

**DATA 698 Research Paper**  
**Topic: Student Counselling Software**

**Irene Jacob**

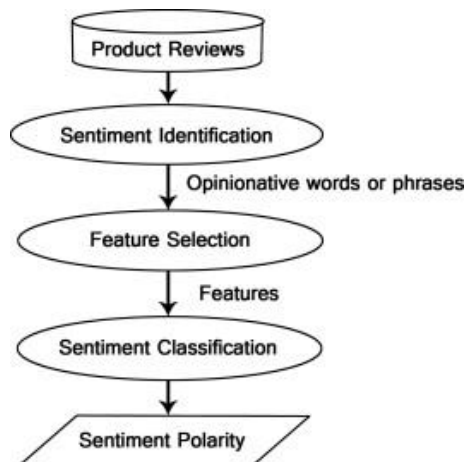
**Abstract**

Sentiment analysis refers to the use of natural language processing, text analysis and computational linguistics to identify and extract subjective information in source materials. Sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document.

Students encounter problems such as heavy study load, lack of social engagement, and sleep deprivation. Chronic student problems can have lasting effects. Student counselling is used to encourage conversation among students. It can also help to process feelings of being excluded or rejected. Students may feel supported and encouraged to manage changes with the help of the student counselling software.

Sentiment Analysis (SA) is the computational study of people's opinions, attitudes, and emotions towards an entity. The entity can represent individuals, events, or topics. These topics are most likely to be covered by reviews.

Sentiment Analysis extracts and analyses people's opinion about an entity by identifying the sentiment expressed in a text. Therefore, the target of SA is to find opinions, identify the sentiments they express, and then classify their polarity.



The social network sites and micro-blogging sites are considered a very good source of information because people share and discuss their opinions about a certain topic freely. They are also used as data sources in the SA process.

There are many applications and enhancements on SA algorithms that were proposed in the last few years.

**Keywords**

Sentiment Analysis, Tweet extraction, Tokenization, N Gram, Stemming

## Literature Review

Large amount of unstructured data is generated from social media, which is classified as positive, negative, or neutral. Positive sentiments are those which contain appreciation for other's tweets, movies, products, etc., while negative sentiments are those which contain bad words or dissatisfactory comments on any product, movie, event, etc. Whereas neutral sentiments are neither positive nor negative sentiment. Different approaches of clustering based on Sentiment Analysis presents a way to find relationships between the tweets based on polarity and subjectivity. For efficient data analysis lexicon-based approach is more efficient. (Deshapande et al., 2019 [1] and Gupta et al., 2017 [3])

Machine learning is the study of algorithms that can learn from and make predictions on data. A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature and this algorithm is very easy to build up and mainly used for a large set of data. On the other hand, a neural network has two phases, which are training and testing. In training phase, the positive and negative comments are trained and assigned weights. The main purpose of training phase is to create the dictionary of positive comments. In the next phase testing is done based on the weighted dictionary. The artificial neural network is trained with labelled data to produce meaningful output. This process by which neural networks learn from labelled data is called back propagation (SIDDHARTH et al., 2018) [2].

Java and other open APIs can be used to create an application which amalgamates the data curation service, knowledge extraction service, user profile building service, and filter engine into any proposed recommendation system. By applying seed list-based classification and sentiment analysis, the system will be able to recommend personalized tweets to users. In order to overcome redundancy problems and formatting issues, Google Refine can be used (Khattak et al., 2020) [4].

Twitter is in a lot of fame at present because of its specific format of writing. Few characteristics of tweets are that they are short messages consisting of maximum 140 characters, approximately 1.2 billion tweets are posted daily, tweets are posted by a wide variety of people, thus the topics discussed is also variant and could almost include any topic starting from politics to mobile products etc. (Faizan, 2019) [5]

Experiments for two classification tasks, namely, positive versus negative and positive versus negative versus neutral can be carried out and for each of the classification tasks three models may be presented and the results for two combinations of these models. For the unigram plus Senti-features model, a feature analysis shall be done to gain insight about what kinds of features are adding most value to the model and the learning curves for each of the models and compare learning abilities of models when limited data is available. All the experiments shall use Support Vector Machines (SVM) and report averaged 5-fold cross-validation test results. Upon investigation of other two models, namely, tree kernel and feature based models shows that both these models outperform the unigram baseline because the most important features are those that combine the prior polarity of words and their parts-of-speech tags. These experiments reveal that sentiment analysis for Twitter data is not that different from sentiment analysis for other genres. (Agarwal et al., 2011) [6]

Machine learning techniques: for example, the Support Vector Machine (SVM) and Multinomial Naive Bayes (MNB) produce the greatest precision, especially when multiple

features are included. SVM classifiers may be viewed as standard learning strategies, while dictionary (lexicon) based techniques are extremely viable at times, requiring little efforts in the human-marked archive. Machine learning algorithms, such as The Naive Bayes, Maximum Entropy, and SVM, achieve an accuracy of approximately 80% when n-gram and bigram model is utilized. Ensemble and hybrid-based Twitter sentiment analysis algorithms tend to perform better than supervised machine learning techniques, as they are able to achieve a classification accuracy of approximately 85%. In general, it is expected that ensemble Twitter sentiment-analysis methods would perform better than supervised machine learning algorithms, as they combined multiple classifiers and occasionally various feature models. However, hybrid methods also perform well and obtain reasonable classification accuracy scores (Alsaedi & Khan, 2019) [7].

For example, a dataset of different Arabic dialects, which consists of over 151,500 tweets/comments, was collected and automatically labelled. The NB, LR, ME, PA, RR, SVM, MNB, Ada-Boost BNB, and SGD classifiers were used to extract and discover the polarity of a given tweet. A 10-fold cross validation was utilized to divide the data into a separated training set and testing set. The best evaluation metric values of 99.96% was achieved by PA and RR using unigram, bigram, or trigram. (Gamal et al., 2019) [8]

## Methodology

Today students face various problems like stress, homesickness, exam fear and so on. This affects their studies as well as makes their life sad and stressful. But in most institutions, there is only one counsellor for the entire student body. With the help of a student counselling software, that is freely and easily accessible with a click of a button, students will be able to identify their issues and problems.

The primary objective of this project is to develop counselling software using sentiment analysis. This can help reduce tensions among students and make school a happy abode.

The secondary objective of this project is to learn programming in python, use of NLTK (Natural Language Tool Kit) and gain real world experience in developing a project.

1. To know and understand the comparison between accuracy/results of different data mining techniques
2. To find out why twitter is the most preferred platform compared to other social media platforms like Facebook or Instagram for data mining
3. Why is python useful for analysis such as sentiment analysis?

The software takes the text the user enters describing his issues as the input. It then processes this text to identify the problems faced by the student to provide counsel.

Here sentiment analysis is used to build student learning. It is to be done in python using NLTK. A person in need of counselling enters his/her grievances in a paragraph or two. This text acts as the input to the program. The program is expected to generate the necessary advice for the client by identifying his problem (say home sickness) from the input. The model is trained with a training set of data obtained from twitter and is expected to produce output when it encounters a new input.

The first stage is about processing natural language, and the second about training the model. The first stage processes text in a way that when we are ready to train our model, we already know what variables the model needs to consider as inputs. The model itself is responsible for learning how to determine the sentiment of a piece of text based on these variables. Output of this software will be the prediction of the cause or root of the described issue (text input to software from student).

## **Experimentation and Results**

### **I. Fetching data from twitter**

Twitter is a collection of large datasets. For performing the sentiment analysis on twitter data, the data extraction processing is important. Compared to the other networking sites twitter allows users to share their views openly. With the help of twitter API, twitter gives the expansive access to tweets.

The tweets are collected using Twitter's streaming API, or any other mining tool (for example WEKA), for the desired timeframe of analysis. The twitter app is created to access the twitter developer account which has same username and password. Using these credentials, the string form of tweets can be obtained from API. [1]

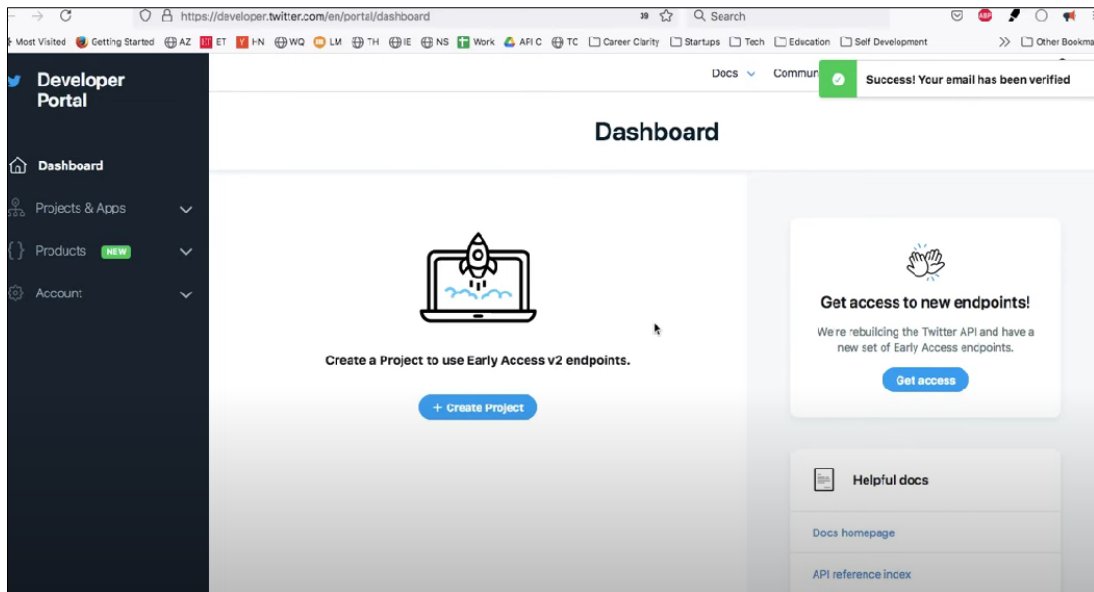
The format of the retrieved text is converted as per convenience (for example JSON). The dataset collected influences the efficiency of the model. The division of dataset into training and testing sets is also a deciding factor for the efficiency of the model. The results depend on the training set. [3]

### **II. Searching for Tweets with Python**

#### **Step 1: Set up your twitter developer account**

Firstly, Twitter developer account needs to be set up. The below steps can be used to create such an account.

1. Navigate to <https://developer.twitter.com/en/apply-for-access> and sign in with your twitter account if you already have one. If not, sign up for Twitter with a new account.
2. Click on "Apply for a developer account". This opens a dialogue where you will be asked how you want to use the Twitter API.
3. Describe your intended use: Subsequently, you will be forwarded to a page where you have to state your intended use of your work with the Twitter API. In total, you need to write about 200–600 characters depending on what you intend to do.
4. Review your access application and read the terms.
5. Set up a project and application: Navigate to the below link where you can set up your Twitter API project and an application.  
<https://developer.twitter.com/en/portal/dashboard>
6. Copy your bearer token of the application — you will be needing this token shortly. If you can't remember your Bearer Token, navigate to "Keys and tokens" and click on regenerate.



## Step 2: Python environment setup

To send API requests in Python and to be able to use the Twitter API, Twitter wrapper modules will not be considered but it will be used on the very handy requests module which can be installed via pip:

pip install requests

## Step 3: Prepare Search Twitter Function

The function is short and needs three parameters:

1. bearer\_token [str]: the bearer token copied in Step 1.
2. query [str]: This is the actual string that will be used for matching the desired Tweets. These query strings can be simple such as “family issues”, “exam stress” or very complex and powerful. They allow you to filter tweets by hashtags, identify retweets, exclude certain words or phrases, or only include tweets of a specific language.
3. tweet\_fields [str]: Fields to return in the query, such as attachments, author\_id, text, etc. For example, tweet.fields=text,author\_id,created\_at

Optional parameters are as follows:

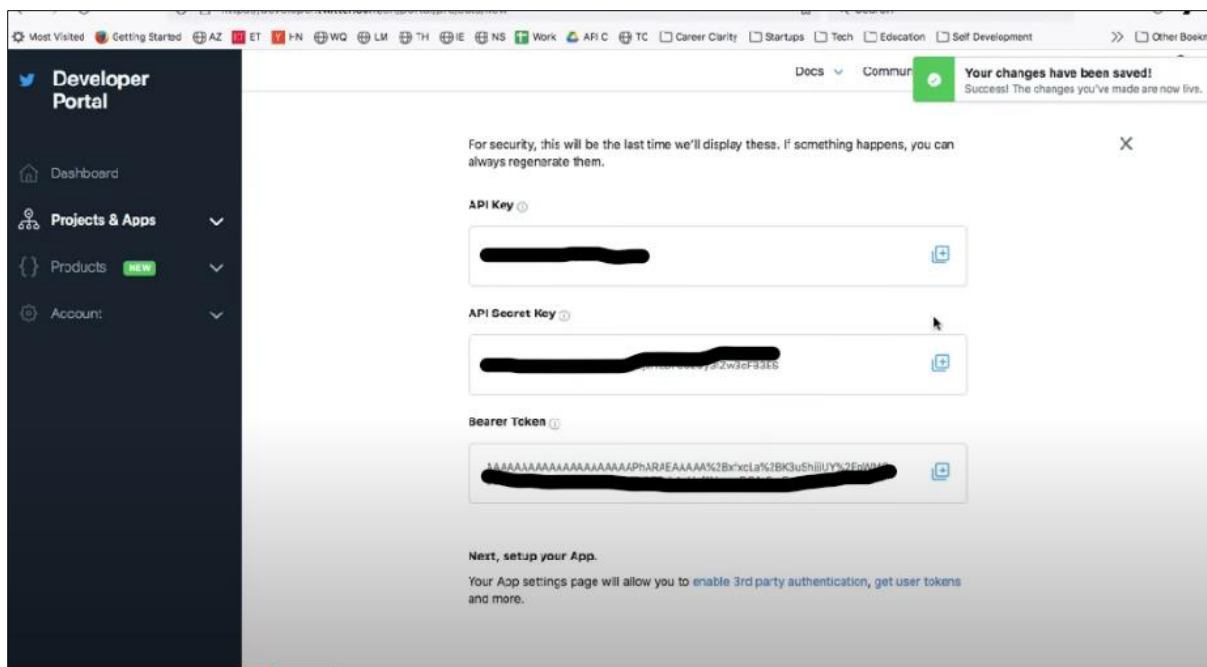
1. max\_results: parameters that specify how many tweets should be returned. By default, a request response will return 10 results.
2. start\_time: The oldest UTC timestamp (from most recent seven days) from which the Tweets will be provided. (YYYY-MM-DDTHH:mm:ssZ (ISO 8601/RFC 3339)).
3. end\_time: The newest, most recent UTC timestamp to which the Tweets will be provided. (YYYY-MM-DDTHH:mm:ssZ (ISO 8601/RFC 3339)).

## Step 4: Run search\_twitter function

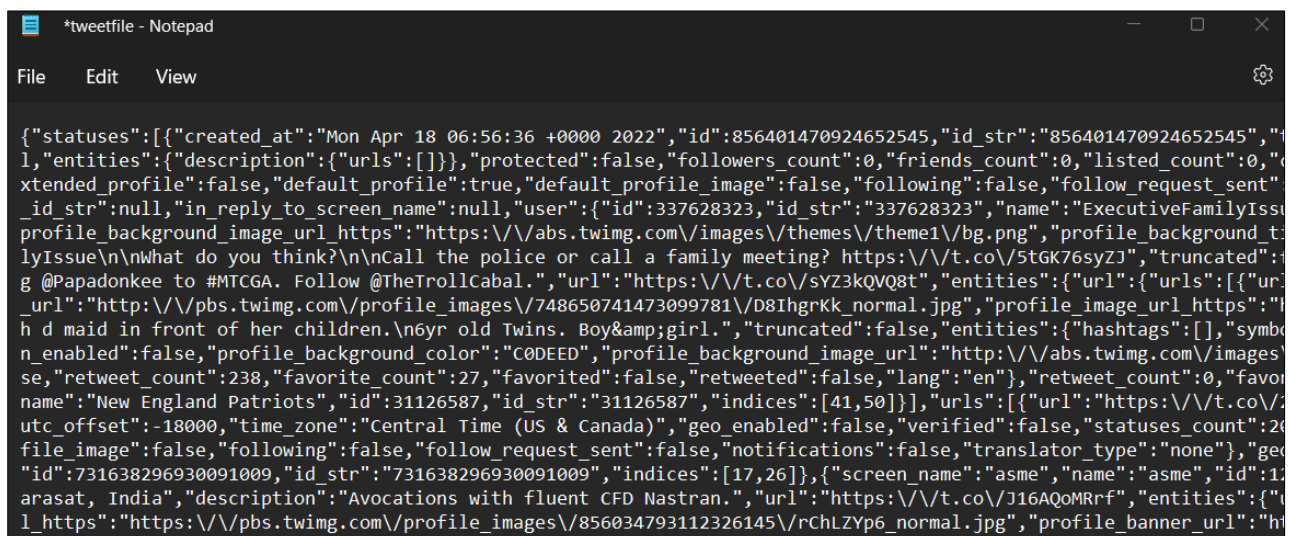
Data is extracted manually from twitter using some API available for twitter. Tweepy is the API used but it is not compatible with the new versions of python. So, for using this API an older version of python is needed (python 2.7). To access tweets on twitter using API, the first

step is to authenticate the console from which twitter's tweets can be accessed. This is done by following the steps listed below:

1. Creation of a twitter account.
2. Logging in at the developer portal of twitter.
3. Select "New App" at developer portal.
4. A form for creation of new app appears.
5. After this the app for which the form was filled out will go for review by twitter team.
6. Once the review is complete and the registered app is authorized then the user is provided with 'API key' and 'API secret'.
7. After this "Access token" and "Access token secret" are given.



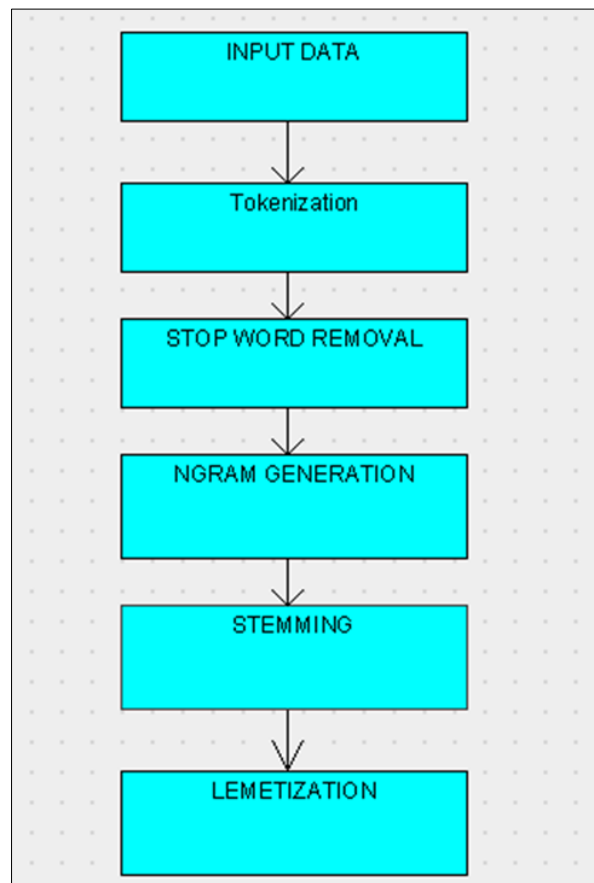
These keys and tokens are unique for each user and only with the help of these one can access the tweets directly from twitter. [5]



To obtain the tweets relevant to the data collection of this project, the following URL is used from Twitter API:

<https://api.twitter.com/1.1/search/tweets.json?q=studentlife>

### III. Data Pre-processing



#### 1. Tokenization

Input data for training is collected from twitter and is stored in the input file. Tokenization is used to split a stream of text into smaller units called tokens (words or phrases). The tokenization is based on regular expressions (regex). Some types of tokens (eg. phone numbers or chemical names) will not be captured and will be broken into several tokens. To overcome this problem, as well as to improve the richness of the pre-processing pipeline, the regular expressions can be improved. The core component of the tokenizer is the `regex_str` variable, which is a list of possible patterns. The `tokenize()` function catches all the tokens in a string and returns them as a list. This function is used within `preprocess()`, which is used as a pre-processing chain.[2]

#### 2. Removing stop words

One of the major forms of pre-processing is filtering out useless data. In natural language processing, useless words are referred to as stop words. It is not desirable to have these words taking up space in our database or taking up processing time. Stop words are words that just contain no meaning and needs to be removed. This can be done by storing a list of stop words.



A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, when indexing entries for searching and when retrieving them as the result of a search query.

### 3. N Gram Generation

Language has a sequential nature, hence the order in which words appear in the text matters a lot. This feature allows us to understand the context of a sentence even if there are some words missing. Gram generation is the process of combining words so that they make more sense. When two words are joined together it is called 2-gram generation.[8]

The formal definition of N Gram is “a contiguous sequence of n items from a given sample of text”. The main idea is that given any text, we can split it into a list of unigrams (1-gram), bigrams (2-gram), trigrams (3-gram) etc. Consider the below example:

Text: “I went biking”

Unigrams: [(I), (went), (biking)]

Bigrams: [(I, went), (went, biking)]

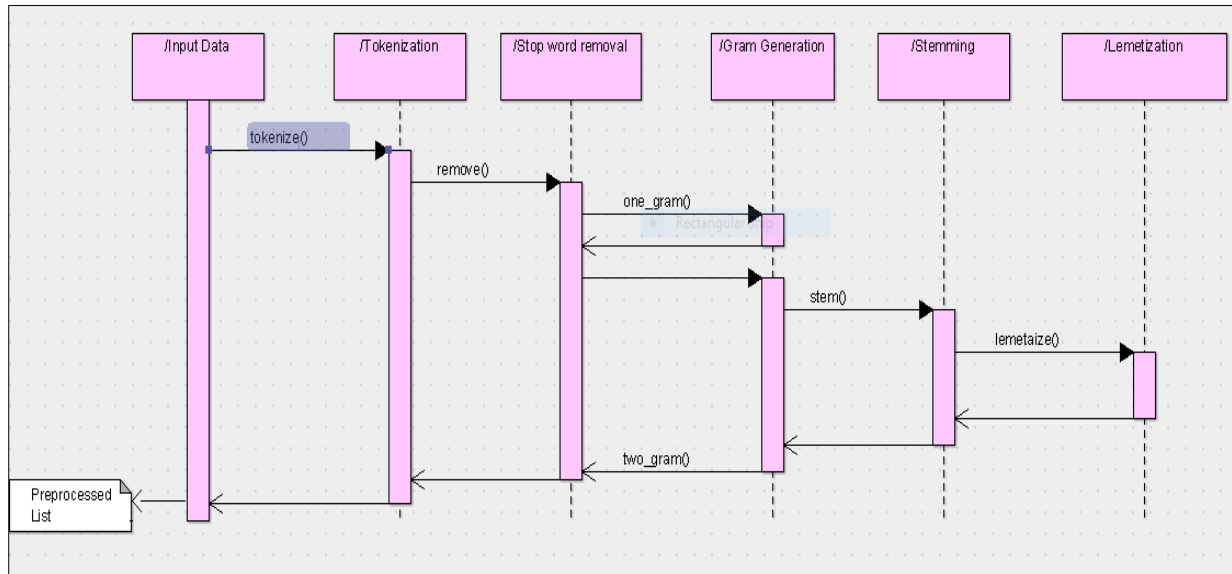
### 4. Stemming

Stemming is considered to be the more crude/brute-force approach to normalization (although this doesn’t necessarily mean that it will perform worse). There are several algorithms, but in general they all use basic rules to chop off the ends of words. Many variations of words carry the same meaning. The reason why stemming is carried out is to shorten the lookup and normalize sentences. For example: ride and riding, are words with the same meaning. Having individual dictionary entries per version would be highly redundant and inefficient. [7]

### 5. Lemmatization

The goal is same as with stemming, but stemming a word sometimes loses the actual meaning of the word. Lemmatization usually refers to doing things properly using vocabulary and morphological analysis of words. It returns the base or dictionary form of a word, also known as the lemma. Stemming can often create non-existent words, whereas lemmas are actual words.

## IV. Architecture Diagram



### Using Collected Data - Tweets

The sentiment model needs to be trained against examples of the type of data that are expected to be seen when the model is being used. Some examples of tweets are as follows:

1. "Feeling kind of low...."
2. "OMG! Just had a fabulous day!"
3. "Eating eggplant. Why bother?"

All the samples should be in the same language (though it doesn't matter what language, as long as it's consistent and space-delimited).

### Labelling the Data

Once the training samples have been collected, each sample is pre-classified with a label. A label is a string that best describes that example, for example: "happy", "sad", "on the fence". So, to assign labels to the previous examples:

1. "sad", "Feeling kind of low...."
2. "excited", "OMG! Just had a fabulous day!"
3. "bored", "Eating eggplant. Why bother?"

A model can have up to 1000 labels, but the best practise is to use as many labels as required for the problem at hand. Each label should have at least a few dozen examples assigned to it. Labels are just strings, so they can have spaces. However, double quotes should be put around any labels that have spaces, and any nested quotation marks should be escaped using a \ mark. Example: "that's fine" Labels are case-sensitive. So "Happy" and "happy" will be seen as two separate labels by the training system. Best practice is to use lowercase for all labels, to avoid mix-ups. Each line can only have one label assigned, but multiple labels can be applied to one example by repeating an example and applying different labels to each one. For example:

1. "excited", "OMG! Just had a fabulous day!"

2. "annoying", "OMG! Just had a fabulous day!"

### Preparing the collected data

The data is formatted as a comma-separated values (CSV) file with one row per example. The format of this file is basically this:

label1, feature1, feature2, feature3, ....

label2, feature1, feature2, feature3, ....

So, the file would look something like this:

"sad", "Feeling kind of low...."

"excited", "OMG! Just had a fabulous day!"

"bored", "Eating eggplant. Why bother?"

### Training the Model with the Naïve Bayes Classifier

The training of the Naive Bayes Classifier is done by iterating through all the documents in the training set. From all the documents, a Hash table with the relative occurrence of each word per class is constructed.

This is done in two steps:

1. Construct a huge list of all occurring words per class:  

```
for ii in range(0,len(Y)):
    label = Y[ii]
    self.nb_dict[label] += X[ii]
```
2. Calculate the relative occurrence of each word in this huge list, with the "calculate\_relative\_occurences" method.

This method uses Python's Counter module to count how much each word occurs and then divides this number with the total number of words. The result is saved in the dictionary nb\_dict.

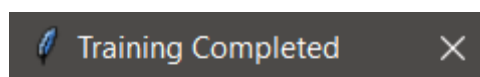
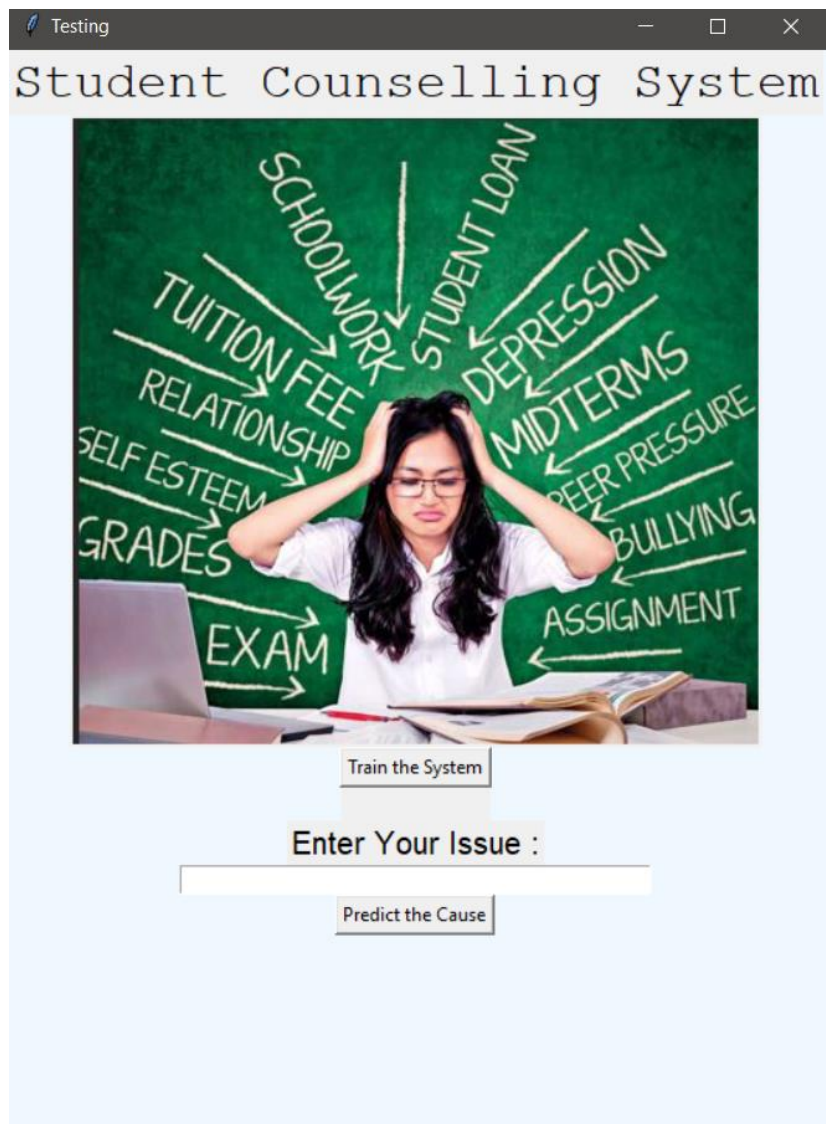
	A	B
1	sentiments	tweets
2	stress and strain	overload
3	stress and strain	overwork
4	stress and strain	stress and strain
5	stress and strain	exam fear
6	stress and strain	overnight study
7	Relationships	relationships are good
8	Relationships	breakup
9	Relationships	disagreement with girlfriend
10	Relationships	disagreement with boyfriend
11	Relationships	affairs
12	Relationships	in love with you
13	stress and strain	huge workload
14	stress and strain	short semesters
15	stress and strain	no time
16	Homesickness	depression
17	Homesickness	feeling lonely
18	Homesickness	want to see parents
19	Homesickness	away from home
20	Debt	debt at company
21	debt	lack of money
22	debt	due to scams
23	debt	cannot pay college fee
24	debt	borrowing money
25	debt	costly textbooks
26	Debt	Business loss
27	debt	debt in family
28	Debt	GDP ratio decline

### Making Predictions and Testing the Prediction Set

Now that the model is successfully built, it is time to make predictions. The output from a prediction call for a classification problem, like sentiment analysis will include several important fields. This dictionary can be updated, saved to file, and loaded back from file. It contains the results of Naive Bayes Classifier training.

Classifying new documents is also done quite easily by calculating the class probability for each class and then selecting the class with the highest probability.

## Testcases




Training Completed

OK

## Case 1:

Testing

# Student Counselling System



SCHOOLWORK  
STUDENT LOAN  
DEPRESSION  
MIDTERMS  
PEER PRESSURE  
BULLYING  
ASSIGNMENT  
EXAM  
GRADES  
SELF ESTEEM  
RELATIONSHIP  
TUITION FEE

Train the System

Enter Your Issue :

I am not feeling well

Predict the Cause

Description of Issue

! Sickness seems to be the cause of your problems. Do you want to get some recommendations to help with this issue?

Yes No

Solution to Sickness


- 1 .Eat healthy and balanced meals.
- 2.Get a good night's sleep.
- 3.Wash your hands often.
- 4.If an illness develop, visit your campus clinic.
- 5.Avoid using drugs and alcohol. This can lead to poor choices, risky behavior, health risks, and even deadly situations.



## Case 2:

Testing

# Student Counselling System




Train the System

Enter Your Issue :

my boyfriend has affairs with other men.

Predict the Cause

Description of Issue

 Relationships and Affairs seem to be the cause of your problems. Do you want to get some recommendations to help with this issue?

Yes No


Solution to Bad Relationships

- 1 . Having sex without sober consent and without taking the necessary precautions can be traumatic, dangerous, and even criminal.
- 2.Establish a clear communication of your needs and expectations from your partner.
- 3.If any problems occur in your relationship, try to talk them out, if not please seek counsellor help.

### Case 3:

Testing

## Student Counselling System




Train the System

Enter Your Issue :

My father comes home after drinking and beat my mother.

Predict the Cause

Description of Issue

 Family Issues seem to be the cause of your problems. Do you want to get some recommendations to help with this issue?

Yes No

Solution to Family Issue

- 1 .Understand that your family loves you irrespective of the circumstances.
- 2.If a family member is alcoholic refer them to a rehab center.
- 3.Seek professional help if needed.
- 4.In case of disputes:
  - 4.1.Try to view events from the perspective of your family members
  - 4.2.Communicate with your family members.

## **Summary and future works**

### **Goals achieved through project**

The goal of this project was to answer the following three questions:

1. To know and understand the comparison between accuracy/results of different data mining techniques
2. To find out why twitter is the most preferred platform compared to other social media platforms like Facebook or Instagram for data mining
3. Why is python useful for analysis such as sentiment analysis?

A Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature and this algorithm is very easy to build up and mainly used for a large set of data. On the other hand, a neural network has two phases, which are training and testing.

Few characteristics of tweets are that they are short messages consisting of maximum 140 characters. Tweets are posted by a wide variety of people, thus the topics discussed also varies and could include almost any topic starting from politics to mobile products.

Python has a wide range of standard libraries and nearly infinite third-party libraries. Some standard Python libraries are Pandas, Numpy, Scikit-Learn, SciPy, and Matplotlib. These libraries are used for data collection, analysis, data mining, visualizations, sentiment analysis and Machine Learning. Libraries such as NLTK is excellent for learning and exploring NLP concepts. SpaCy is a new NLP library that's designed to be fast and production ready.

Here the focus was on students' Twitter posts to understand issues and problems in their educational experiences. This project can help people receive recommendations and suggestions that are evaluated and judged based upon their own opinions and thoughts. As this model is trained with tweets obtained from people, it basically uses input from human beings themselves to train and categorize their problems to help them figure out a solution or provide helpful feedback. This project delivers a user-friendly, fully automated student counselling software. Hopefully this project would help in resolving tensions in existing student life.



## **Future Research**

The project can be made to study the intensity of a problem. This can be done by identifying the adjectives used like very, many, little etc. It is difficult for the computer to address the problems faced by human beings. So, the software can provide initial counsel. But if the problem is identified to be too intense, the person can be directed to a doctor or a counsellor.

A registration and login module can be implemented. If a person is found to be logging in repeatedly with the same problem, he can also be directed to a counsellor. A feedback module can also be included to learn the effectiveness of the system.

Dictionaries like wordnet can be linked to the excel spreadsheet while the input is taken to the classifier. Wordnet maps a word to words which have similar meaning in a certain context. For example, the words 'died' and 'passed away' has same meaning. But if we train the model using the word 'died' and the user uses the term 'passed away' in his input this is not identified by the classifier during the prediction stage. It is not practical to have a training set with all words that has similar meaning. This problem can be solved by using wordnet.

Sentiment analysis bases its results on factors that are so inherently humane, it is bound to become one the major drivers of many business decisions in future. Uncovering and translating social data into tangible insights has become one of the key challenges for Sentiment Analysis technology providers wanting to propose a real social mining solution to their users. A Sentiment Analysis solution needs to provide a rich set of contextual information that helps you understand what is really being said about you, your products, or your brand and to what extent, and through which channels are impacting you and what you can do about it.

Sentiment analysis will improve more in the future, beyond the concept of positive, negative, or neutral, so as to be able to understand the different conversations and learn what these say about the consumers. Hence the most important task for sentiment analysis in the future has more to do with determining where you can correlate sentiment with behaviour rather than improving the accuracy of the algorithms.

Furthermore, brands will be able to easily customize and personalize their services with a thorough understanding and a much larger and more comprehensive database. Instead of segmenting markets based on age, gender, income, and other surface demographics, businesses can segment even further based on how their members feel about the brand.

There is a growing interest in deep sentiment analysis of text in different areas of application. It is not enough to say that a text is overall positive or overall negative. Users would like to know the separate topics discussed and which of them are positive or negative. So, a trend could develop where NLP techniques are largely used (such as syntactic parsing, co-reference resolution, etc), in addition to machine learning methods.

Natural Language Processing (NLP) – with slang, context, and the ability to deal, detect and understand sarcasm are few of the biggest stumbling blocks. Social listening is much more than just a sentiment score.

We must one day be able to harness the wisdom of crowds – scale and let the algorithm learn and not just stop at positive, negative, or neutral sentiment by taking into consideration wishes, caveats, comparisons, and preferences.

Hopefully the future of sentiment analysis will plug the existing gaps in being able to interpret and understand human emotions of happiness, sadness, fear, anger, surprise, disgust and increase the accuracy when compared to human processing.

## References

1. Deshapande, P., Joshi, P., Pawar, P., Madekar, D., & Salunke, P. (2019). *A Survey On: Sentiment Analysis framework of Twitter data Using Classification*. Retrieved 30 March 2022.
2. Siddharth, S., Darsini, R., & Sujithra, D. (2018). *Sentiment Analysis on Twitter data using Machine Learning Algorithms in Python*. Retrieved 30 March 2022.
3. Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). *Study Of Twitter Sentiment Analysis Using Machine Learning Algorithms On Python*. Retrieved 30 March 2022.
4. Khattak, A., Batool, R., Satti, F., Hussain, J., Khan, W., Khan, A., & Hayat, B. (2020). *Tweets Classification And Sentiment Analysis For Personalized Tweets Recommendation*. Retrieved 30 March 2022.
5. Faizan, F. (2019). *Twitter Sentiment Analysis*. Retrieved 30 March 2022.
6. Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). *Sentiment Analysis Of Twitter Data*. Retrieved 30 March 2022.
7. Alsaeedi, A., & Khan, M. (2019). *A Study On Sentiment Analysis Techniques Of Twitter Data*. Retrieved 30 March 2022.
8. Gamal, D., Alfonse, M., El-Horbarty, E., & Salem, A. (2019). *Implementation Of Machine Learning Algorithms In Arabic Sentiment Analysis Using N-Gram Feature*. Retrieved 30 March 2022.
9. X. Chen, M. Vorvoreanu and K. Madhavan, "Mining Social Media Data for Understanding Students' Learning Experiences," in IEEE Transactions on Learning Technologies, vol. 7, no. 3, pp. 246-259, July-Sept. 2014, doi: 10.1109/TLT.2013.2296520.
10. Walaa Medhat, Ahmed Hassan, Hoda Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal, Volume 5, Issue 4, 2014, Pages 1093-1113, ISSN 2090-4479, <https://doi.org/10.1016/j.asej.2014.04.011>.
11. C. Kaur and A. Sharma, "Social Issues Sentiment Analysis using Python," 2020 5th International Conference on Computing, Communication and Security (ICCCS), 2020, pp. 1-6, doi: 10.1109/ICCCS49678.2020.9277251.
12. Kharde, Vishal & Sonawane, Sheetal. (2016). Sentiment Analysis of Twitter Data: A Survey of Techniques. International Journal of Computer Applications. 139. 5-15. 10.5120/ijca2016908625.

## Appendices

### twitterstream.py

```
import oauth2 as oauth
import urllib.request as urllib

api_key = "tM0jtou34cPgrgk4U6InKrKuK"
api_secret = "JR7dH33oc0OpU4jYW6L8h0EtPOoZJyPxwa5i1h2mN2k6uyjqSU"
access_token_key = "1521026820266725378-WhYmWvQBXYDJvy5ERW0w0wzfwZPrI"
access_token_secret = "XGABVmuQ89LWZxu1eCWOo6mjFyX955JnmTXp3SVL0ozSR"

_debug = 0

oauth_token = oauth.Token(key=access_token_key, secret=access_token_secret)
oauth_consumer = oauth.Consumer(key=api_key, secret=api_secret)

signature_method_hmac_sha1 = oauth.SignatureMethod_HMAC_SHA1()

http_method = "GET"

http_handler = urllib.HTTPHandler(debuglevel=_debug)
https_handler = urllib.HTTPSHandler(debuglevel=_debug)

'''
Construct, sign, and open a twitter request
using the hard-coded credentials above.
'''

def twitterreq(url, method, parameters):
    req = oauth.Request.from_consumer_and_token(oauth_consumer,
                                                token=oauth_token,
                                                http_method=http_method,
                                                http_url=url,
                                                parameters=parameters)

    req.sign_request(signature_method_hmac_sha1, oauth_consumer, oauth_token)

    #headers = req.to_header()

    if http_method == "POST":
        encoded_post_data = req.to_postdata()
    else:
        encoded_post_data = None
        url = req.to_url()

    opener = urllib.OpenerDirector()
    opener.add_handler(http_handler)
    opener.add_handler(https_handler)

    response = opener.open(url, encoded_post_data)
```

```

return response

def fetchsamples():
    url = "https://api.twitter.com/1.1/search/tweets.json?q=studentlife"
    parameters = []
    response = twitterreq(url, "GET", parameters)
    f = open('tweetfile.txt', 'w', encoding="utf-8")
    for line in response:
        print (line.strip())
        f.write(str(line.strip()))

    f.close()

if __name__ == '__main__':
    fetchsamples()

```

### **main.py**

```

from tkinter import *
from tkinter import filedialog
from tkinter.filedialog import askopenfilename
import nltk
from collections import *
import re
from nltk import ngrams
from tkinter import Button
from math import *

class Abc:
    def __init__(self,root):
        self.content=[]
        b1=Button(root,text="Choose File",command=lambda: self.chose())
        b1.pack()
        b2=Button(root,text="Display",command=lambda: self.disp())
        b2.pack()
        global fnm
    def show(self):
        pass
    def chose(self):
        global fnm
        filename=askopenfilename()
        fnm=filename

        print (filename)
        with open(filename, 'r') as content_file:
            self.content.append(content_file.read())

    def disp(self):
        print (self.content)
        w1=Toplevel()

```

```

t1=Text(w1)
t1.pack()
t1.insert(INSERT,self.content)
b3=Button(w1,text="Tokenize",command=lambda: self.tok())
b3.pack()

def processTweet(self):
    print (self.content)
    tweet = self.content
    l=[]
    l1=[]
    ndefinition=[]
    #Convert to lower case
    tweet = tweet.lower()
    #Convert www.* or https?://* to URL
    tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))',",",tweet)
    #Convert @username to AT_USER
    tweet = re.sub('@[^\s]+',"",tweet)
    #Remove additional white spaces
    tweet = re.sub('[\s]+', ' ', tweet)
    #Replace #word with word
    tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
    #trim
    tweet = tweet.strip("\'")
    for i in tweet.split():
        l.append(i)

    for i in l:
        if (i not in STOP_WORDS):
            l1.append(i)
    tweet=" ".join(l1)

    self.content = tweet
    print (self.content)

def tok(self):
    print (self.content)
    tok1=[nltk.word_tokenize(i) for i in self.content]
    print (tok1)
    w2=Toplevel()
    t2=Text(w2)
    t2.pack()
    t2.insert(INSERT,tok1)
    b4=Button(w2,text="Uni gram",command=lambda: self.unigrm())
    b4.pack()

def unigrm(self):
    print (self.content)
    w4=Toplevel()
    t4=Text(w4)

```

```

t4.pack()
for i in self.content:
    onegrams = ngrams(i.split(), 1)
    for grams in onegrams:
        print (grams)
        t4.insert(INSERT,grams)
        t4.insert(INSERT,"\n")
b6=Button(w4,text="Bi gram",command=lambda: self.bigrm())
b6.pack()
def bigrm(self):
    print (self.content)
    w4=Toplevel()
    t4=Text(w4)
    t4.pack()
    for i in self.content:
        twograms = ngrams(i.split(), 2)
        for grams in twograms:
            print (grams)
            t4.insert(INSERT,grams)
            t4.insert(INSERT,"\n")

if __name__=="__main__":
    w=Tk()
    cd=Abc(w)
    w.geometry("%dx%d+%d+%d" % (300 ,200,80, 50))
    w.mainloop()

```

## Testing.py

```

import nltk
from tkinter import *
from tkinter import messagebox
import pandas as pd
import numpy as np
import time
import re

# get the word lists of tweets
def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
        all_words.extend(words)

    return all_words

def close(wndw):
    wndw.destroy()

def result(result):
    rs=result

```

```

print(rs[0])
print("-----")

if rs=="familyissue":
    result = messagebox.askquestion("Description of Issue","Family Issues seem to be the
cause of your problems. Do you want to get some recommendations to help with this issue?",
icon='warning')
    if result == 'yes':
        print("Recommendation for familyissue")
        soln = """1 .Understand that your family loves you irrespective of the
circumstances.
2.If a family member is alcoholic refer them to a rehab center.
3.Seek professional help if needed.
4.In case of disputes:
    4.1.Try to view events from the perspective of your family members
    4.2.Communicate with your family members.
"""

        wn=Tk()
        w=350;h=300
        x=(wn.winfo_screenwidth()- w)/2
        y=(wn.winfo_screenheight()-h)/2
        wn.geometry("%dx%d+%d+%d" % (w,h,x,y))
        wn.configure(bg="AliceBlue")
        wn.title("Solution to Family Issue")
        text = Text(wn)
        text.insert(INSERT, soln)
        text.pack(side=LEFT , fill=Y)
        wn.mainloop()

    else:
        print("Finished")

elif rs=="stress and strain":
    result = messagebox.askquestion("Description of Issue","Stress and Strain seems to
be the cause of your problems. Do you want to get some recommendations to help with this
issue?", icon='warning')
    if result == 'yes':
        print("Recommendation for stress and strain")
        soln = """1 .Know your limits. If you can't handle that much amount of work, it
will help to take a break and slow down.
2.It is important to schedule time.
4.Take breaks to keep your mind fresh and clear.
5.Manage academic stress using effective study habits.
6.Prioritize and schedule events, games, meetings, social events, and studies accordingly.
"""

        wn=Tk()
        w=350;h=300
        x=(wn.winfo_screenwidth()- w)/2

```

```

y=(wn.winfo_screenheight()-h)/2
wn.geometry("%dx%d+%d+%d" % (w,h,x,y))
wn.configure(bg="AliceBlue")
wn.title("Solution to Stress and Strain")
text = Text(wn)
text.insert(INSERT, soln)
text.pack(side=LEFT , fill=Y)
wn.mainloop()

```

```

else:
    print("Finished")

```

```

elif rs=="Relationships":
    result = messagebox.askquestion("Description of Issue","Relationships and Affairs
seem to be the cause of your problems. Do you want to get some recommendations to help
with this issue?", icon='warning')
    if result == 'yes':
        print("Recommendation for Relationships")
        soln = """1 . Having sex without sober consent and without taking the necessary
precautions can be traumatic, dangerous, and even criminal.
2.Establish a clear communication of your needs and expectations from your partner.
3.If any problems occur in your relationship, try to talk them out, if not please seek
counsellor help.
"""

```

```

wn=Tk()
w=350;h=300
x=(wn.winfo_screenwidth()- w)/2
y=(wn.winfo_screenheight()-h)/2
wn.geometry("%dx%d+%d+%d" % (w,h,x,y))
wn.configure(bg="AliceBlue")
wn.title("Solution to Bad Relationships")
text = Text(wn)
text.insert(INSERT, soln)
text.pack(side=LEFT , fill=Y)
wn.mainloop()

```

```

else:
    print("Finished")

```

```

elif rs=="sickness":
    result = messagebox.askquestion("Description of Issue","Sickness seems to be the
cause of your problems. Do you want to get some recommendations to help with this issue?",
icon='warning')
    if result == 'yes':
        print("Recommendation for Sickness")
        soln = """1 .Eat healthy and balanced meals.
2.Get a good night's sleep.
3.Wash your hands often.
4.If an illness develop, visit your campus clinic.

```



5. Avoid using drugs and alcohol. This can lead to poor choices, risky behavior, health risks, and even deadly situations.

```
"""
```

```
    wn=Tk()
    w=350;h=300
    x=(wn.winfo_screenwidth()- w)/2
    y=(wn.winfo_screenheight()-h)/2
    wn.geometry("%dx%d+%d+%d" % (w,h,x,y))
    wn.configure(bg="AliceBlue")
    wn.title("Solution to Sickness")
    text = Text(wn)
    text.insert(INSERT, soln)
    text.pack(side=LEFT , fill=Y)
    wn.mainloop()
```

```
    else:
        print("Finished")
```

```
# get the unique word from the word list
def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features
```

```
def testit(tweet_test):
    if tweet_test == "":
        messagebox.showinfo("Null value!!!", "Null value not allowed!!!")
```

```
word_features = get_word_features(get_words_in_tweets(tweets))
def extract_features(document):
    document_words = set(document)
    features = { }
    for word in word_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
print (tweet_test)
```

```
#Convert to lower case
tweet = tweet_test.lower()
#Convert www.* or https?://* to URL
tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', "", tweet)
#Convert @username to AT_USER
tweet = re.sub('@[^\s]+', "", tweet)
#Remove additional white spaces
tweet = re.sub('[\s]+', ' ', tweet)
#Replace #word with word
tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
#trim
tweet = tweet.strip("\n")
```

```

print(classifier.classify(extract_features(tweet.split())))
result(classifier.classify(extract_features(tweet.split())))

```

```

def mainn():
    train = pd.read_csv("output.csv", header=0, delimiter=",", quoting=1)
    num_reviews = train["tweets"].size
    print(num_reviews)
    data=[]
    sentiments=[]
    global tweets
    tweets = []

    for i in range( 0,num_reviews ):
        #Convert www.* or https?://* to URL
        tweet = re.sub('((www\.[^\s]+)|(https?://[^\s]+))', "", train["tweets"][i])
        #Convert @username to AT_USER
        tweet = re.sub('@[^\s]+', "", tweet)
        #Remove additional white spaces
        tweet = re.sub('[\s]+', ' ', tweet)
        #Replace #word with word
        tweet = re.sub(r'#([^\s]+)', r'\1', tweet)
        #trim
        tweet = tweet.strip("\'")
        words_filtereds = [e.lower() for e in tweet.split() if len(e) >= 3]
        tweets.append((words_filtereds, train["sentiments"][i]))

    word_features = get_word_features(get_words_in_tweets(tweets))

    def extract_features(document):
        document_words = set(document)
        features = { }
        for word in word_features:
            features['contains(%s)' % word] = (word in document_words)
        return features

    training_set = nltk.classify.util.apply_features(extract_features, tweets)
    time.sleep(5)

    global classifier
    classifier = nltk.NaiveBayesClassifier.train(training_set)

    def train(labeled_featuresets, estimator=nltk.probability.ELEProbDist):
        # Create the P(label) distribution
        label_probdist = estimator(label_freqdist)
        # Create the P(fval|label, fname) distribution
        feature_probdist = { }
        return NaiveBayesClassifier(label_probdist, feature_probdist)
    messagebox.showinfo("Training Completed", "Training Completed")

```

## TestwithGUI.py

```
from tkinter import *
from tkinter import messagebox
import pandas as pd
import re
from PIL import ImageTk, Image
from sklearn.ensemble import ExtraTreesClassifier
#from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import VotingClassifier
from sklearn import decomposition
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier
from sklearn.pipeline import Pipeline
from sklearn.svm import LinearSVC
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import tkinter.messagebox as mb
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
import warnings
warnings.filterwarnings("ignore")
import main
from Testing import *
def test():
    from PIL import ImageTk, Image
    wndw=Tk()
    w=760;h=700
    x=(wndw.winfo_screenwidth()- w)/2
    y=(wndw.winfo_screenheight()-h)/2
    wndw.geometry("%dx%d+%d+%d" % (w,h,x,y))
    wndw.configure(bg="AliceBlue")
    wndw.title("Testing")
    l1 = Label(wndw, text="Student Counselling System")
    l1.config(font=("Courier", 25))
    l1.grid(row=1,column=1)
    img_ar = Image.open("student.jpg")
```

```

ph_ar = ImageTk.PhotoImage(img_ar)
lbl_ar = Label(wndw,image=ph_ar)
lbl_ar.image = ph_ar
lbl_ar.grid(row=2,columnspan=3)
btn_test=Button(wndw,text="Train the System",command=lambda:mainn())
btn_test.grid(row=4,column=1)
l3 = Label(wndw, text="")
l3.grid(row=5,column=1)
l2 = Label(wndw, text="Enter Your Issue :")
l2.config(font=("Normal", 15))
l2.grid(row=6,column=1)

global entry_box
entry_box=Entry(wndw)
entry_box.config(width=50)
entry_box.grid(row=7,column=1)
btn_test=Button(wndw,text="Predict the
Cause",command=lambda:testit(str(entry_box.get()))))
btn_test.grid(row=8,column=1)
wndw.resizable(2,2)
wndw.mainloop()
def result(result):
    rs=result
    messagebox.showinfo( "Test Result", rs)
def act2():
    from sklearn import svm
    from sklearn.preprocessing import label_binarize
    sentii="Sorry!! Not Found"
    path=str(entry_box.get())
    if (path==""):
        result("Null Value!!!")
        exit(0)
    train = pd.read_csv("output.csv", header=0,delimiter=",", quoting=1)
    num_reviews = train["tweets"].size
    print(num_reviews)
    data=[]
    sentiments=[]
    for i in range( 0,num_reviews ):
        data.append(train["tweets"][i])
        sentiments.append(train["sentiments"][i])
    for i in range(len(data)):
        if(data[i]==path):
            print("Here...")
            sentii=sentiments[i]

## print sentiments
print("Sentiments and tweets copied")
from sklearn import svm
from sklearn.preprocessing import label_binarize
print(sentii)

```

```

if(sentii=="Sorry!! Not Found"):
    print("Not found")
    data1=[]
    trainingnumber=train["tweets"].size
    data1.append(path)
    from sklearn import svm
    from sklearn.preprocessing import label_binarize

Y=label_binarize(sentiments,classes=["stressandstrain","Relationships","timeissue","Debt","s
ickness","familyissue"])
arr=np.array(data)
arr1=np.array(data1)

print("Classifier creation")
clf1 = RandomForestClassifier(random_state=1)
clf2 =DecisionTreeClassifier(random_state=1)

eclf = VotingClassifier(estimators=[('rand', clf1), ('dt', clf2)], voting='soft')
crfcls = Pipeline([
    ('vectorizer', CountVectorizer( ngram_range=(2, 3))),
    ('tfidf', TfidfTransformer()),
    ('selectbest',SelectKBest(chi2, k=1500)),
    ('clf',eclf) ] )

ExtraTree = Pipeline([
    ('vectorizer', CountVectorizer( ngram_range=(2, 3))),
    ('tfidf', TfidfTransformer()),
    ('selectbest',SelectKBest(chi2, k=10)),
    ('clf',ExtraTreesClassifier(n_estimators=100))] )

print("completed")
print("Training the Extra tree ...")
ExtraTrees = ExtraTree.fit( arr, sentiments)
print("completed")
resultOfExtra=ExtraTrees.predict(arr1)
print(resultOfExtra)
result(resultOfExtra)
else:
    result(sentii)
print("Starting process")
print("Reading lines...")
test()

```