Vivekanand Education Society's

Institute of Technology

(Affiliated to University of Mumbai, Approved by AICTE & Recognized by Govt. of Maharashtra)

Department of Information Technology

AIDS - 2 Lab

Experiment - 02

Aim: To build a Cognitive text based application to understand context for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

Roll No.	70
Name	MAYURI SHRIDATTA YERANDE
Class	D20B
Subject	AIDS - 2
Grade:	

EXPERIMENT - 02

AIM: To build a Cognitive text based application to understand context for a Customer service application/ Insurance/ Healthcare Application/ Smarter Cities/ Government etc.

DATASET:
https://drive.google.com/file/d/1bDuFy8mrQUOBTmZh-5KfIwsnHtxNqcnm/view?usp=drivesdk

THEORY:

Sentiment Analysis is a use case of Natural Language Processing (NLP) and comes under the category of text classification. To put it simply, Sentiment Analysis involves classifying a text into various sentiments, such as positive or negative, Happy, Sad or Neutral, etc. Thus, the ultimate goal of sentiment analysis is to decipher the underlying mood, emotion, or sentiment of a text. This is also known as Opinion Mining.

The process involves several steps:

1. **Text Preprocessing:** The text data is cleaned by removing irrelevant information, such as special characters, punctuation, and stopwords.
2. **Tokenization:** The text is divided into individual words or tokens to facilitate analysis.
3. **Feature Extraction:** Relevant features are extracted from the text, such as words, n-grams, or even parts of speech.
4. **Sentiment Classification:** Machine learning algorithms or pre-trained models are used to classify the sentiment of each text instance. This can be achieved through supervised learning, where models are trained on labeled data, or through pre-trained models that have learned sentiment patterns from large datasets.
5. **Post-processing:** The sentiment analysis results may undergo additional processing, such as aggregating sentiment scores or applying threshold rules to classify sentiments as positive, negative, or neutral.
6. **Evaluation:** The performance of the sentiment analysis model is assessed using evaluation metrics, such as accuracy, precision, recall, or F1 score.

Various types of sentiment analysis can be performed, depending on the specific focus and objective of the analysis. Some common types include:

1. **Document-Level Sentiment Analysis:** This type of analysis determines the overall sentiment expressed in a document, such as a review or an article. It aims to classify the entire text as positive, negative, or neutral.
2. **Sentence-Level Sentiment Analysis:** Here, the sentiment of each sentence within a document is analyzed. This type provides a more granular understanding of the sentiment expressed in different text parts.

3. **Aspect-Based Sentiment Analysis:** This approach focuses on identifying and extracting the sentiment associated with specific aspects or entities mentioned in the text. For example, in a product review, the sentiment towards different features of the product (e.g., performance, design, usability) can be analyzed separately.
4. **Entity-Level Sentiment Analysis:** This type of analysis identifies the sentiment expressed towards specific entities or targets mentioned in the text, such as people, companies, or products. It helps understand the sentiment associated with different entities within the same document.
5. **Comparative Sentiment Analysis:** This approach involves comparing the sentiment between different entities or aspects mentioned in the text. It aims to identify the relative sentiment or preferences expressed towards various entities or features.

IMPLEMENTATION:

- Import required libraries

```
[3] import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns; sns.set_theme()
    import nltk
    import re
    from textblob import TextBlob
    from nltk.stem import PorterStemmer
    from textblob import Word
    from nltk.probability import FreqDist
    from nltk.corpus import stopwords
    import string
    nltk.download('stopwords')
    nltk.download('wordnet')
    nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

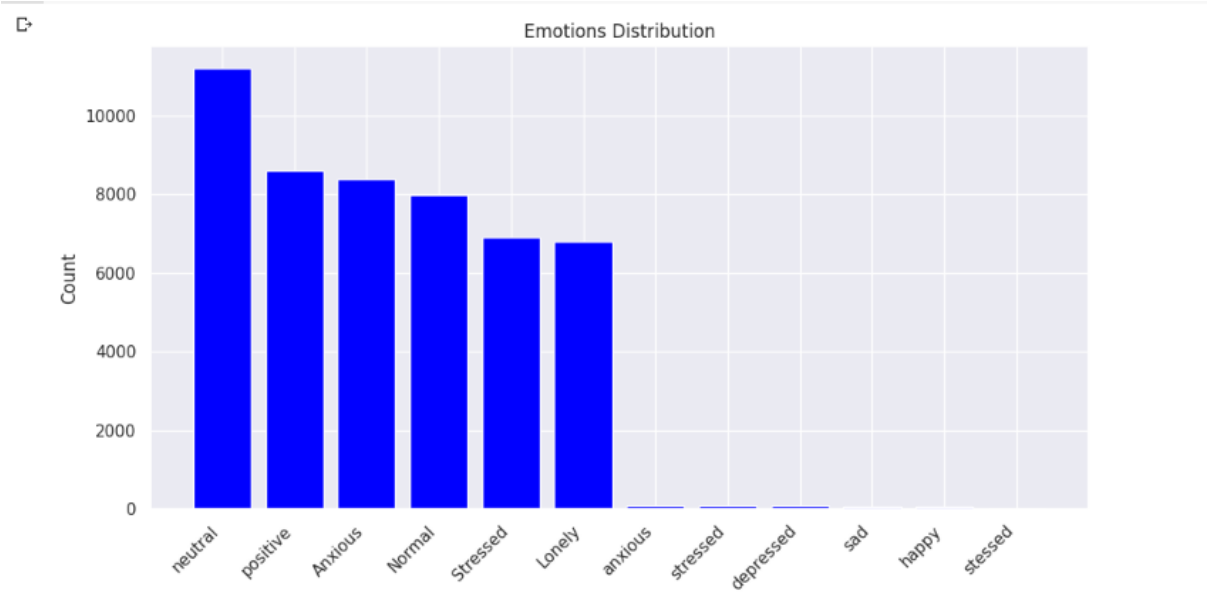
- This code snippet calculates and displays the count of each unique value in the 'labels' column of the DataFrame 'df'.

```
df['labels'].value_counts()
```

```
neutral     11193
positive     8582
Anxious      8389
Normal       7976
Stressed     6910
Lonely       6788
anxious        78
stressed       78
depressed      75
sad            69
happy          51
stessed         1
Name: labels, dtype: int64
```

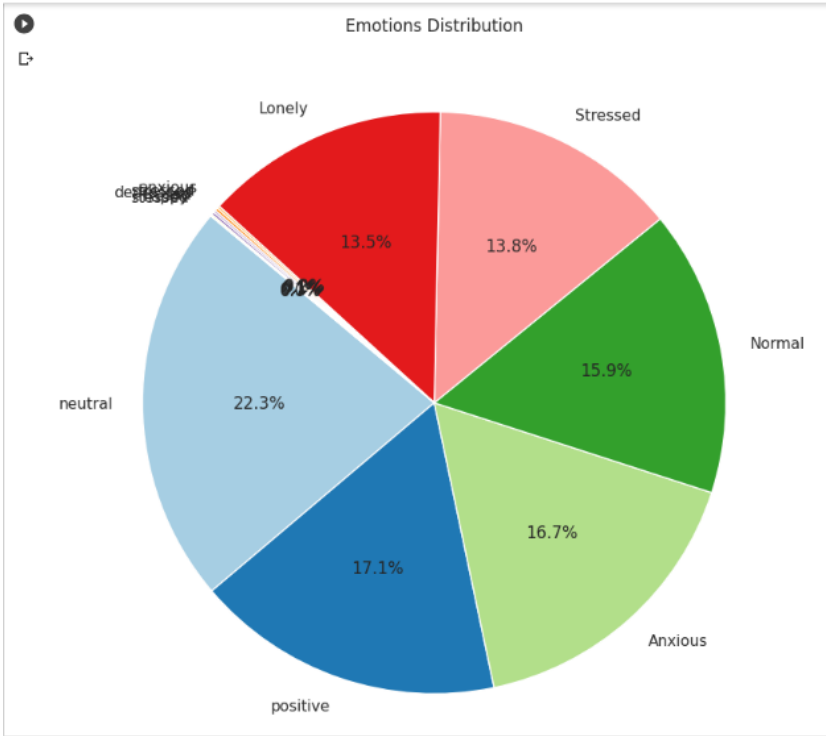
- This code calculates the count of each unique value in the 'labels' column of the DataFrame and then creates a bar plot to visualize the distribution of these counts for different emotions. The x-axis represents emotions, the y-axis represents the count.

```
emotion_counts = df['labels'].value_counts()

# Plotting
plt.figure(figsize=(10, 6))
plt.bar(emotion_counts.index, emotion_counts.values, color='blue')
plt.xlabel('Emotions')
plt.ylabel('Count')
plt.title('Emotions Distribution')
plt.xticks(rotation=45, ha='right')

plt.tight_layout()
plt.show()
```



- Pie chart for the same:-

```
plt.figure(figsize=(8, 8))
plt.pie(emotion_counts.values, labels=emotion_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)
plt.title('Emotions Distribution')

plt.tight_layout()
plt.show()
```



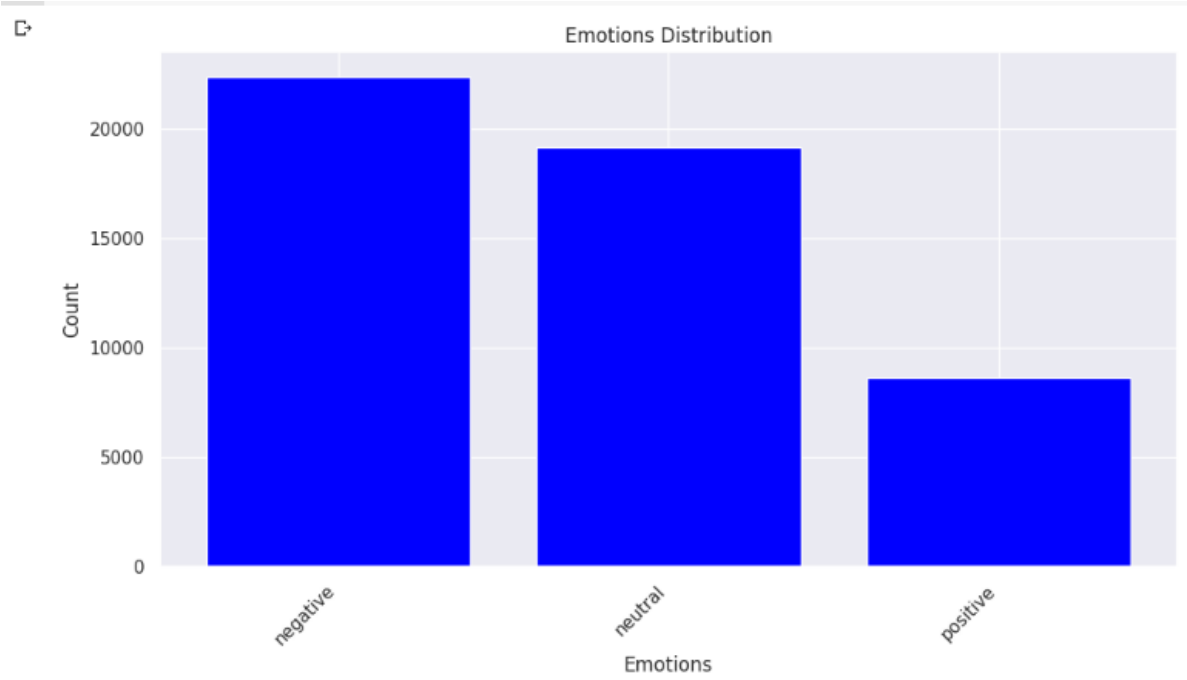
- Now we will divide our parameters into 3 parts: Negative, Positive and Neutral. The label count is:

```
df['labels'].value_counts()
```

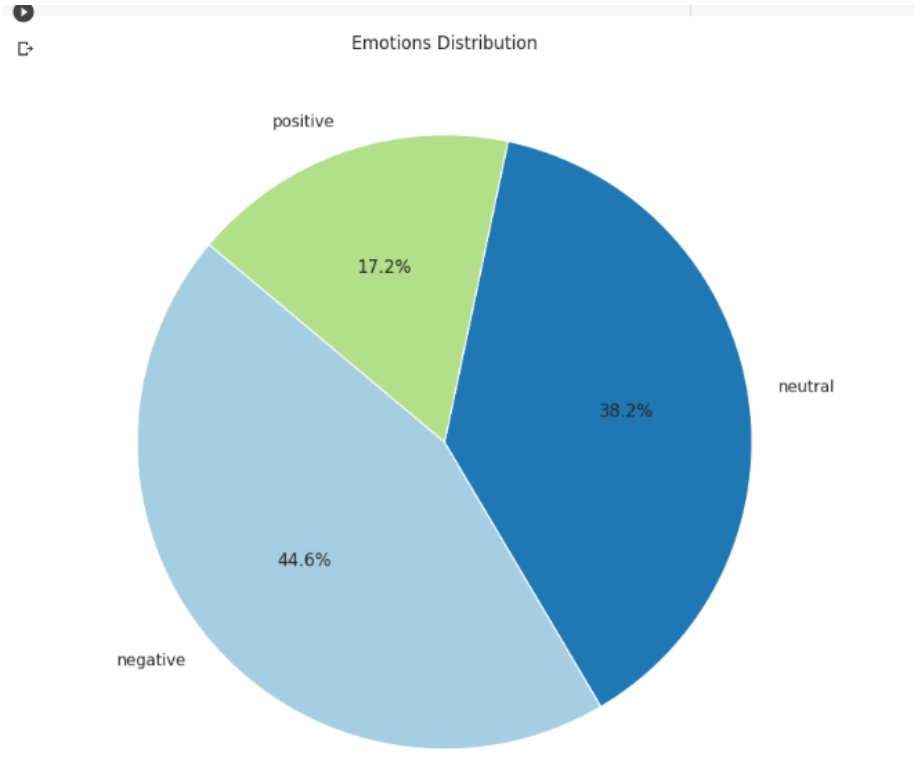
```
negative    22388
neutral     19169
positive     8633
Name: labels, dtype: int64
```

- Plotting a new bar graph and pie chart to show the bifurcation of positive , negative and neutral.

```
[10] emotion_counts = df['labels'].value_counts()

     # Plotting
     plt.figure(figsize=(10, 6))
     plt.bar(emotion_counts.index, emotion_counts.values, color='blue')
     plt.xlabel('Emotions')
     plt.ylabel('Count')
     plt.title('Emotions Distribution')
     plt.xticks(rotation=45, ha='right')

     plt.tight_layout()
     plt.show()
```




```
plt.figure(figsize=(8, 8))
plt.pie(emotion_counts.values, labels=emotion_counts.index, autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)
plt.title('Emotions Distribution')

plt.tight_layout()
plt.show()
```



- This code defines a function called `preprocess_text` that takes a DataFrame (`df`) and a column name (`column_name`) as inputs. The function performs stemming, removal of stopwords on the specified column:

  1. Converts the text to lowercase.
  2. Removes punctuation and special characters.
  3. Removes English stop words using NLTK's stopwords corpus.
  4. Lemmatizes the words using the WordNetLemmatizer from NLTK.

```
preproeessed_df_train = preprocess_text(df,"tweets")
preproeessed_df_train
```

```
<ipython-input-12-c38664b83d9c>:4: FutureWarning: The default value of regex will change from True to False in a future version.
  df[column_name] = df[column_name].str.replace('[^\w\s]','')
```

	tweets	labels
0	make mind	neutral
1	baked dinner yummy cant wait new short stack t...	neutral
2	hah thanks	positive
3	morning blessed day	positive
4	lithium	neutral

- In the provided code: `X_train` holds the preprocessed text data from the 'tweets' column of a DataFrame called `preproeessed_df_train`. y_train` contains the emotion or label information from the 'labels' column of the original DataFrame `df`. These variables are set up to be used for training a machine learning model where `X_train` represents the input text data, and `y_train` represents the corresponding output labels.

```
[14] X_train=preproeessed_df_train['tweets']
     y_train=df['labels']
```

```
[15] X_train.head()
```

```
0                                        make mind
1    baked dinner yummy cant wait new short stack t...
2                                       hah thanks
3                                  morning blessed day
4                                          lithium
Name: tweets, dtype: object
```

- Now we perform Vectorization and Also perform Classification using multinomial Naive Bayes Algorithm

```
[16] from sklearn.feature_extraction.text import TfidfVectorizer
     from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.naive_bayes import MultinomialNB
```

```
[17] count_vect = CountVectorizer()
     count_vect.fit(X_train)
     count_vect_transform = count_vect.transform(X_train)
```

```
[18] count_vect_transform
```

```
<50190x35705 sparse matrix of type '<class 'numpy.int64'>'
        with 425454 stored elements in Compressed Sparse Row format>
```

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
```

```
[20] tfidf_vect = TfidfTransformer()
     X_train_tfidf = tfidf_vect.fit(count_vect_transform)
     X_train_tfidf_transformer = X_train_tfidf.transform(count_vect_transform)
```

```
clf=MultinomialNB().fit(X_train_tfidf_transformer, y_train)
```

```
[22] y_pred = clf.predict(X_train_tfidf_transformer)
```

- The accuracy of our model is 79.59%.

```
[23] from sklearn.metrics import accuracy_score
```

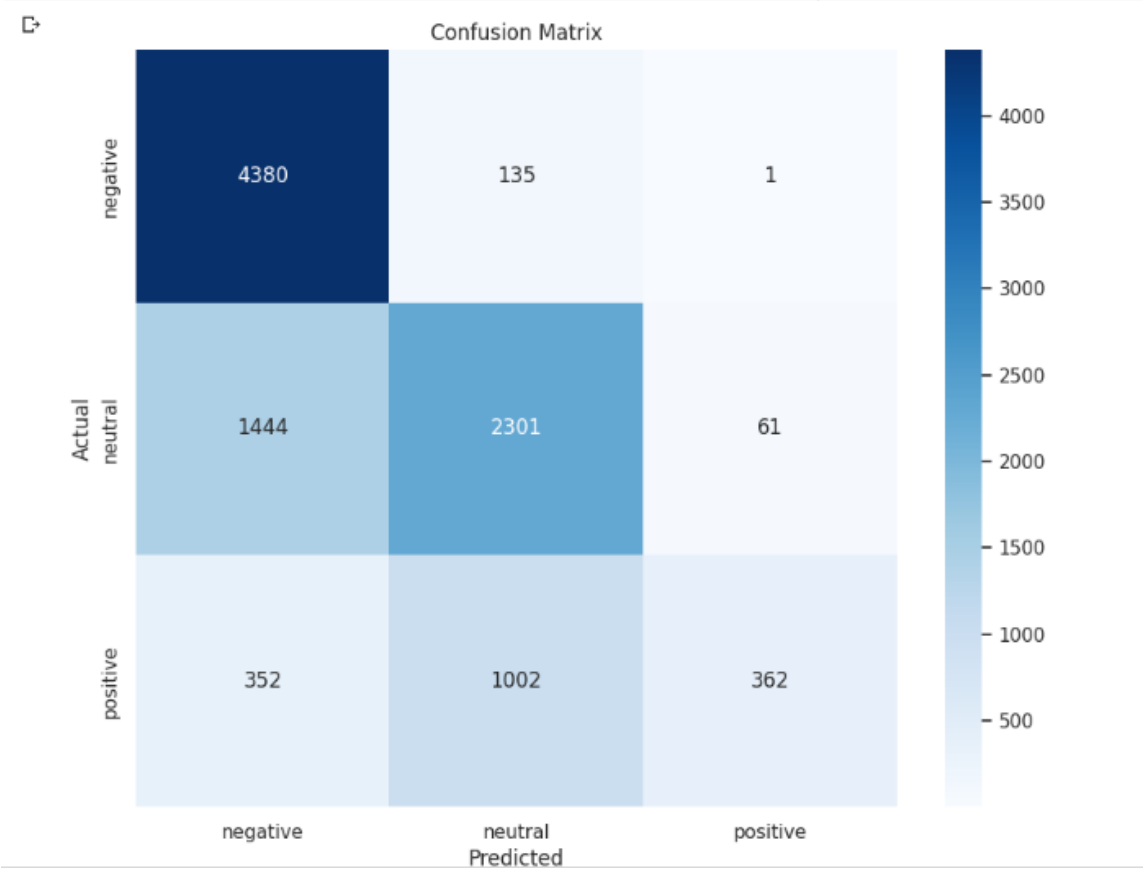
```
[24] accuracy = accuracy_score(y_train, y_pred)
     print("Accuracy:", accuracy)
```

```
Accuracy: 0.7959752938832437
```

- Confusion matrix:
  The output heatmap visually represents how well the classifier's predictions match the actual labels. Diagonal cells represent correct predictions, and off-diagonal cells indicate misclassifications. The x-axis represents predicted labels, while the y-axis represents actual labels. The intensity of color in each cell corresponds to the number of occurrences in the confusion matrix, with darker shades indicating higher counts.

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import seaborn as sns

X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2, random_state=42)


count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
```

```
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

clf = MultinomialNB().fit(X_train_tfidf, y_train)

X_test_counts = count_vect.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)

y_pred = clf.predict(X_test_tfidf)

conf_mat = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10, 8))
sns.heatmap(conf_mat, annot=True, fmt="d", cmap="Blues",
            xticklabels=clf.classes_, yticklabels=clf.classes_)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



CONCLUSION: Sentiment analysis is an NLP task, which involves classifying texts or parts of texts into a predefined sentiment. We have done sentiment analysis for a tweet dataset. It determines the sentiment from a set of emotions like anxious, stressed, happy, sad, depressed, etc. categorized into Negative, positive and Neutral.