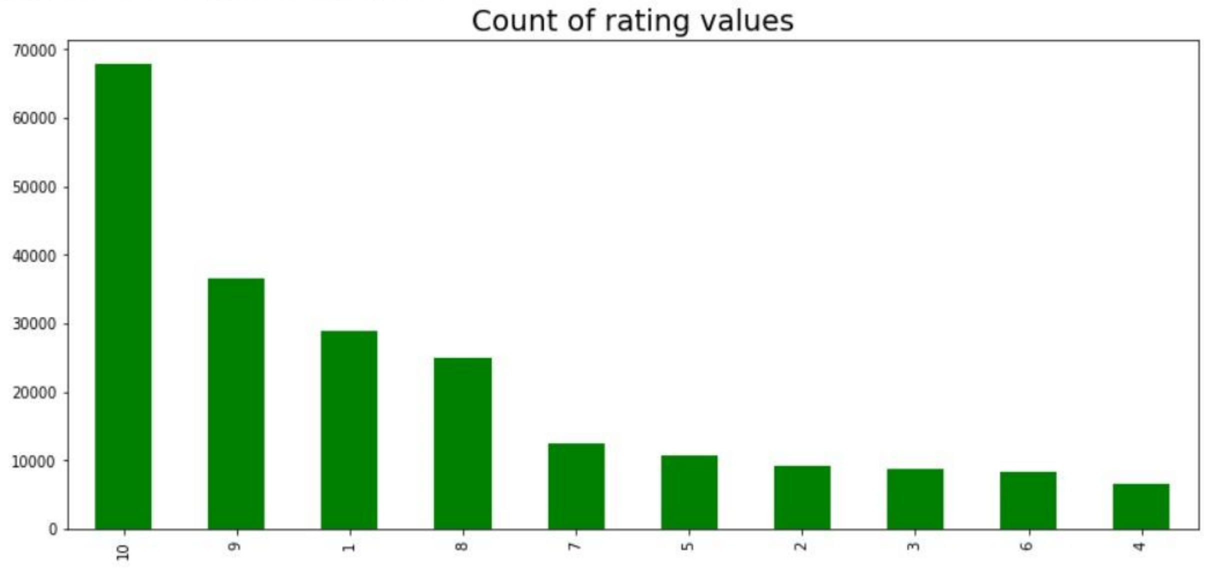
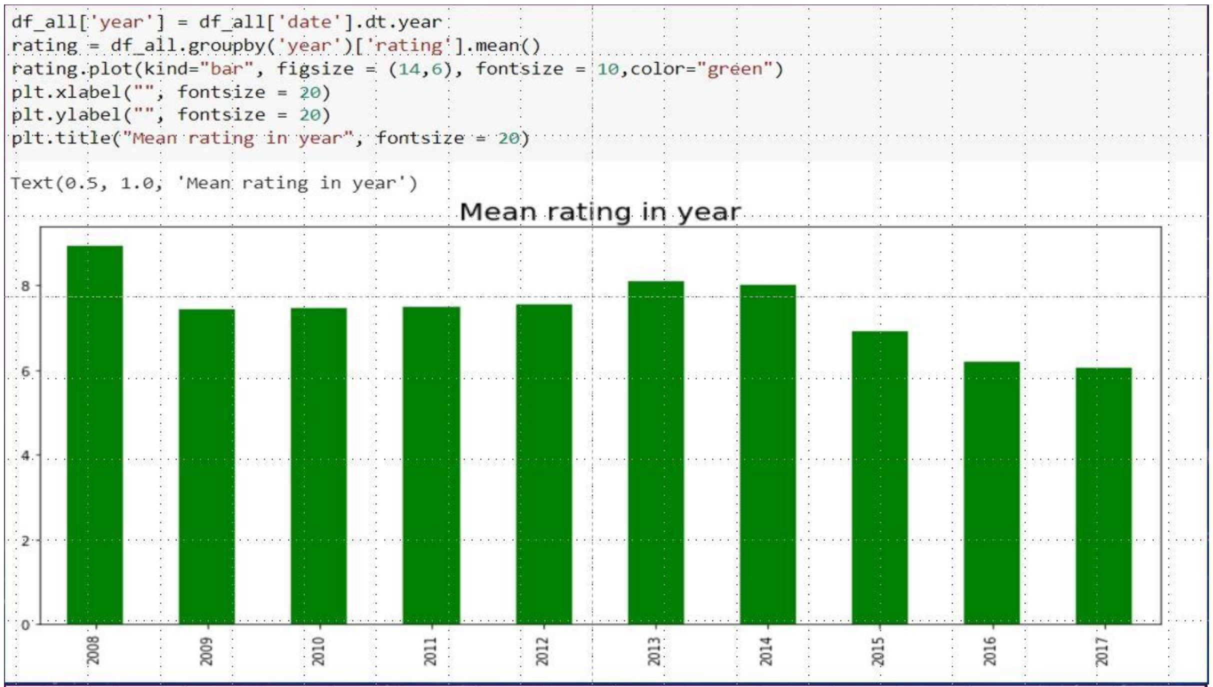


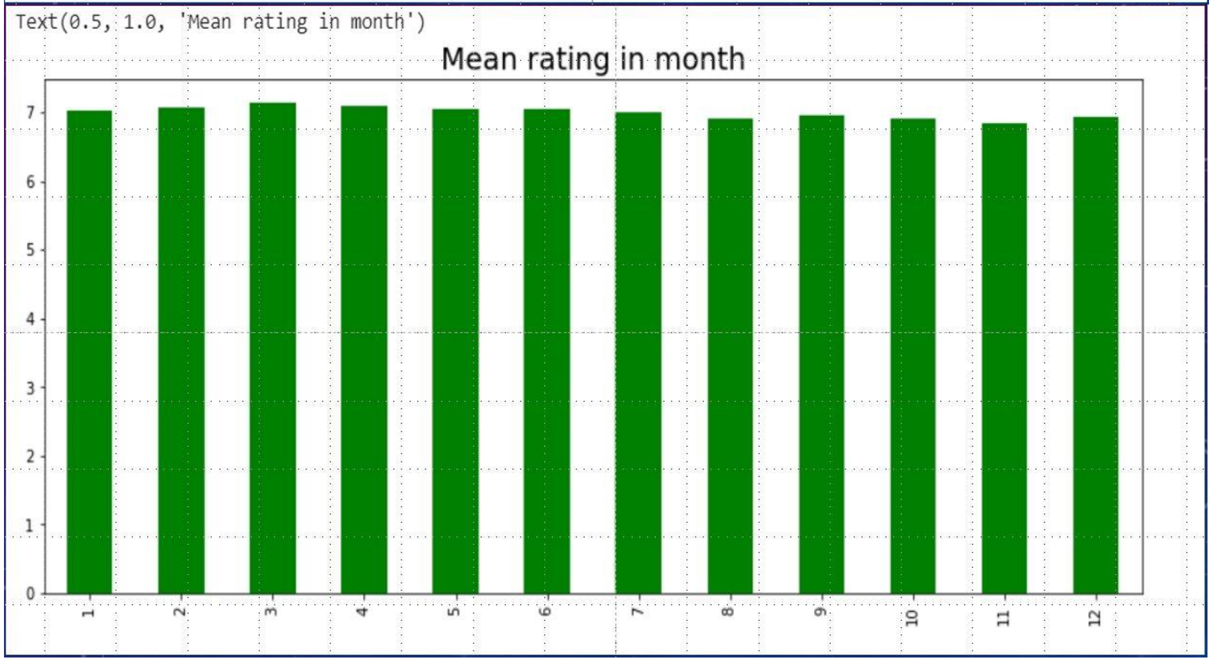
Fig.13 Graph of count of rating values in the dataset.

Impact of weather on rating: As the date on which the patient entered the review is noted. We are analyzing this date column and the plots are here:



```
df_all['year'] = df_all['date'].dt.year
rating = df_all.groupby('year')['rating'].mean()
rating.plot(kind="bar", figsize = (14,6), fontsize = 10,color="green")
plt.xlabel("", fontsize = 20)
plt.ylabel("", fontsize = 20)
plt.title("Mean rating in year", fontsize = 20)
```

```
Text(0.5, 1.0, 'Mean rating in year')
```

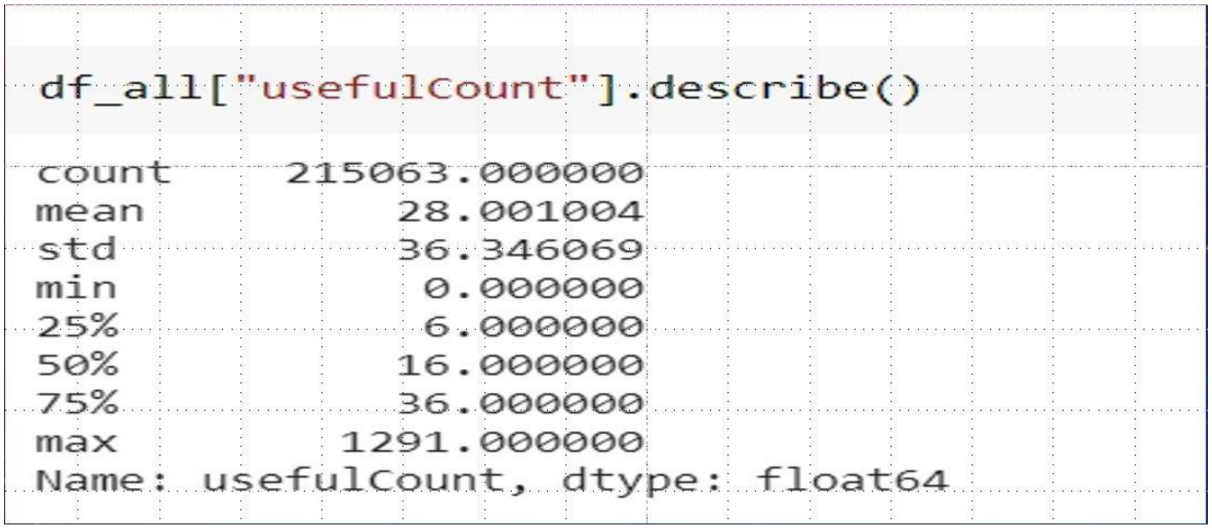


```
Text(0.5, 1.0, 'Mean rating in month')
```

Fig.14 Graphs showing the mean rating over a month and year

We can clearly see that the mean rating in month is almost same and whereas the number of reviews given in a year varies year to year. So, we can take the year inorder to analyze this date column but this may not be very useful because of the variation is small.

Study of useful count to analyse what most of the people prefer:



```
df_all["usefulCount"].describe()

count    215063.000000
mean         28.001004
std          36.346069
min           0.000000
25%           6.000000
50%          16.000000
75%          36.000000
max        1291.000000
Name: usefulCount, dtype: float64
```

Fig.15 Figure showing the Stats of the 'usefulCount'.

The difference between the max and min values is 1291 and is very high and the standard deviation 36 which is also very high.

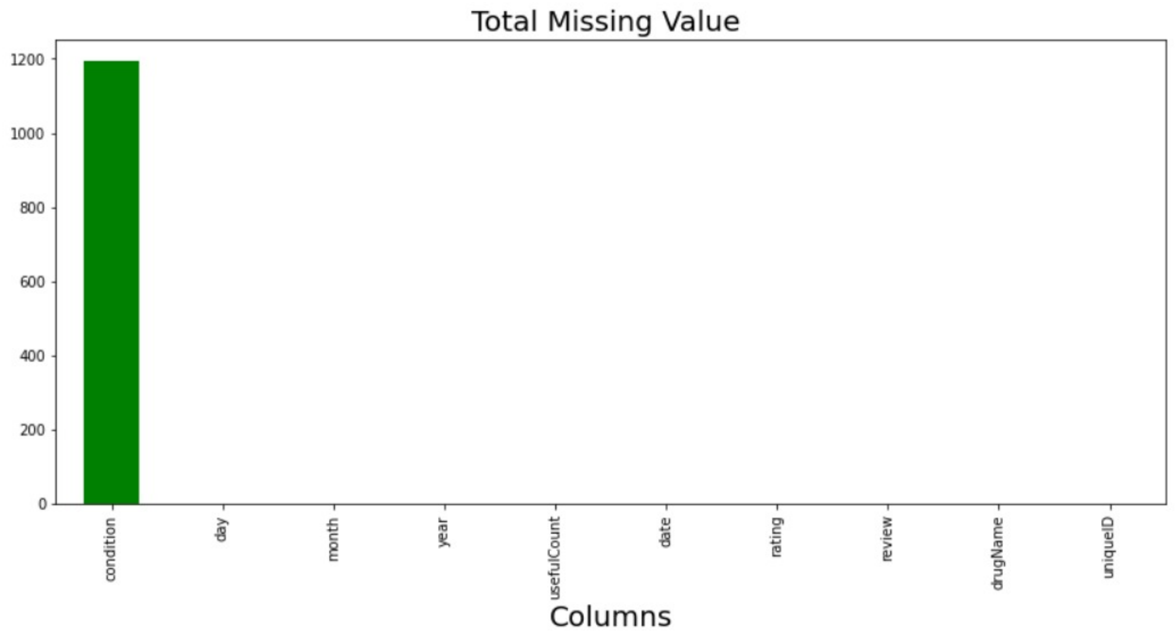
This is because the patients are viewing the conditions which they might not be suffering from and this type of cases increasing the useful count of the condition irrespective of the review is good or bad.

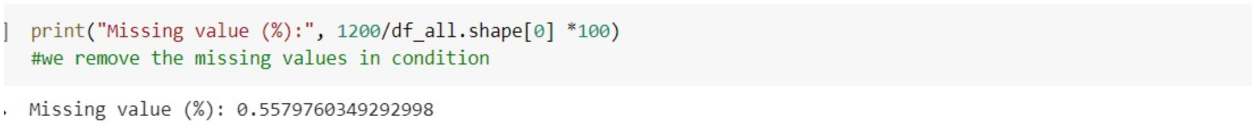
For example: when a person visits Amazon website for a mobile phone he would see many mobiles not only of the company which he prefers but also of different companies. Thus, it increases the useful count of the product.

So, we try to normalize these types of reviews also to better results.

Inorder to train the data using machine learning we may or may not need to use all the features available in the dataset and we should select the features which are best suitable for our output. (i.e. The features on which our results are dependent).

All the data features are analyzed during data analysis and then only certain features are selected which would give the accurate and efficient results which will increase the performance. Here we are trying to take the features corelated to 'rating' column.





```
] print("Missing value (%):", 1200/df_all.shape[0] *100)
  #we remove the missing values in condition
```

```
Missing value (%): 0.5579760349292998
```

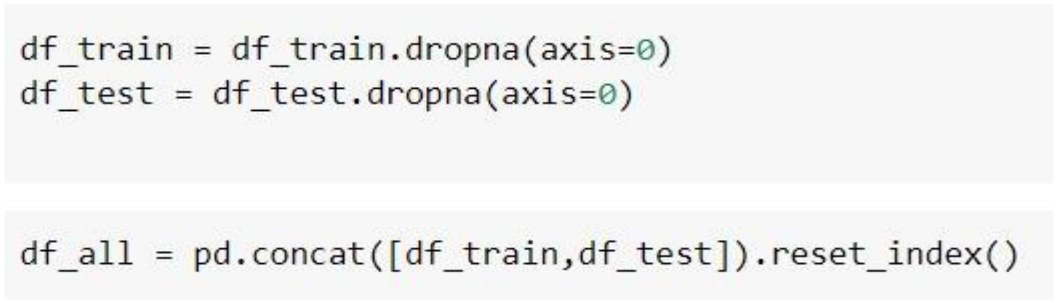
Fig.16 Count of number of Missing values and value of Missing value percentage in ‘condition’.

We are trying to find any missing values in the dataset because when we train for machine learning there should not be any missing values or null values in the input. If there are any missing values the training for machine learning will stop.

The missing values is less than 1% i.e. 1200 nearly. So, we will try to remove these missing values in the further pre-processing process.

4.2.3 PRE-PROCESSING

Missing values removal:



```
df_train = df_train.dropna(axis=0)
df_test = df_test.dropna(axis=0)
```

```
df_all = pd.concat([df_train,df_test]).reset_index()
```

Fig.17 Figure showing the process of removing null values in the datasets.

```
Text(0.5, 1.0, 'Total Missing Value ')
```

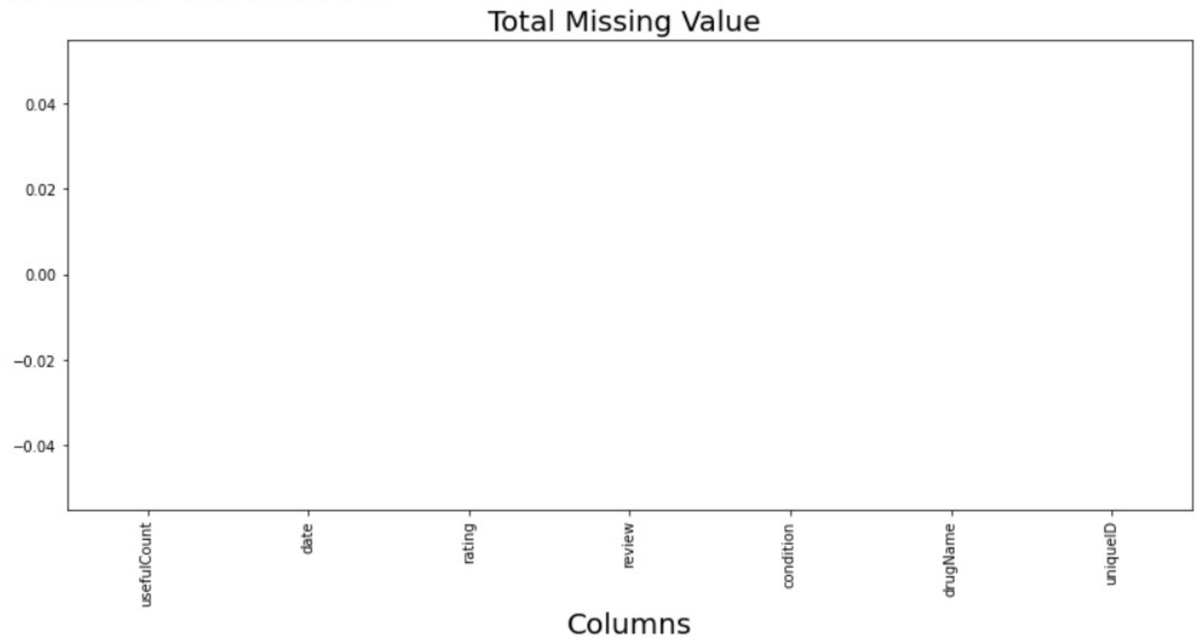


Fig.18 Graph suggesting the clean dataset without any null values.

By implementing the code lines mentioned in the above figure the missing values in the dataset are removed i.e. the row consisting of null values in the dataset is removed, which can be understood by seeing the above figure where there are no null values in any column and row.

Condition pre-processing:

Removal of rows consisting </span> kind of statements in the condition column. As we have observed in earlier data analysis part that in the condition column there were sentences </span> by mistake, instead of having a name of disease.

These kinds are being removed in this condition pre-processing, by using simple data-frame functions i.e. pandas data-frame functions.

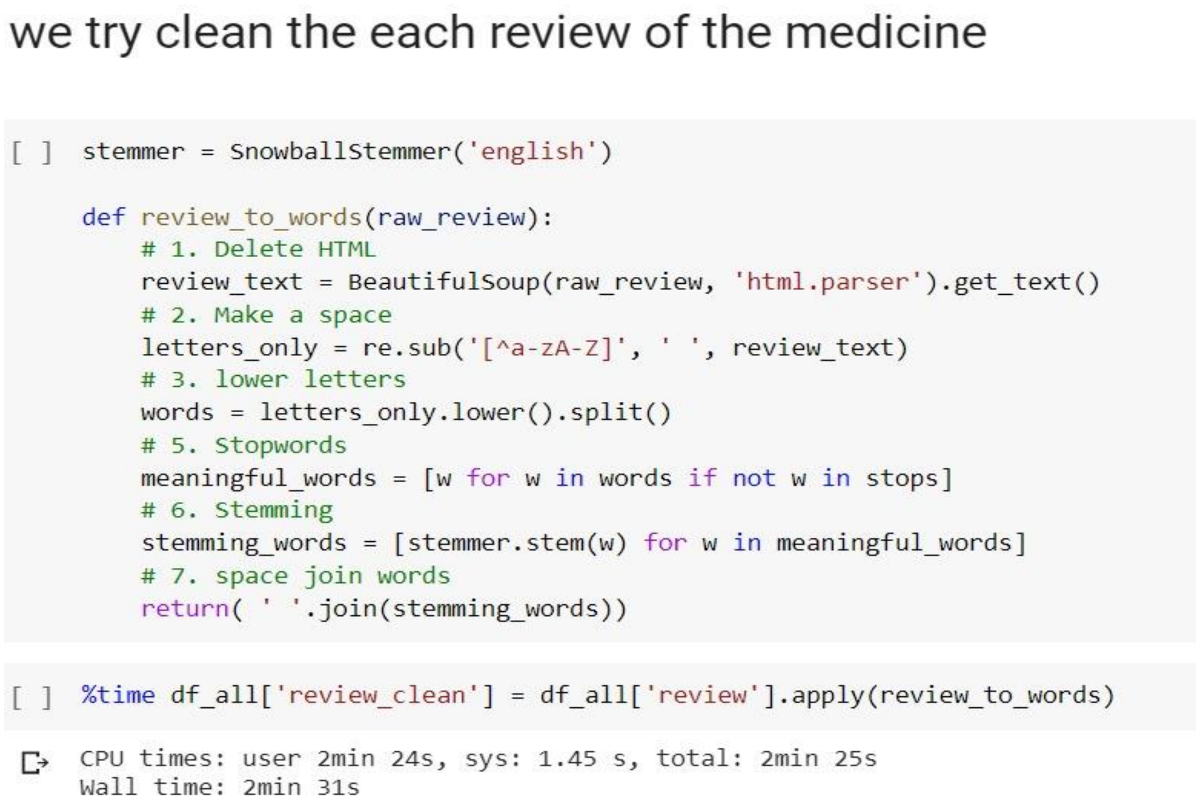
Removal of rows having only one drug per disease:

In Earlier analysis we saw that if on condition has only one drug then there won't be any alternative for the patient to choose, if the drug doesn't work fine.

If a condition has only one medicine in the dataset and the medicine is not suggestible then that kind of data will be useless i.e., we atleast need two drugs per disease to differentiate between them and prescribe a medicine among them.

So, we make sure that there will be atleast two drugs per condition when a person searches for a cure to some disease.

Review statement pre-processing:
As we have seen earlier in exploratory analysis the statement wasn't purely a sentence it has some random symbols, capital letters and some html tags (</>) kind of symbols. So, remove the html tag kind symbols using BeautifulSoup which removes the HTML tags and symbols from a sourced text, filters the words and converts into simple sentence. Then we add space between words and convert Capital case letters into small case letters. Then we extract meaningful words and convert into root word using stemmer and construct final cleaned review statement.



we try clean the each review of the medicine

```
[ ]  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))
```

```
[ ]  %time df_all['review_clean'] = df_all['review'].apply(review_to_words)
```

```
CPU times: user 2min 24s, sys: 1.45 s, total: 2min 25s
Wall time: 2min 31s
```

Fig.19 Procedure for preprocessing the review statement.

So, now we have cleaned the review statements as pictured above.

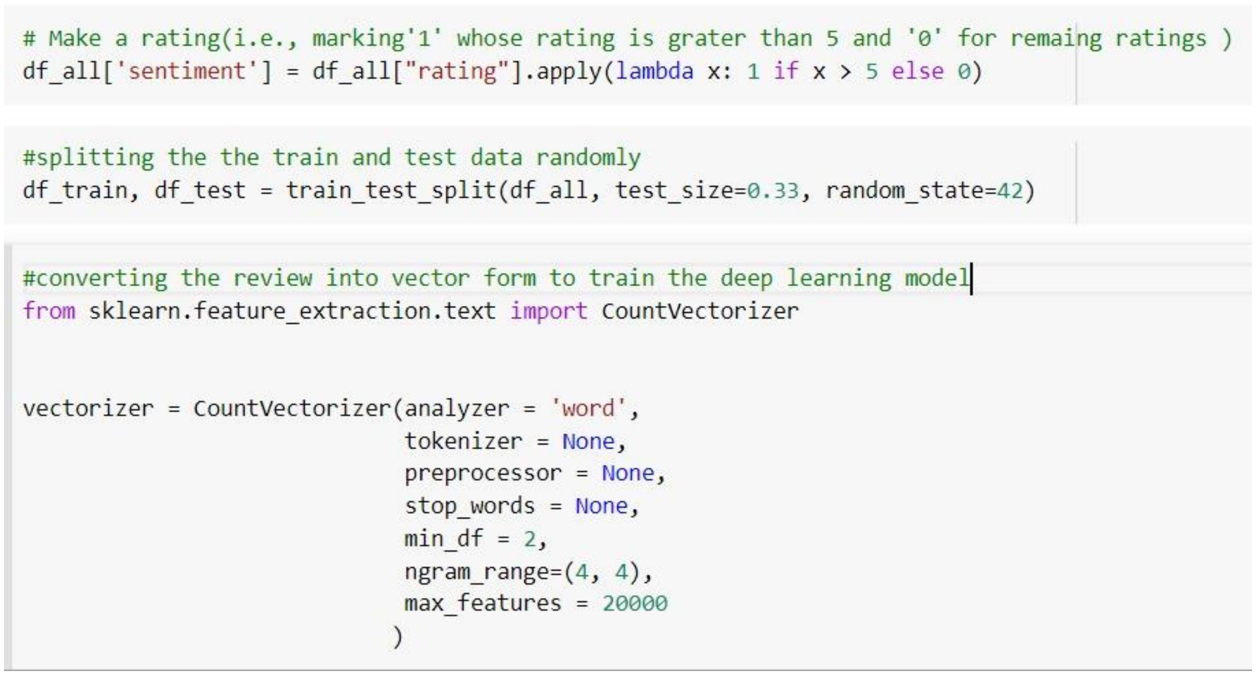
4.2.4 MODEL BUILDING

Deep-learning model: The rating given in rating column is 10 star-rating so we try to convert this into a normalized format, we classify the ratings between 6-10 as positive i.e. 1 in numerical format and 1-5 as negative i.e. 0 in numerical format and we store these normalized 1,0 values for respective rows in 'sentiment' column.

We have two datasets Train_Data and Test_Data, we will merge those datasets into one dataset and call it as df_all and train_test_data contains Train_Data 67% and TestData 33%, it is picked by random sampling from df_all, so that we will be training with randomly picked data which will be efficient and by which we can avoid the biased outputs as here we are picking up the values by random sampling (i.e. taking up the data randomly).

We can't train text as text. So, we convert in to numerical format as mentioned below

Count Vectorization: we count vectorize the sentence like how many times a word(token i.e. token is given token number) has been repeated and store as term frequency of tokens and in n-gram model we take 'n' as 4-gram because we can differentiate better between 1-5 rating and 6-10 rating much easier as observed in exploratory data analysis. We even consider stop words also. Convert it as a Vector of tokens and its term frequencies and passed as input to deep learning model.

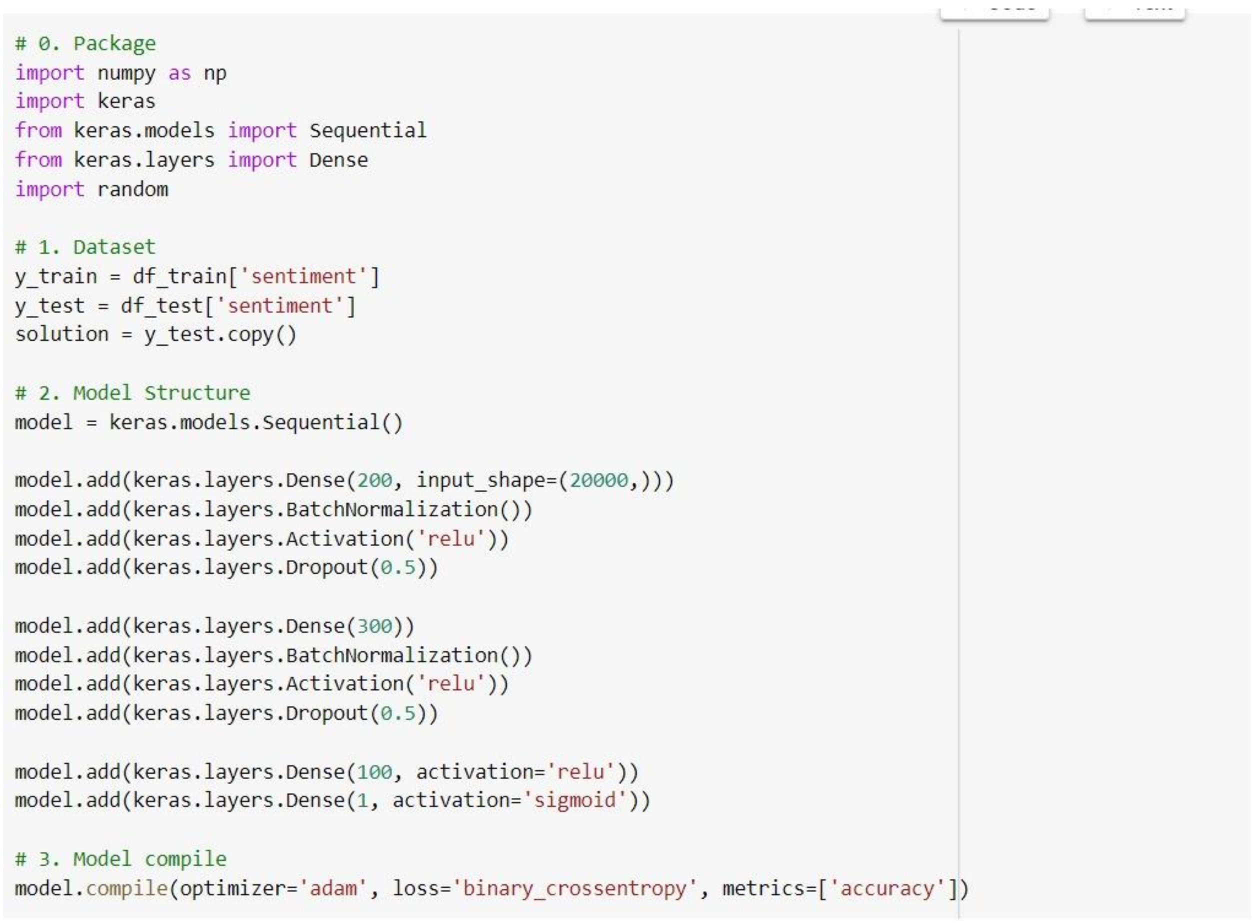

```
# Make a rating(i.e., marking'1' whose rating is grater than 5 and '0' for remaing ratings )
df_all['sentiment'] = df_all["rating"].apply(lambda x: 1 if x > 5 else 0)
```

```
#splitting the the train and test data randomly
df_train, df_test = train_test_split(df_all, test_size=0.33, random_state=42)
```

```
#converting the review into vector form to train the deep learning model
from sklearn.feature_extraction.text import CountVectorizer


vectorizer = CountVectorizer(analyzer = 'word',
                             tokenizer = None,
                             preprocessor = None,
                             stop_words = None,
                             min_df = 2,
                             ngram_range=(4, 4),
                             max_features = 20000
                            )
```

Fig.20 Procedure for countvectorizing the review statement.

Building deep learning model: Here we are constructing a neural network which consists of 4 layers, as we have seen max features as 20000 we will take the shape of input to be 20000 columns i.e. (20000,*),first three layers activation function is reLU and for the fourth layer i.e. output layer we take sigmoid function as activation function. Here we are applying Dropout to first two hidden layers inorder to avoid the over fitting of model. Here we adding batch-normalization layer to the first two layers by which the output of the previous layers will be normalized such that it will improve the performance of the model. With above attributes we have constructed our deep learning model. Now we will train this deep learning model with our dataset inputs.



```
# 0. Package
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense
import random

# 1. Dataset
y_train = df_train['sentiment']
y_test = df_test['sentiment']
solution = y_test.copy()

# 2. Model Structure
model = keras.models.Sequential()

model.add(keras.layers.Dense(200, input_shape=(20000,)))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Activation('relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(300))
model.add(keras.layers.BatchNormalization())
model.add(keras.layers.Activation('relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(100, activation='relu'))
model.add(keras.layers.Dense(1, activation='sigmoid'))

# 3. Model compile
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Fig.21 Construction of deep-learning model using keras.

We will train our model up to 10 epochs and we can see that losses decrease with each epoch and corresponding accuracy is increasing with each epoch from this we can conclude that our model is working fine.

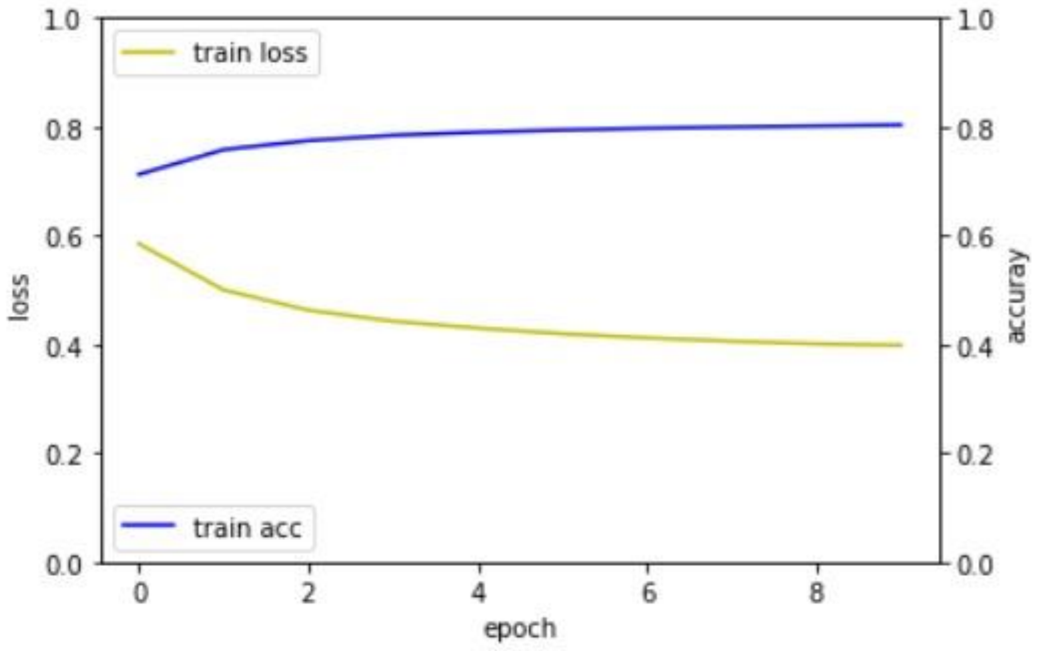
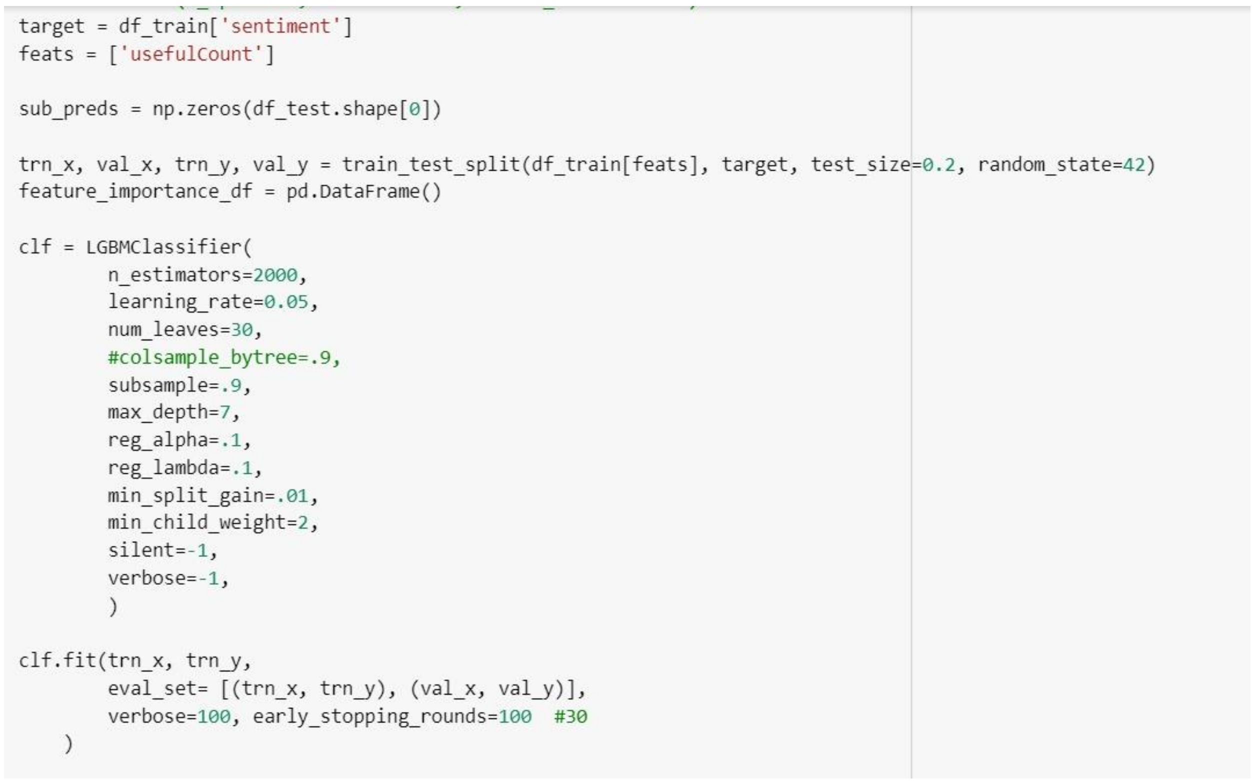


Fig.22 Graph showing pattern of loss and accuracy against epochs.

We observed the final accuracy rate to be 80.32% so we will be trying to increase the accuracy in further training by taking machine learning models.

LightGBM Classifier model construction: To improve the accuracy of the previous model we use machine learning model, So we will try to use LightGBM because LightGBM is the latest high performance machine learning model, here we can have the leverage of GPU's, this is mainly used for large datasets as our dataset is very large we prefer LightGBM classifier.

First, we are trying to build the model by considering only one attribute i.e. useful_count later we will add other attributes. So, here we are considering the number of leaves to be thirty which must be less than 2^(max_depth) and also the number leaves to be 30 is a optimal value, now here we are considering the regularization alpha lambda which are the l1 and l2 regularizations used to avoid the overfitting of the model. The test size is 20% and train size is 80% of the total datasets. So, we have considered the above hyper parameters and imputed the above values of hyper parameters in the above model. Here we are considering the parameter "early_stopping_rounds" which helps in avoiding the overfitting of the model. By stopping the further iterations if there is no improvement in the learning.



```
target = df_train['sentiment']
feats = ['usefulCount']

sub_preds = np.zeros(df_test.shape[0])

trn_x, val_x, trn_y, val_y = train_test_split(df_train[feats], target, test_size=0.2, random_state=42)
feature_importance_df = pd.DataFrame()

clf = LGBMClassifier(
        n_estimators=2000,
        learning_rate=0.05,
        num_leaves=30,
        #colsample_bytree=.9,
        subsample=.9,
        max_depth=7,
        reg_alpha=.1,
        reg_lambda=.1,
        min_split_gain=.01,
        min_child_weight=2,
        silent=-1,
        verbose=-1,
        )

clf.fit(trn_x, trn_y,
        eval_set= [(trn_x, trn_y), (val_x, val_y)],
        verbose=100, early_stopping_rounds=100  #30
    )
```

Fig.23 Construction and Training of LGBM Classifier model.

Now we train the model

So, we have considered only one column i.e. useful Count. Now we will add other variables to improve the accuracy. So, we divide date column into day, month and year column to be precise about the date so that it will easier to consider each and every factor.

We will apply the Sentiment function of Textblob on the cleaned review in the review_clean column which we have pre-processed previously. Textblob return polarity which means it decides whether the statement is positive or negative if yes it also tells how much percentage it is positive or negative.

Generally, the range is between -1 to +1. Now we are storing this polarity value in the new column named as 'Predict_Sentiment'. After that we are trying to find the correlation of Predict_Sentiment column with rating column and Sentiment column which we have obtained as 25.7% with rating column and 23.5% Sentiment column.

Similarly, we perform the Textblob Sentiment function to the uncleaned review statement also and we store the polarity in the new column Predict_Sentiment2. Now once again we will try to find the correlation Predict _Sentiment2 column with rating column and sentiment column which we have obtained as 34.8 with rating column and 31.7 with sentiment column.

So, as we have observed that the predict_sentiment2 has higher correlation with rating and sentiment columns, so, we consider this also as a feature.

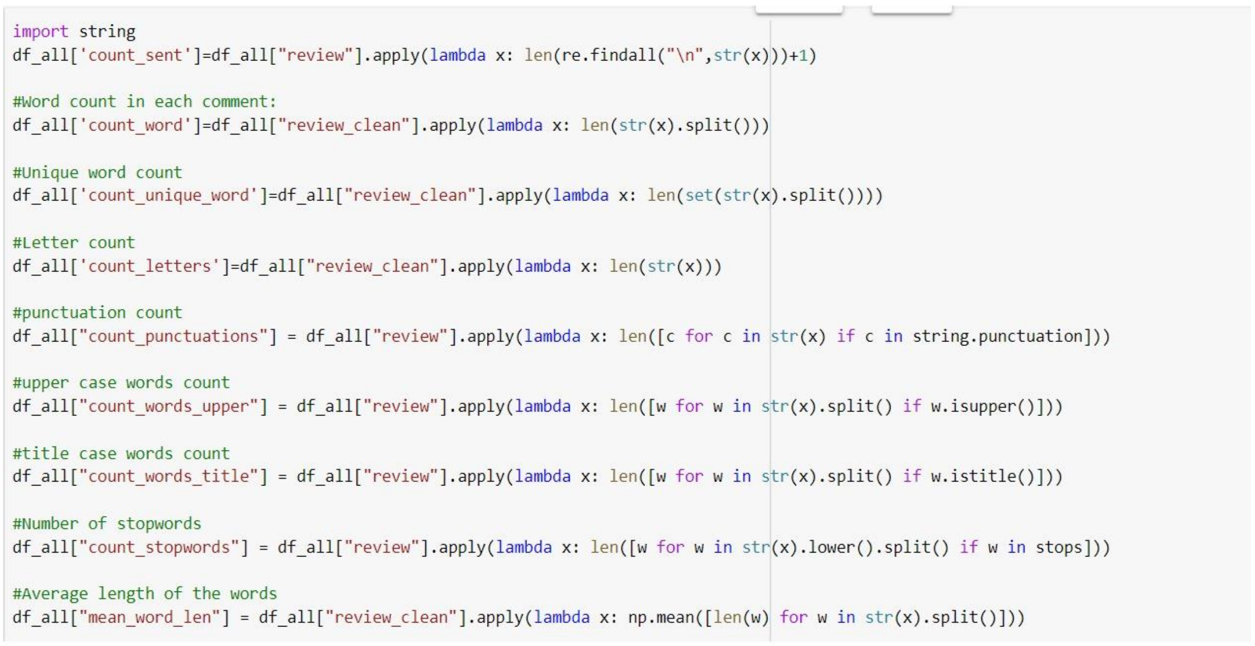
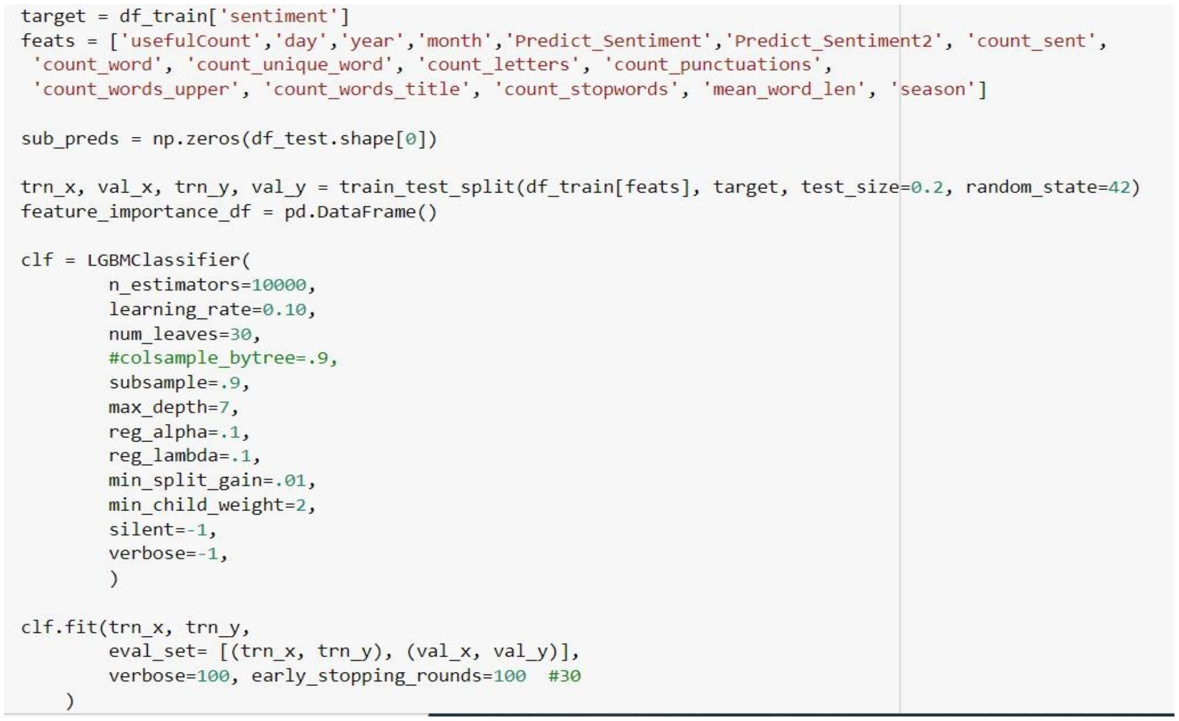

```
import string
df_all['count_sent']=df_all["review"].apply(lambda x: len(re.findall("\n",str(x)))+1)

#Word count in each comment:
df_all['count_word']=df_all["review_clean"].apply(lambda x: len(str(x).split()))

#Unique word count
df_all['count_unique_word']=df_all["review_clean"].apply(lambda x: len(set(str(x).split())))

#Letter count
df_all['count_letters']=df_all["review_clean"].apply(lambda x: len(str(x)))

#punctuation count
df_all["count_punctuations"] = df_all["review"].apply(lambda x: len([c for c in str(x) if c in string.punctuation]))

#upper case words count
df_all["count_words_upper"] = df_all["review"].apply(lambda x: len([w for w in str(x).split() if w.isupper()]))

#title case words count
df_all["count_words_title"] = df_all["review"].apply(lambda x: len([w for w in str(x).split() if w.istitle()]))

#Number of stopwords
df_all["count_stopwords"] = df_all["review"].apply(lambda x: len([w for w in str(x).lower().split() if w in stops]))

#Average length of the words
df_all["mean_word_len"] = df_all["review_clean"].apply(lambda x: np.mean([len(w) for w in str(x).split()]))
```

Fig.24 Process of detailed analysis of review statement and storing the results as separate attributes.

Now we will go into the detailed analysis of the review statement and store the results as separate attributes (‘count_word’ means counting the number of words in each comment., etc.,) as shown in the above figure.

We are doing the above process so that each and every factor contributing towards the result is considered which will increase the performance of the model. We will also try to add a season variable where like if the month is between 2 to 6, we consider the category as 1 in the season, 5 to 9 as 2, 9 to 12 as 3 and remaining as 4 i.e. we are also considering the seasons like summer, winter, etc.

Now considering all these attributes as features now we will train lgbm classifier model with these attributes as our features.



```
target = df_train['sentiment']
feats = ['usefulCount','day','year','month','Predict_Sentiment','Predict_Sentiment2', 'count_sent',
 'count_word', 'count_unique_word', 'count_letters', 'count_punctuations',
 'count_words_upper', 'count_words_title', 'count_stopwords', 'mean_word_len', 'season']

sub_preds = np.zeros(df_test.shape[0])

trn_x, val_x, trn_y, val_y = train_test_split(df_train[feats], target, test_size=0.2, random_state=42)
feature_importance_df = pd.DataFrame()

clf = LGBMClassifier(
        n_estimators=10000,
        learning_rate=0.10,
        num_leaves=30,
        #colsample_bytree=.9,
        subsample=.9,
        max_depth=7,
        reg_alpha=.1,
        reg_lambda=.1,
        min_split_gain=.01,
        min_child_weight=2,
        silent=-1,
        verbose=-1,
        )

clf.fit(trn_x, trn_y,
        eval_set= [(trn_x, trn_y), (val_x, val_y)],
        verbose=100, early_stopping_rounds=100  #30
    )
```

Fig.25 LGBM Classifier model after considering all the attributes obtained from the analysis.

We train our model with the features which selected above and hyper parameters discussed before. After training we have observed the below following features as more important.

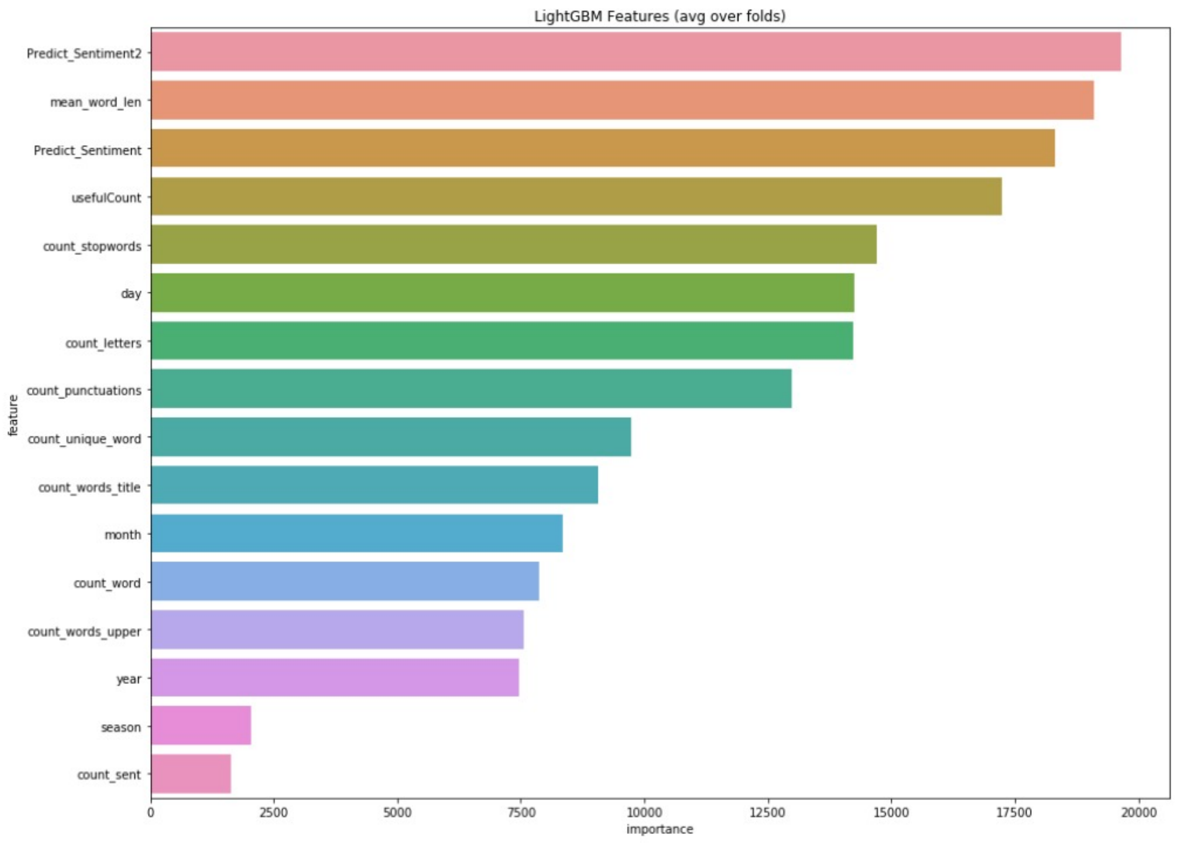
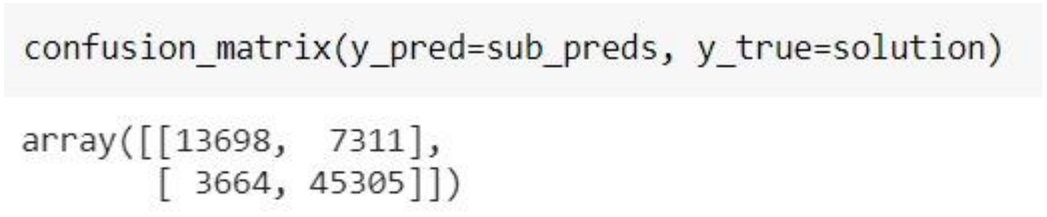


Fig.26 Graph showing the important features obtained during training of LGBM Classifier model.

So, the final confusion matrix of our model is below.



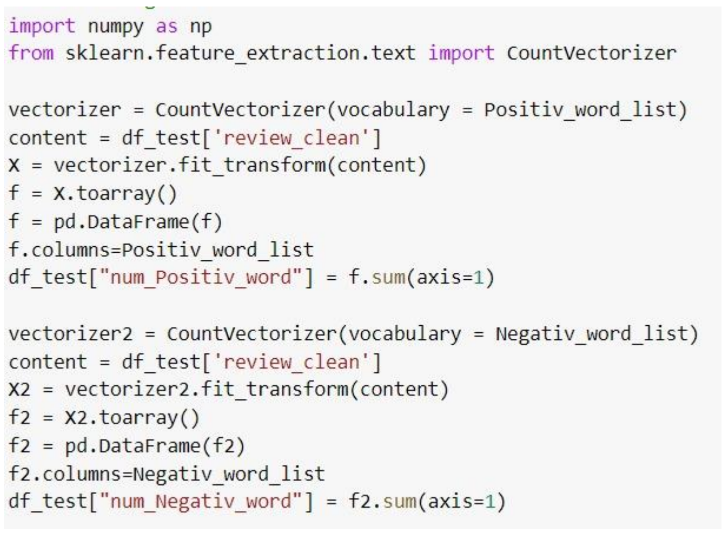
```
confusion_matrix(y_pred=sub_preds, y_true=solution)
```

```
array([[13698,  7311],
       [ 3664, 45305]])
```

Fig.27 Confusion matrix inorder to find the accuracy of the model.

From the above confusion matrix, we have obtained the accuracy as 84.3%.

Dictionary Sentiment Analysis: Inorder to improve the more emotional analysis of the review statement we try to use additional emotional analysis using emotional dictionary of Harvard University. The dataset of this emotional dictionary is inquirer basic.csv, this dataset consists the emotion of a particular word like whether it positive or negative, it consists columns entry, source, positive, negative. We will try to make the list of sentiment i.e. positive word list and negative word list. Now what we do is map the words of positive word list and negative word list with words in review clean using Count Vectorizer and store the value of number of positive words and number of negative words under the column num_Positv_word and under the column num_Negativ_word respectively.

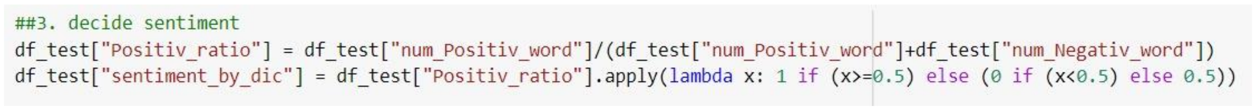


```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary = Positiv_word_list)
content = df_test['review_clean']
X = vectorizer.fit_transform(content)
f = X.toarray()
f = pd.DataFrame(f)
f.columns=Positiv_word_list
df_test["num_Positiv_word"] = f.sum(axis=1)

vectorizer2 = CountVectorizer(vocabulary = Negativ_word_list)
content = df_test['review_clean']
X2 = vectorizer2.fit_transform(content)
f2 = X2.toarray()
f2 = pd.DataFrame(f2)
f2.columns=Negativ_word_list
df_test["num_Negativ_word"] = f2.sum(axis=1)
```

Fig.28 Construction of positive and negative words list from the dictionary.

We find the positive ratio of each review statement i.e. number of positive words / total number of words. So, with this ratio we will decide the sentiment of the review statement as follows...

1. If positive ratio is greater than or equal to 0.5, we classify it as positive.
2. If it is less than 0.5, we classify it as negative.
3. other than the above is considered as neutral.



```
##3. decide sentiment
df_test["Positiv_ratio"] = df_test["num_Positiv_word"]/(df_test["num_Positiv_word"]+df_test["num_Negativ_word"])
df_test["sentiment_by_dic"] = df_test["Positiv_ratio"].apply(lambda x: 1 if (x>=0.5) else (0 if (x<0.5) else 0.5))
```

Fig.29 Deciding the sentiment of the statement from the calculation of positive ratio.

Now we will try to normalize the usefulcount column because there can be bias in the score given to that particular condition which can be given by some consumers who are used to some specific medicines. So, we will try to normalize the usefulcount.

Now finally our overall predicted value will the sum of sentiment values predicted by the deep learning model, say x1, machine learning model i.e. LightGBM classifier, say x2 and Sentiment analysis by Dictionary, say x3, whole multiplied by normalized usefulcount, y1.

i.e. (x1+x2+x3) * y1.

5.RESULTS AND CONCLUSIONS:

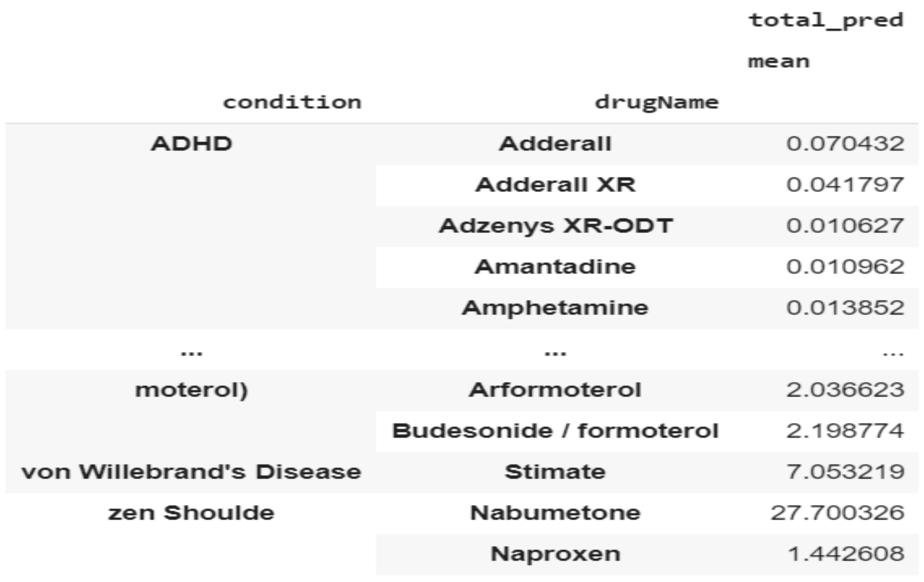
Our topic is to prescribe medicine using knowledge graph and sentiment analysis of review statements.

Knowledge graph creation and finding the entity pairs:

We find entity pairs i.e. name and disease, from the input statement using knowledge graph and we identify the disease from that.

Sentiment Analysis of Review Statements for a particular disease:

Now we will prescribe the right medicine for patient's condition using review statements, For Sentiment analysis we do the exploratory analysis of the data to find the importance of it, during data analysis we try to look all the data using data visualization and statistical techniques. We found the N-gram which is best to classify the emotions. We found out relation between date and rating. After that we did the data pre-processing part in which we try to remove the conditions which recommends only one drug per disease and we try to clean the review statement after that we go through the model construction. In model construction process we try to construct a deep learning model with four grams and also we used machine learning model using Light GBM to improve accuracy. In addition to this we also constructed a emotional analysis using Emotional word Dictionary of Harvard University inorder to understand the emotions of the words in the statement more accurately. In addition to this we also normalised the usefulcount normalised by condition for better performance. So .by doing all this process we finally try calculate the predicted value and recommend the best drug among the drugs available in the dataset for particular condition using this predicted value.



		total_pred
		mean
condition	drugName	
ADHD	Adderall	0.070432
	Adderall XR	0.041797
	Adzenys XR-ODT	0.010627
	Amantadine	0.010962
	Amphetamine	0.013852
...	...	...
moterol)	Arformoterol	2.036623
	Budesonide / formoterol	2.198774
von Willebrand's Disease	Stimate	7.053219
zen Shoulde	Nabumetone	27.700326
	Naproxen	1.442608

Fig.30 Figure showing the final prediction of medicines for a disease.

Limitations:

While doing Sentiment Analysis using sentiment word dictionary if there are less number of positive and negative words it will have low reliability because if we consider the sentence with only zero positive words and on negative words it would be classified as negative which is not feasible, therefore if the number of sentiment words are less than five then we may not consider such observations.

As we have normalized the usefulcount and multiplied it with the sum of predicted values for our results but here we may also need to consider the time and normalizing the useful count because usefulcount may be higher for older reviews because the number of site visitors increases. This might be the factor which also have to be considered while normalizing the usefulcount.

The sentiment of the review statement is depended upon the emotion, when the emotion is positive the sentiment of the review statement would incline more towards positive nature and if it is negative then it would be more inclined towards negative nature. Instead of simply multiplying the usefulcount with predicted value we could have multiplied with the sign of useful count according to the nature of the emotion.

REFERENCES

Shown as [number] in report

1 https://digital.library.unt.edu/ark:/67531/metadc863008/m2/1/high_res_d/1119939.pdf?cv=1

2 https://www.tandfonline.com/doi/abs/10.1080/12460125.2015.1087293?cv=1&journalCode=tjds20

3 https://id.123dok.com/document/z1l388vq-an-analysis-of-register-in-business-page-of-e-jakarta-globe-iain-syekh-nurjati-cirebon.html?cv=1

4 https://www.slideshare.net/akshatapandey/morphological-analysis-47051109?cv=1

5 https://www.freecodecamp.org/news/an-introduction-to-part-of-speech-tagging-and-the-hidden-markov-model-953d45338f24/?cv=1

6 https://www.basicknowledge101.com/subjects/knowledgemanagement.html?cv=1

7 https://genylabs.io/the-ultimate-glossary-of-online-reputation-management-terminology/?cv=1

8 https://onlinelibrary.wiley.com/doi/book/10.1002/9781119720430?cv=1

9 https://thenextweb.com/podium/2019/06/11/what-is-a-knowledge-graph-and-how-does-one-work/

10 https://monkeylearn.com/sentiment-analysis/?cv=1

11 https://docs.python.org/2/library/index.html?cv=1

12 https://www.springpeople.com/blog/all-you-need-to-know-about-anaconda-distribution-for-python/?cv=1

13 https://problemsolvingwithpython.com/01-Orientation/01.03-Installing-Anaconda-on-Windows/?cv=1

14 https://docs.anaconda.com/anaconda/navigator/?cv=1

15 https://hackernoon.com/data-science-toolkit-concepts-code-20628af23cd?cv=1

16 https://numpy.org/?cv=1

17 https://www.guru99.com/download-install-nltk.html?cv=1

18 https://pythonsecret.blogspot.com/2016/11/?cv=1

19 https://keras.io/

20 https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

These websites were referred when we faced any problems or doubts in knowledge graph and machine learning.

https://developers.google.com/

https://stackoverflow.com/

https://towardsdatascience.com/

https://www.analyticsvidhya.com/

https://www.geeksforgeeks.org/machine-learning/