




Text Sentimental Analysis

Submitted By:-

Ayush Kumar(2K21/SE/48)

Bharat Mishra(2K21/SE/54)



Topics

Introduction

Data Collection and Cleaning

Data Preprocessing

Exploratory Data Analysis (EDA)

Feature Engineering

ML Model

Conclusion





Abstract

This report presents the process and results of performing sentiment analysis on a dataset of text data obtained from Kaggle. The objective of the project was to analyze each text and assign a corresponding sentiment label, namely positive, negative, or neutral. The project encompassed data collection, cleaning, preprocessing, feature engineering, model selection, and evaluation. This report outlines the methodology, key findings, and insights gained from the analysis.

Topic 1:

INTRODUCTION



Introduction

- Sentiment analysis involves determining the sentiment expressed in a piece of text, which is a fundamental task in natural language processing (NLP). The objective of this project was to perform sentiment analysis on a text dataset sourced from Kaggle, with the goal of classifying each text into one of three sentiment categories: positive, negative, or neutral.
- Example: Product Reviews on an Online Store
- Imagine you are working for an online store that sells various consumer products. The online store receives numerous product reviews from customers, expressing their thoughts and opinions about the products they purchased. Analyzing these reviews manually would be time-consuming and inefficient. This is where sentiment analysis comes in.

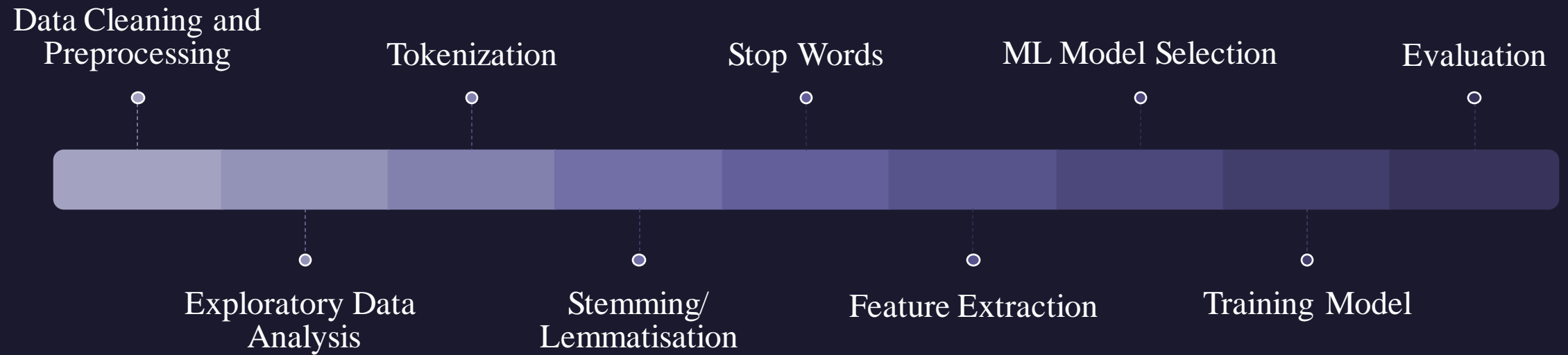


Introduction

- Consider the following product review:
- Review: "I absolutely love this product! It exceeded my expectations and has made my daily routine so much easier."
- In this example, the sentiment expressed in the review is clearly positive. A human reader can easily recognize the positive sentiment based on phrases like "absolutely love," "exceeded my expectations," and "easier." However, for a computer to understand and categorize this sentiment, it needs to follow a systematic process.



Timeline



Data collection and cleaning serve as the cornerstone of any data analysis or machine learning endeavor. These initial steps are pivotal in setting the stage for the rest of the project. In this section, we'll delve into the significance of these steps and discuss the approach for collecting, cleaning, and preparing the data for further analysis. The selection of data involves following steps:

- Identify Problem
- Choose Data according to that problem
- Clean the data (Remove duplicated, Fill Null values with Mode, Mean etc.)
- Removing Punctuation and number's (Text Analysis case)
- Visualize Data
- Correlation hunting

```
df = pd.read_csv('E:/Dataset for intership_AI/Text_sentimental_Analysis/train.csv',  
                 delimiter=',', encoding='ISO-8859-1')  
df.head()
```

	textID	text	selected_text	sentiment	Time of Tweet	Age of User	Country	Population -2020	Land Area (Km²)	Density (P/Km²)
0	cb774db0d1	I'd have responded, if I were going	I'd have responded, if I were going	neutral	morning	0-20	Afghanistan	38928346	652860.0	60
1	549e992a42	Sooo SAD I will miss you here in San Diego!!!	Sooo SAD	negative	noon	21-30	Albania	2877797	27400.0	105
2	088c60f138	my boss is bullying me...	bullying me	negative	night	31-45	Algeria	43851044	2381740.0	18
3	9642c003ef	what interview! leave me alone	leave me alone	negative	morning	46-60	Andorra	77265	470.0	164
4	358bd9e861	Sons of ****, why couldn't they put them on t...	Sons of ****,	negative	noon	60-70	Angola	32866272	1246700.0	26

Topic 2:

Data Collection and Cleaning



The data which we need for our problem is Text Base data and that Data is the tweet data for many users which is basically a Social Media base data which is shown in figure 1 above. It has 6 Columns in which there is a,

- textID
- text
- text_selected
- sentiment
- Time of Tweet
- Age of user
- Country
- Population
- Land area
- Density

```
#Check missing values in the dataset
```

```
df.isna().sum().sum()
```

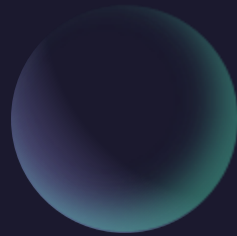
```
#after data cleaning
```

```
df = df[['text', 'sentiment']]  
df
```

	text	sentiment
0	I'd have responded, if I were going	neutral
1	Sooo SAD I will miss you here in San Diego!!!	negative
2	my boss is bullying me...	negative
3	what interview! leave me alone	negative
4	Sons of ****, why couldn't they put them on t...	negative
...
27476	wish we could come see u on Denver husband I...	negative
27477	I've wondered about rake to. The client has ...	negative
27478	Yay good for both of you. Enjoy the break - y...	positive
27479	But it was worth it ****.	positive
27480	All this flirting going on - The ATG smiles...	neutral

Topic 3:

Data Collection and Cleaning



```
## remove stopwords and punctuation marks
stuff_to_be_removed = list(stopwords.words('english'))+list(punctuation)
stemmer = LancasterStemmer()

corpus = df['text'].tolist()
print(len(corpus))
print(corpus[0])

27480
I'd have responded, if I were going

%time
final_corpus = []
final_corpus_joined = []
for i in df.index:

    text = re.sub('[^a-zA-Z]', ' ', df['text'][i])
    #Convert to lowercase
    text = text.lower()
    #remove tags
    textre.sub("</?.*?>", "", &#38;#38;text)

    # remove special characters and digits
    textre.sub("(\\d|\\W)+", " ",text)

    ##Convert to list from string
    text = text.split()

    #lemmatization
    lem = WordNetLemmatizer()
    text = [lem.lemmatize(word) for word in text
             if not word in stuff_to_be_removed]
    text1 = " ".join(text)
    final_corpus.append(text)
    final_corpus_joined.append(text1)

CPU times: total: 4.41 s
Wall time: 4.41 s
```

```
data_cleaned = pd.DataFrame()
data_cleaned['text'] = final_corpus_joined
data_cleaned['sentiment'] = df['sentiment'].values

data_cleaned['sentiment'].value_counts()

neutral    11117
positive    8582
negative    7781
Name: sentiment, dtype: int64

data_cleaned.head()
```

	text	sentiment
0	responded going	neutral
1	soooo sad miss san diego	negative
2	bos bullying	negative
3	interview leave alone	negative
4	son put release already bought	negative

Text Preprocessing is traditionally an important step for Natural Language Processing (NLP) tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better. The Preprocessing steps taken are:

- 1.Lower Casing:** Each text is converted to lowercase.
- 2.Replacing URLs:** Links starting with "http" or "https" or "www" are replaced by "URL".
- 3.Replacing Emojis:** Replace emojis by using a pre-defined dictionary containing emojis along with their meaning. (eg: ":)") to "EMOJIsmile")
- 4.Replacing Usernames:** Replace @Usernames with word "USER". (eg: "@Kaggle" to "USER")
- 5.Removing Non-Alphabets:** Replacing characters except Digits and Alphabets with a space.
- 6.Removing Consecutive letters:** 3 or more consecutive letters are replaced by 2 letters. (eg: "Heyyyy" to "Heyy")
- 7.Removing Short Words:** Words with length less than 2 are removed.
- 8.Removing Stopwords:** Stopwords are the English words which does not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. (eg: "the", "he", "have")
- 9.Lemmatizing:** Lemmatization is the process of converting a word to its base form. (e.g:“Great” to “Good”)

```
data_eda = pd.DataFrame()
data_eda['text'] = final_corpus
data_eda['sentiment'] = df['sentiment'].values
data_eda.head()
```

	text	sentiment
0	[responded, going]	neutral
1	[soooo, sad, miss, san, diego]	negative
2	[bos, bullying]	negative
3	[interview, leave, alone]	negative
4	[son, put, release, already, bought]	negative




Topic 4:

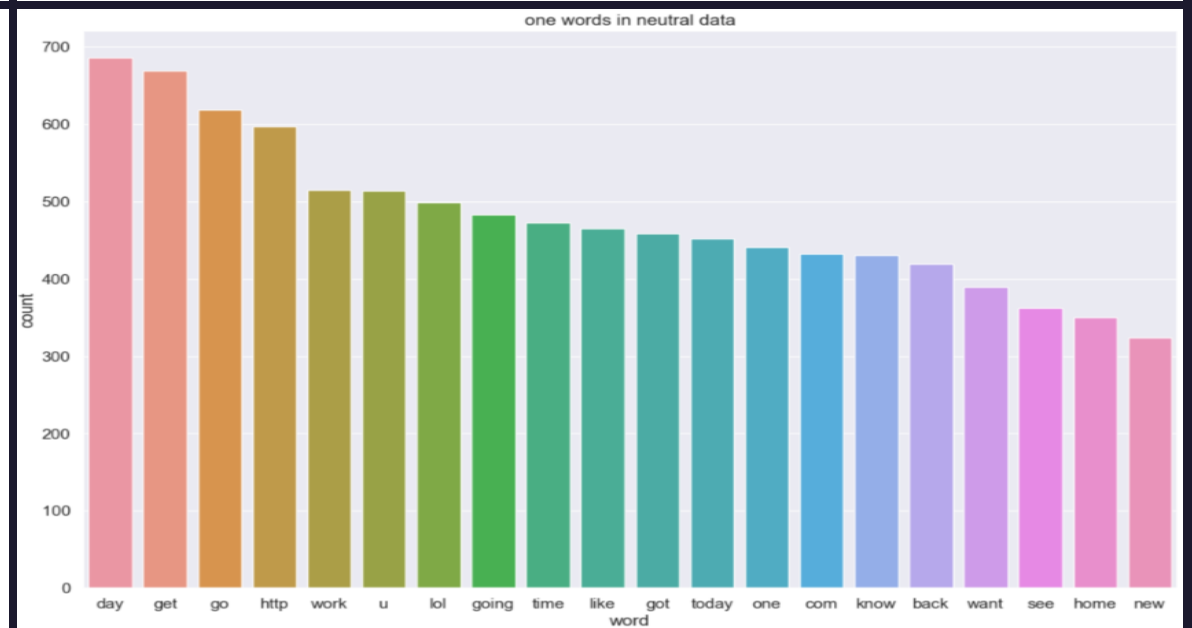
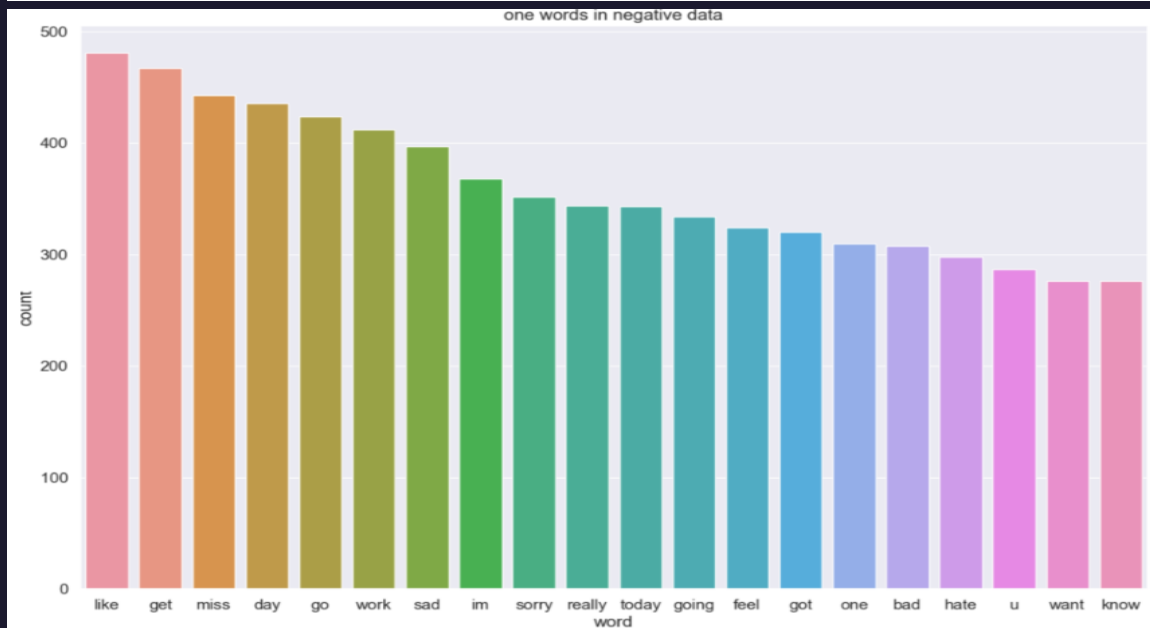
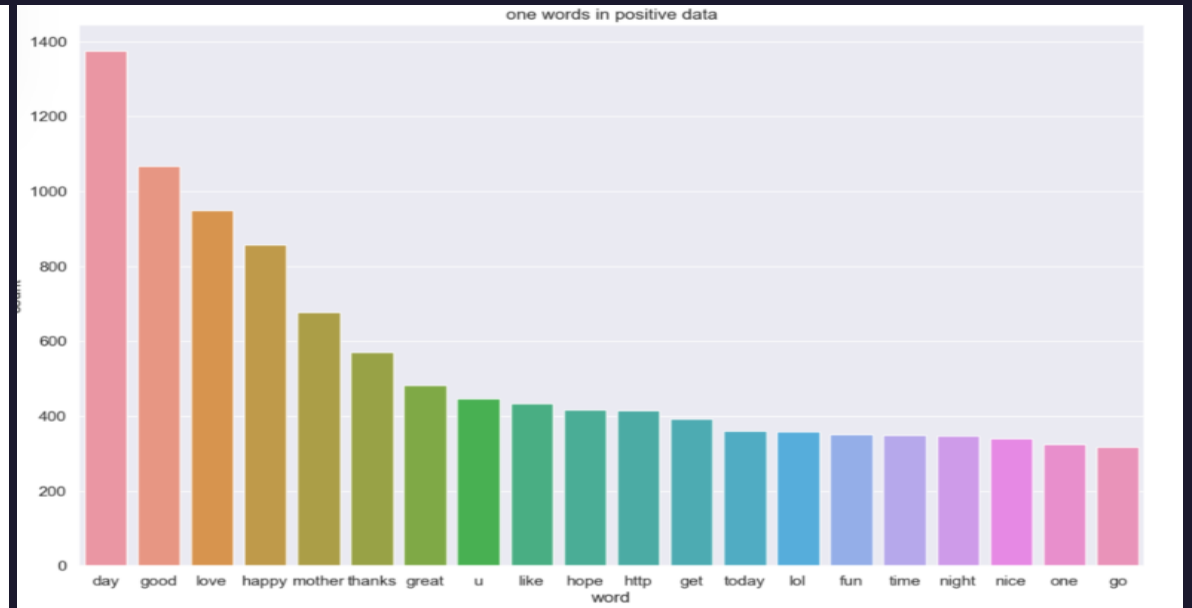
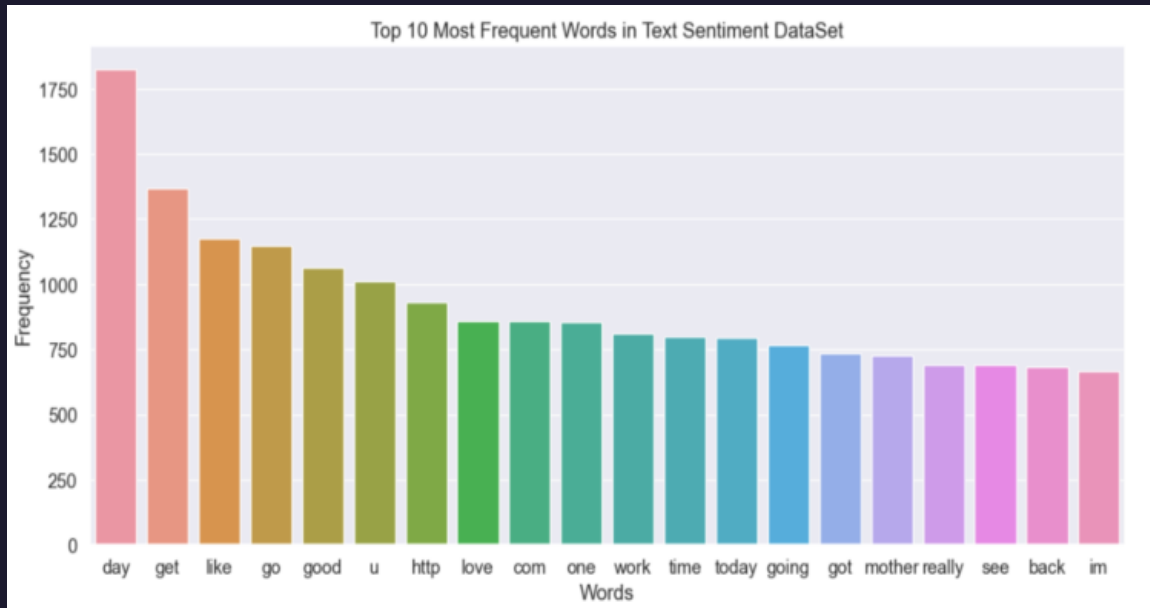
Data Collection and Cleaning



In this process we have to Visualize our data which give us an insight knowledge about our data. It is not possible for humans to read all the rows and columns of data set. So to make it easy we use Graphs to see the behavior or pattern in the data. It helps us in following,

- Visualization of word frequency to identify common words.
 - Analysis of sentiment label distribution to understand class imbalances.
 - Creation of word clouds to visualize sentiment-specific word usage.
- 

Word frequency:

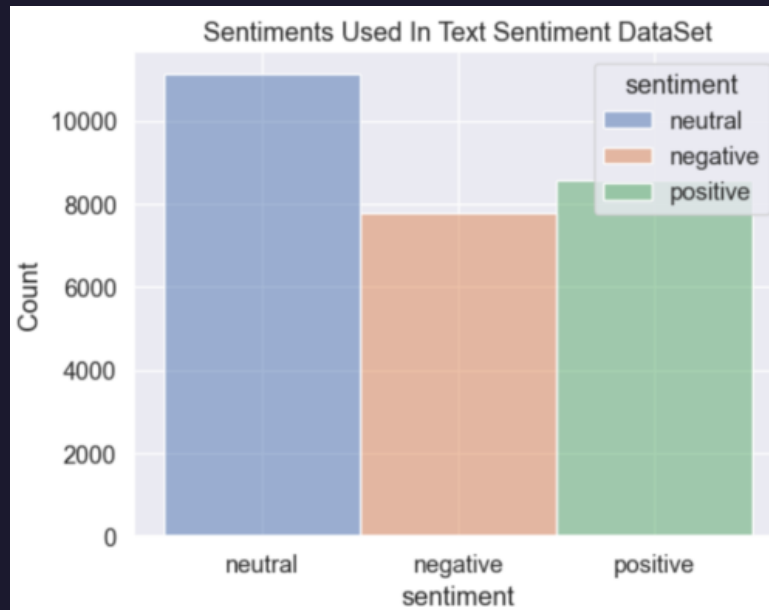


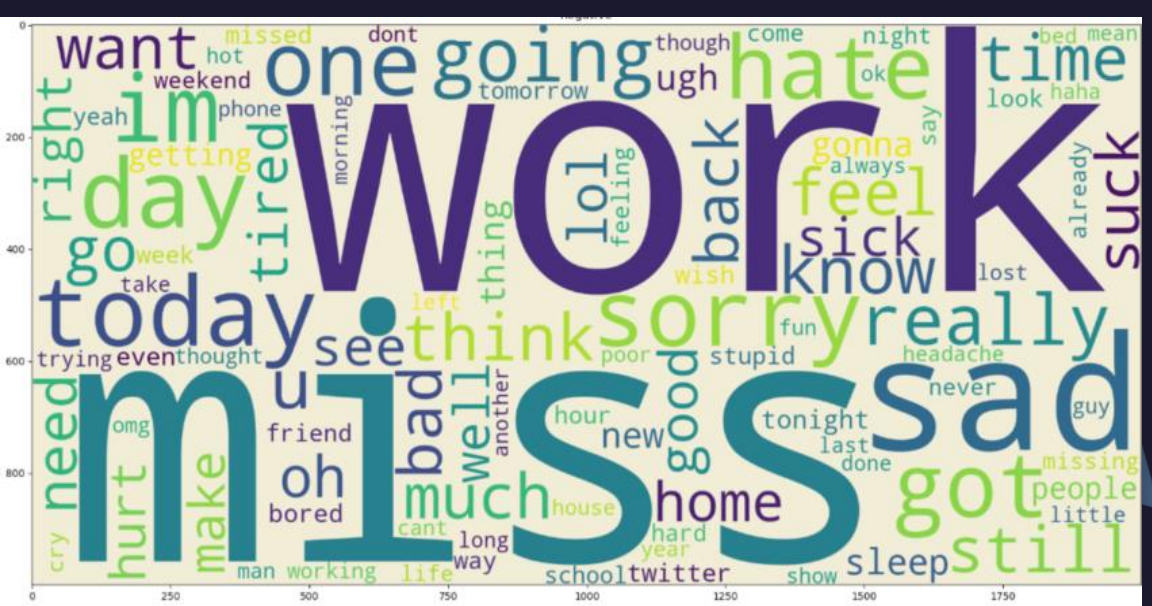
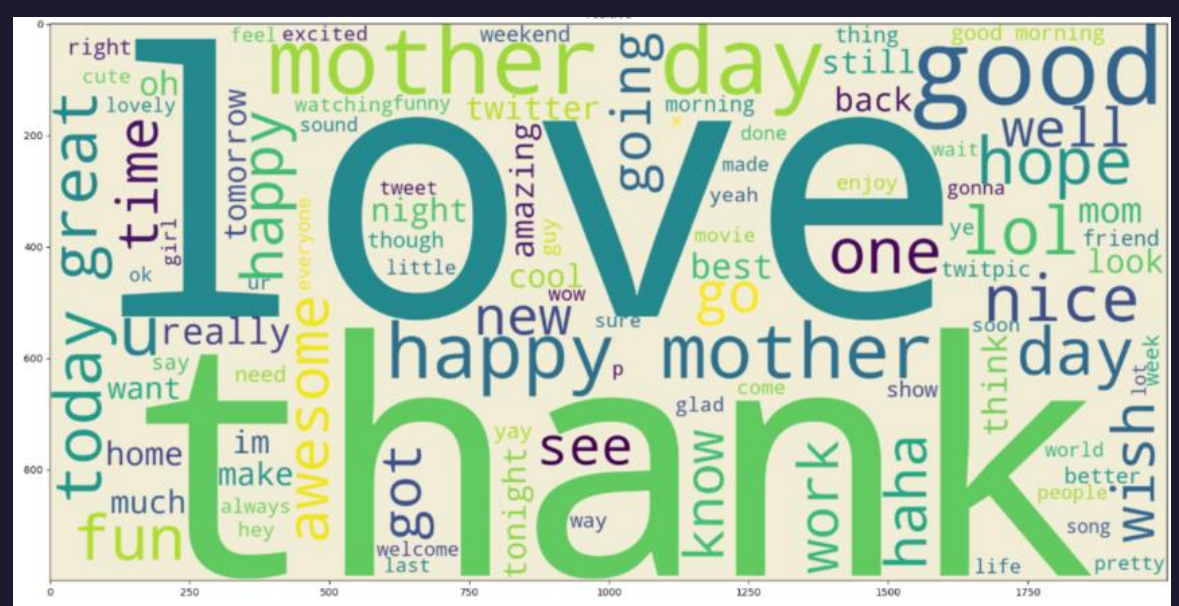
Sentiment Distribution:

```
# Visualising the Sentiment Data
```

```
sns.histplot(data = data_eda, x = data_eda['sentiment'], hue = 'sentiment').set(title=f'Sentiments Used In Text Sentiment Dat
```

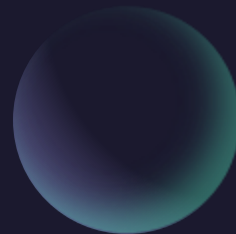
```
[Text(0.5, 1.0, 'Sentiments Used In Text Sentiment DataSet')]
```





Topic 5:

Feature Engineering



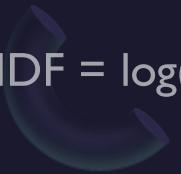
TF-IDF stands for Term Frequency-Inverse Document Frequency. It's a numerical representation used in natural language processing and information retrieval to evaluate the importance of words in a document within a collection of documents (corpus). TF-IDF is particularly useful for feature engineering in text analysis tasks like text classification, sentiment analysis, and information retrieval.

Here's how TF-IDF works:

1. Term Frequency (TF): This measures how often a word appears in a specific document. It's calculated as the ratio of the number of times a word (term) appears in the document to the total number of words in that document. The idea is that words appearing frequently within a document are likely to be important to the document's content.

$$TF = (\text{Number of times the term appears in the document}) / (\text{Total number of words in the document})$$

2. Inverse Document Frequency (IDF): This measures the rarity of a word across all documents in the corpus. Words that appear in many documents are considered less informative, while words that appear in fewer documents are given higher weight. IDF is calculated as the logarithm of the total number of documents divided by the number of documents containing the term.


$$IDF = \log((\text{Total number of documents}) / (\text{Number of documents containing the term}))$$

3.TF-IDF:The TF-IDF score for a term in a specific document combines both the term's frequency in that document (TF) and its rarity across all documents (IDF).The higher the TF-IDF score, the more important the term is to the document's content.

$$TF-IDF = TF * IDF$$

TF-TDF performs various tasks which are given below:

- Convert text into Tokens
- Lower case the text and remove punctuations by default
- Calculate how many times a word occurs in document
- Find the Importance of that word in the Document
- Make a Vector based on above operations which have score of each word

The important point here is that we should do first 4 steps before feature extraction otherwise he counts “is”, “am” or other words which are irrelevant which effects our accuracy score of model.

```
data_cleaned['text']
0      responded going
1      sooo sad miss san diego
2      bos bullying
3      interview leave alone
4      son put release already bought
...
27475  wish could come see u denver husband lost job ...
27476  wondered rake client made clear net force devs...
27477  yay good enjoy break probably need hectic week...
27478                                     worth
27479      flirting going atg smile yay hug
Name: text, Length: 27480, dtype: object
```

```
data_cleaned['text']
0      responded going
1      sooo sad miss san diego
2      bos bullying
3      interview leave alone
4      son put release already bought
...
27475  wish could come see u denver husband lost job ...
27476  wondered rake client made clear net force devs...
27477  yay good enjoy break probably need hectic week...
27478                                     worth
27479      flirting going atg smile yay hug
Name: text, Length: 27480, dtype: object
```

Topic 6:

Model Selection



- The choice of model influences the accuracy of sentiment classification. We consider traditional machine learning models like Logistic Regression and Support Vector Machines, along with deep learning models like Recurrent Neural Networks and Transformers.. Selection of right model is depending on the following things,
- Type of Problem (Regression, Classification, Clustering)
 - Nature of data
 - Data Size
 - Data Complexity
 - Computation Time and Cost
 - Scalability
- Machine Learning Models:
- Logistic Regression: This simple yet effective model is often used as a baseline. It's suitable when the dataset is not overly complex and linear separability is reasonable.
- Support Vector Machines (SVM): SVMs can handle non-linear relationships between features and labels. They work well for sentiment analysis tasks with a clear margin of separation between sentiment classes.



Deep Learning Models:

Recurrent Neural Networks (RNNs): RNNs are suitable for sequential data like text. They can capture contextual information and are effective when word order matters. However, they might struggle with long-range dependencies.

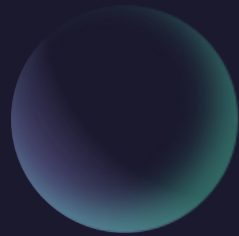
Transformers (e.g., BERT, GPT): Transformers have revolutionized NLP tasks. BERT, for example, is a pre-trained language model that excels in capturing contextual information and relationships between words. Fine-tuning BERT for sentiment analysis can yield excellent results.

	Model	Accuracy	Precision	Recall	F1_Score
0	Logistic Regression	0.680	0.700	0.670	0.680
1	Support Vector Machine	0.690	0.720	0.680	0.690
2	BERT(transformers)	0.792	0.798	0.794	0.799
3	LSTM(RNN)	0.400	0.160	0.400	0.230

RESULT

Topic 7:

Conclusion



Conclusion

- **1.Imbalanced Data:** If the dataset has a significant class imbalance (e.g., more positive than negative samples), the model might become biased towards the majority class. Solution: Implement techniques like oversampling, undersampling, or using different evaluation metrics that account for class imbalance.
- **2.Sarcasm and Irony:** Sentiments conveyed through sarcasm or irony can be challenging for models to accurately interpret. Solution: Incorporate contextual information, advanced sentiment analysis models, or sentiment lexicons that capture nuanced language.
- **3. Contextual Understanding:** Some words might have different sentiments based on context. For instance, "bad" can refer to product quality or imply a negative experience. Solution: Utilize models like BERT or other contextual embeddings that capture word meaning in context.
- **4.Domain Specificity:** Sentiment expressions can vary across domains, and models trained on one domain may not perform well on another. Solution: Fine-tune or train models on domain-specific data to capture domain-specific sentiment patterns.
- **5.Data Noise:** Noisy text data with typos, slang, and grammatical errors can affect model performance. Solution: Implement robust text preprocessing techniques, possibly using spell-checkers and language correction tool

Thank You

