Teammates:

Sri Nikhitha Boddapati – sb4dz@umsystem.edu - 16322565

GitHub link: https://github.com/UMKC-APL-BigDataAnalytics/icp-2-Srinikhitha98

Sukumar Bodapati – sb5zh@umsystem.edu - 16326105

GitHub link: https://github.com/UMKC-APL-BigDataAnalytics/icp-2-sukumarbodapati

Video link: https://youtu.be/9VmQZ_epmRs

ICP - Big data App & Analytics – NLP

Outcomes of ICP:

What is NLP?

It is an interaction between humans and machines.

Sentimental analysis:

It is the use of NLP, analysis of text, computational linguistics to identify, quantify and study the information systematically.

Tokenization:

This process will turn the meaningful pieces of data into some random string that has no meaning which are known as token.

Stemming:

It is processing of reducing a word into a base form called lamma.

Vectorization:

This process will make us to find distinct feature and converts the text into numerical data.In this ICP, we have used count vectorization and TFIDF.

Training and testing:

These two are important in the Machine learning. As the name suggests, training will help us out to train our algorithms and testing helps us to validate the trained data and adjust accordingly.

Objective:

To perform sentimental analysis for the given data

Tasks:

1.Install the required libraries.

2.Read the input csv file

3. Read all the tweet columns into a string.

4.Now we will tokenize the entire text into sentences

5.Similarly we will tokenize the text into words

```
[5]  #Tokenizing the text into words
     words = word_tokenize(text)

     #Printing the length and top 100 words

     print (len(words))
     print("\n")
     print (words[:100])
```

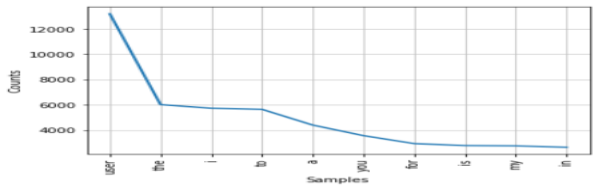
```
'for', '#', 'lyft', 'credit', 'i', 'ca', "n't", 'us',
```

6.For analysis further, we are counting the frequency of each word and printing the top 10 frequency elements.

7.Plot it in the form of linear graph

8.Remove the punctuation using the user defined function and convert them into lower.

```
[10] #Plot the common words on grpah:
     fdist.plot(10)
```




```
[11] from nltk.corpus import stopwords

     #Get the list of stopwords in English dictonary
     stopwords = stopwords.words("english")
     print(stopwords)
     print(len(stopwords))
```

9. After importing stopwords module from the library mapped all the stopwords into a list.

10. From the analysis, since user word is recurring, we have removed it by extending the stopwords module.

11. Cleaned all the data by removing the stopwords from the list of words.

12.Arranging all the words into a wordcloud with a lion logo in the background.

```
[17] plt.figure(figsize = (15, 10))
     plt.imshow(wordcloud)
     plt.axis("off")
     plt.show()
```

13. Then we processed the words through lemmatization and tried stemming the words.

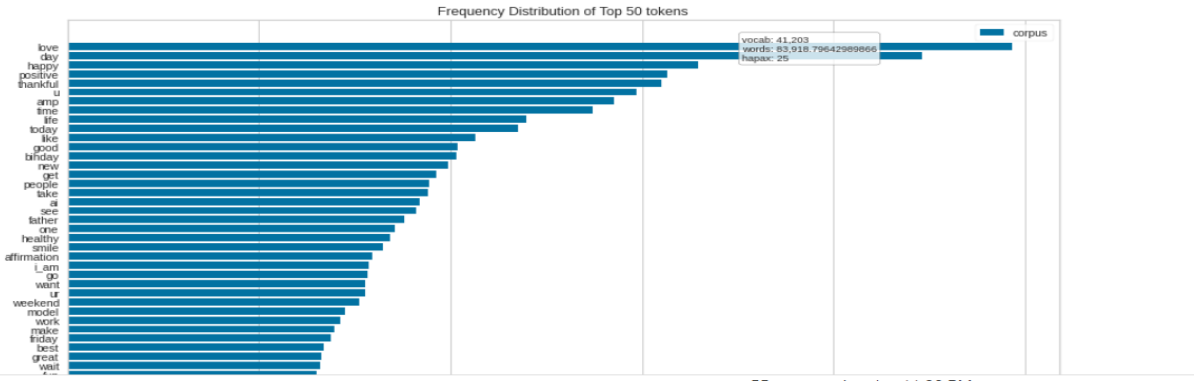
14. Applied part of speech tagging to the lemmatized words.

15. To get the distinct values we used vectorization which converts the words into distinct numerical data.

16.After vectorization plotting the data into bar graph.

```
# Displaying the content in a bar chart
hist_visualizer = FreqDistVisualizer(features=key_features_tdidf, orient='h')
plt.figure(figsize = (15, 10))
hist_visualizer.fit(words_bunch_tdidf)
hist_visualizer.show()
```

```
{'father': 13319, 'dysfunctional': 11694, 'selfish': 32359, 'drags': 11377, 'kids': 20473, 'dysfunction': 11693, 'run': 31492, 'thar
```

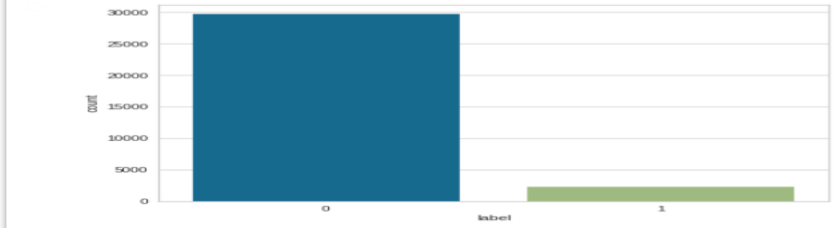


17.Inorder to train and test the data, we need to check the sample sizes. As the data is shorted, we have up sampled the data.

```
[50]
0     29720
1      2242
Name: label, dtype: int64
```

```
import seaborn

#using seaborn plotting the graph
ax = seaborn.countplot(x=Data.iloc[:,1])
```



18.We need to train and test the data, for that we created a modelbuilder function which will be called by different algorithms like SVM, KNN and Logistic regression.

19. From the analysis of the data through training and testing we will get the accuracy score for the data based on each algorithm.

```
#Training and testing the data on the models
for model in models:
  print(model['title'])


  ModelBuilder(model['type'], x_tr, x_te, y_tr, y_te)

  print("\n")
```

```
LogisticRegression
Accuracy_score for training  0.9505654136682609


----------------------

Accuracy_score for testing 0.9457711961622692


----------------------
```

Challenges:

- Model building and applying algorithms is the biggest challenge we face.
- Understanding each and every concept and implementing the methods to create a solution.