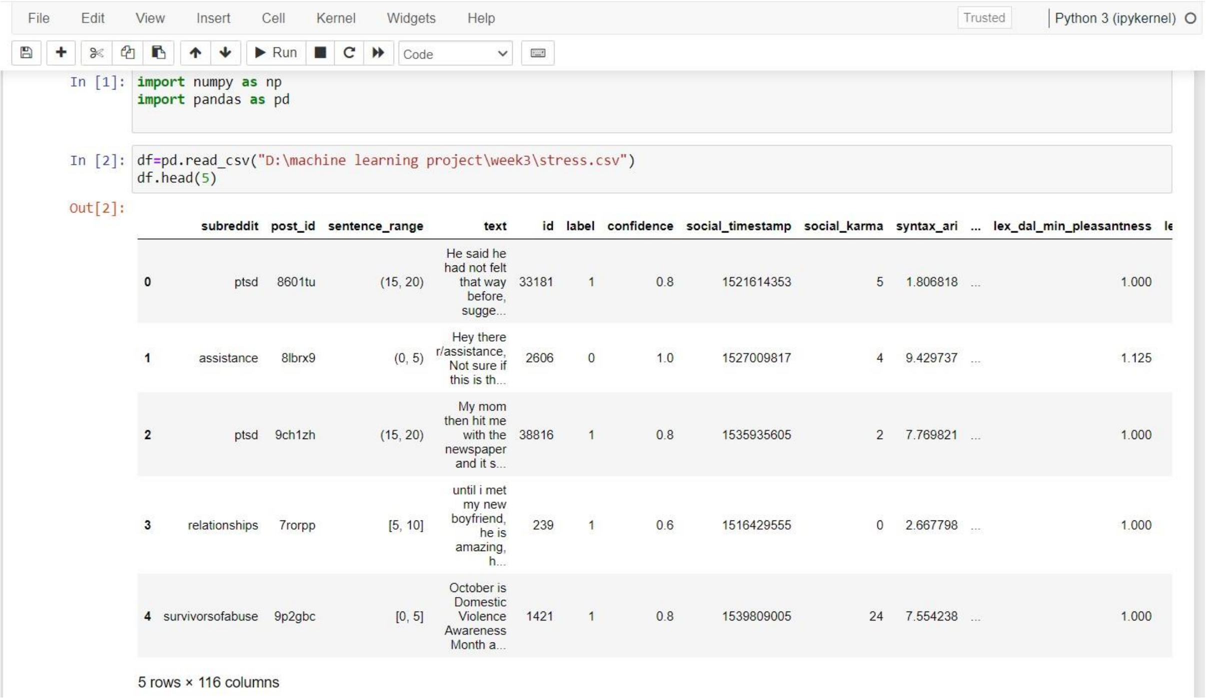
NAME: SAI KIRAN PANDIRI
Email: kiransai2128@gmail.com

Step 1: Open Jupyter Notebook.

Step2: Import numpy and pandas.

Step3: Import Dataset.



```
In [1]: import numpy as np
        import pandas as pd
```

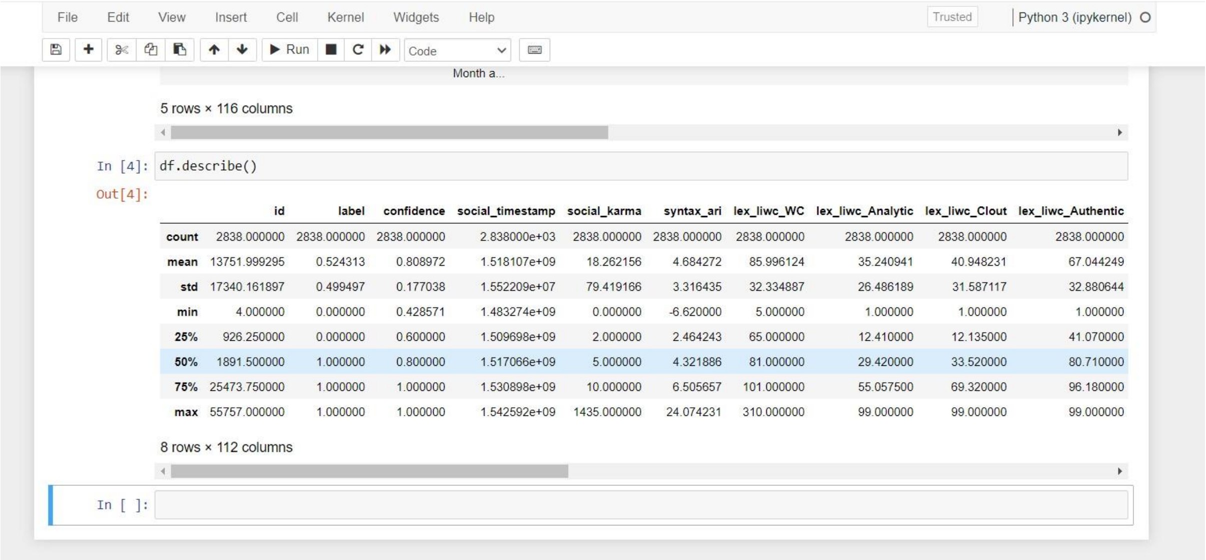
```
In [2]: df=pd.read_csv("D:\machine learning project\week3\stress.csv")
        df.head(5)
```

Out[2]:

	subreddit	post_id	sentence_range	text	id	label	confidence	social_timestamp	social_karma	syntax_ari	...	lex_dal_min_pleasantness	le
0	ptsd	8601tu	(15, 20)	He said he had not felt that way before, sugge...	33181	1	0.8	1521614353	5	1.806818	...	1.000	
1	assistance	8lbrx9	(0, 5)	Hey there r/assistance, Not sure if this is th...	2606	0	1.0	1527009817	4	9.429737	...	1.125	
2	ptsd	9ch1zh	(15, 20)	My mom then hit me with the newspaper and it s...	38816	1	0.8	1535935605	2	7.769821	...	1.000	
3	relationships	7rorpp	[5, 10]	until i met my new boyfriend, he is amazing, h...	239	1	0.6	1516429555	0	2.667798	...	1.000	
4	survivorsofabuse	9p2gbc	[0, 5]	October is Domestic Violence Awareness Month a...	1421	1	0.8	1539809005	24	7.554238	...	1.000	

5 rows × 116 columns

Step 4: Check Description of our data



Month a...

5 rows × 116 columns

```
In [4]: df.describe()
```

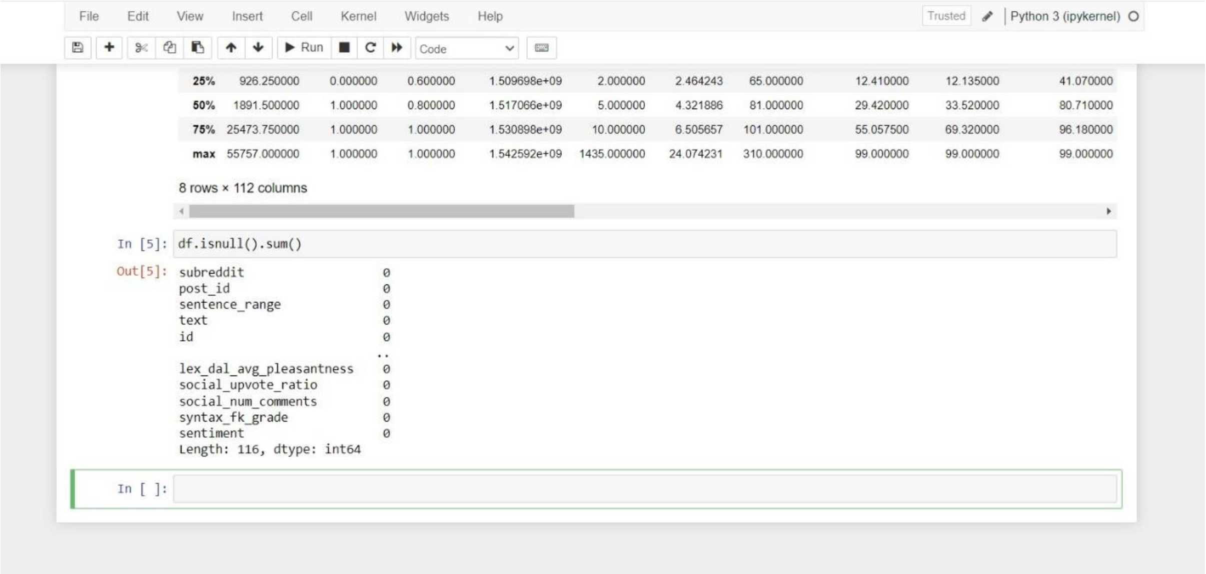
Out[4]:

	id	label	confidence	social_timestamp	social_karma	syntax_ari	lex_liwc_WC	lex_liwc_Analytic	lex_liwc_Clout	lex_liwc_Authentic
count	2838.000000	2838.000000	2838.000000	2.838000e+03	2838.000000	2838.000000	2838.000000	2838.000000	2838.000000	2838.000000
mean	13751.999295	0.524313	0.808972	1.518107e+09	18.262156	4.684272	85.996124	35.240941	40.948231	67.044249
std	17340.161897	0.499497	0.177038	1.552209e+07	79.419166	3.316435	32.334887	26.486189	31.587117	32.880644
min	4.000000	0.000000	0.428571	1.483274e+09	0.000000	-6.620000	5.000000	1.000000	1.000000	1.000000
25%	926.250000	0.000000	0.600000	1.509698e+09	2.000000	2.464243	65.000000	12.410000	12.135000	41.070000
50%	1891.500000	1.000000	0.800000	1.517066e+09	5.000000	4.321886	81.000000	29.420000	33.520000	80.710000
75%	25473.750000	1.000000	1.000000	1.530898e+09	10.000000	6.505657	101.000000	55.057500	69.320000	96.180000
max	55757.000000	1.000000	1.000000	1.542592e+09	1435.000000	24.074231	310.000000	99.000000	99.000000	99.000000

8 rows × 112 columns

```
In [ ]:
```

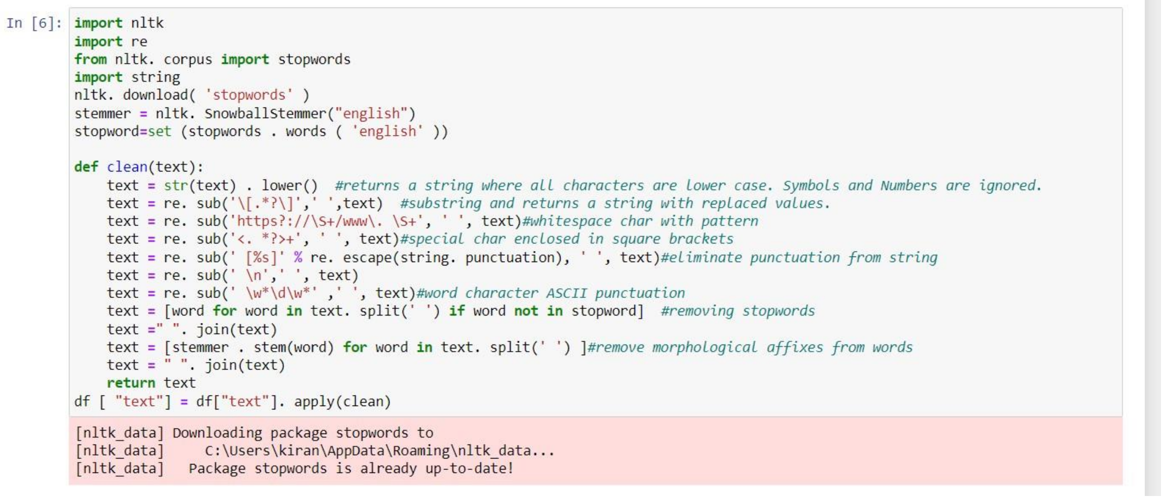
Step 5: Check if dataset contains null value or not.



```
 25%    926.250000  0.000000  0.600000  1.509698e+09     2.000000   2.464243   65.000000  12.410000  12.135000  41.070000
 50%   1891.500000  1.000000  0.800000  1.517066e+09     5.000000   4.321886   81.000000  29.420000  33.520000  80.710000
 75%  25473.750000  1.000000  1.000000  1.530898e+09    10.000000   6.505657  101.000000  55.057500  69.320000  96.180000
 max  55757.000000  1.000000  1.000000  1.542592e+09  1435.000000  24.074231  310.000000  99.000000  99.000000  99.000000

8 rows × 112 columns
```

```
In [5]: df.isnull().sum()

Out[5]: subreddit                  0
        post_id                    0
        sentence_range             0
        text                       0
        id                         0
                                  ..
        lex_dal_avg_pleasantness   0
        social_upvote_ratio        0
        social_num_comments        0
        syntax_fk_grade            0
        sentiment                  0
        Length: 116, dtype: int64

In [ ]:
```

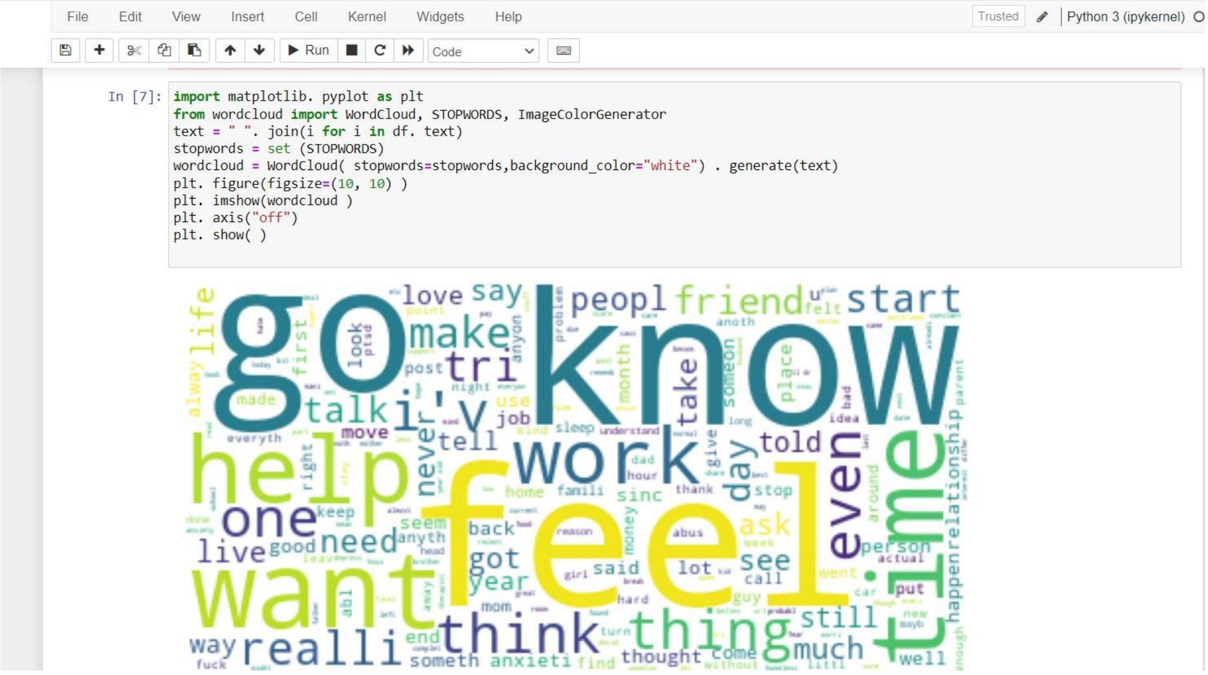
6.Prepare the text column of this dataset to clean the text column with stop words, links, special symbols and language errors.



```
In [6]: import nltk
        import re
        from nltk. corpus import stopwords
        import string
        nltk. download( 'stopwords' )
        stemmer = nltk. SnowballStemmer("english")
        stopword=set (stopwords . words ( 'english' ))

        def clean(text):
            text = str(text) . lower()  #returns a string where all characters are lower case. Symbols and Numbers are ignored.
            text = re. sub('\[.*?\]',' ',text)  #substring and returns a string with replaced values.
            text = re. sub('https?://\S+/www\. \S+', ' ', text)#whitespace char with pattern
            text = re. sub('<. *?>+', ' ', text)#special char enclosed in square brackets
            text = re. sub(' [%s]' % re. escape(string. punctuation), ' ', text)#eliminate punctuation from string
            text = re. sub(' \n',' ', text)
            text = re. sub(' \w*\d\w*' ,' ', text)#word character ASCII punctuation
            text = [word for word in text. split(' ') if word not in stopword]  #removing stopwords
            text =" ". join(text)
            text = [stemmer . stem(word) for word in text. split(' ') ]#remove morphological affixes from words
            text = " ". join(text)
            return text
        df [ "text"] = df["text"]. apply(clean)

        [nltk_data] Downloading package stopwords to
        [nltk_data]     C:\Users\kiran\AppData\Roaming\nltk_data...
        [nltk_data]   Package stopwords is already up-to-date!
```

NAME: SAI KIRAN PANDIRI
Email: kiransai2128@gmail.com

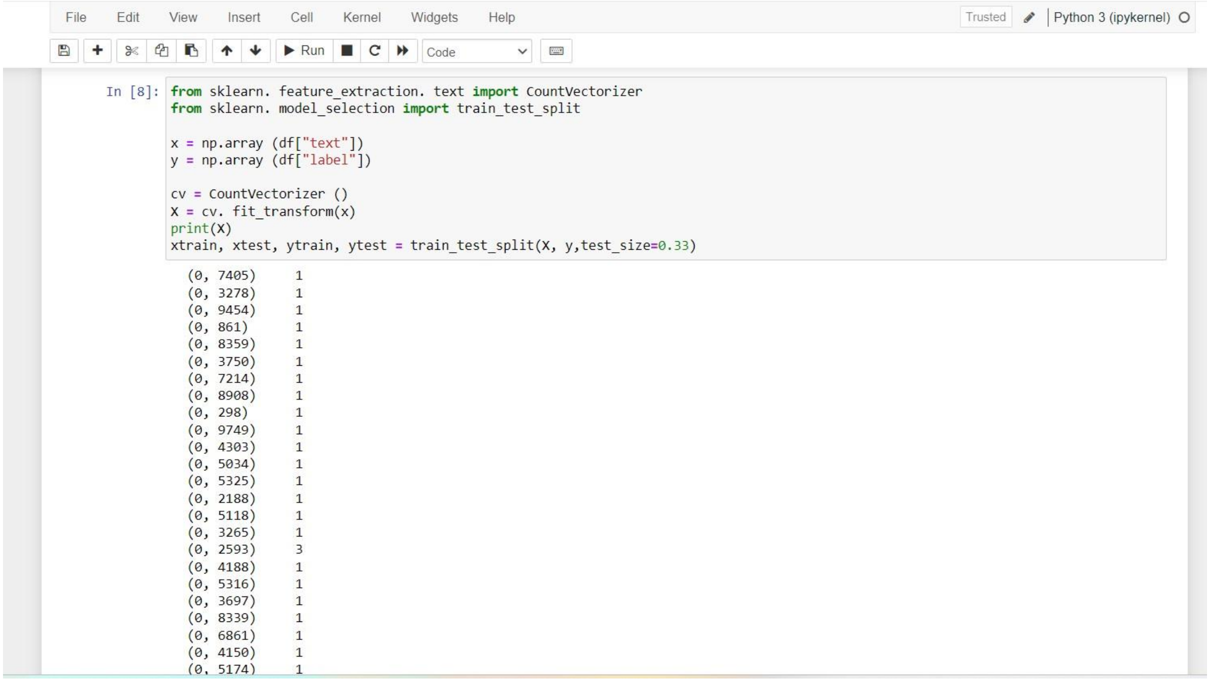
Step 7: View the most utilized words by individuals sharing about their life issues via online entertainment by picturing a word cloud of the text column.



```
In [7]: import matplotlib. pyplot as plt
        from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
        text = " ". join(i for i in df. text)
        stopwords = set (STOPWORDS)
        wordcloud = WordCloud( stopwords=stopwords,background_color="white") . generate(text)
        plt. figure(figsize=(10, 10) )
        plt. imshow(wordcloud )
        plt. axis("off")
        plt. show( )
```

Step 8: The label column in this dataset contains labels as 0 and 1. 0 means no stress, and 1 means stress. We will use Stress and No stress labels instead of 1 and 0. So let’s prepare this column accordingly and select the text and label columns for the process of training a machine learning model.

Step 9: Split the dataset into training and test sets.

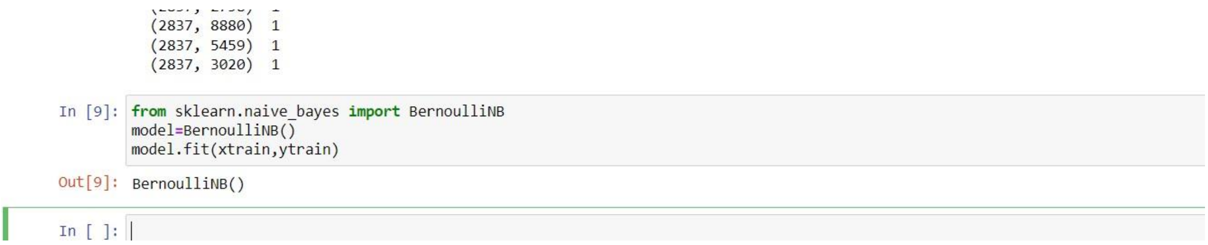


```
In [8]: from sklearn. feature_extraction. text import CountVectorizer
        from sklearn. model_selection import train_test_split

        x = np.array (df["text"])
        y = np.array (df["label"])

        cv = CountVectorizer ()
        X = cv. fit_transform(x)
        print(X)
        xtrain, xtest, ytrain, ytest = train_test_split(X, y,test_size=0.33)
```

```
  (0, 7405)	1
  (0, 3278)	1
  (0, 9454)	1
  (0, 861)	1
  (0, 8359)	1
  (0, 3750)	1
  (0, 7214)	1
  (0, 8908)	1
  (0, 298)	1
  (0, 9749)	1
  (0, 4303)	1
  (0, 5034)	1
  (0, 5325)	1
  (0, 2188)	1
  (0, 5118)	1
  (0, 3265)	1
  (0, 2593)	3
  (0, 4188)	1
  (0, 5316)	1
  (0, 3697)	1
  (0, 8339)	1
  (0, 6861)	1
  (0, 4150)	1
  (0, 5174)	1
```

NAME: SAI KIRAN PANDIRI
Email: kiransai2128@gmail.com

Step 10: This task is based on the problem of binary classification, we will be using the Bernoulli Naïve Bayes algorithm, which is one of the best algorithms for binary classification problems.

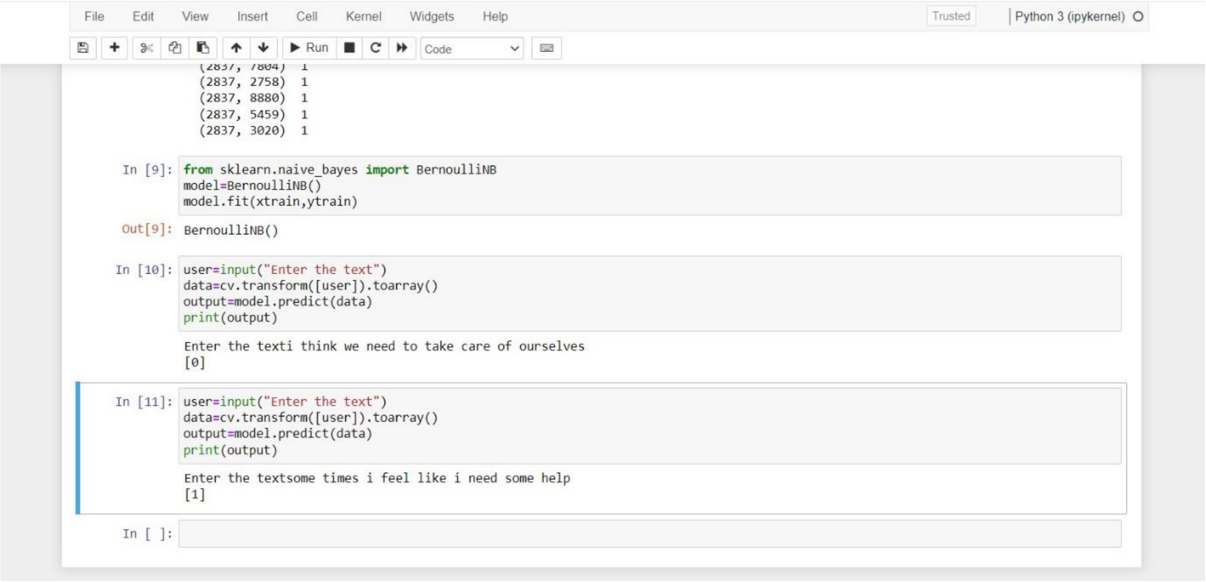


```
  (2837, 8880)	1
  (2837, 5459)	1
  (2837, 3020)	1

In [9]: from sklearn.naive_bayes import BernoulliNB
        model=BernoulliNB()
        model.fit(xtrain,ytrain)

Out[9]: BernoulliNB()

In [ ]: 
```

Step 11: Test the performance of our model on some random sentences based on mental health.

Ex: “I think we need to take care of ourselves.”

“Sometimes I feel like I need some help”



```
  (2837, 2758)	1
  (2837, 8880)	1
  (2837, 5459)	1
  (2837, 3020)	1

In [9]: from sklearn.naive_bayes import BernoulliNB
        model=BernoulliNB()
        model.fit(xtrain,ytrain)

Out[9]: BernoulliNB()

In [10]: user=input("Enter the text")
         data=cv.transform([user]).toarray()
         output=model.predict(data)
         print(output)

         Enter the texti think we need to take care of ourselves
         [0]

In [11]: user=input("Enter the text")
         data=cv.transform([user]).toarray()
         output=model.predict(data)
         print(output)

         Enter the textsome times i feel like i need some help
         [1]

In [ ]: 
```