Mini Project Report on

Sentiment Analysis Of Social Media Posts

Submitted in partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING
With AI & DS

Submitted by: **University Roll No. 2023936**
Sumit Kumar

Under the Mentorship of

Dr. Gulshan Dhasmana
Assistant Professor

Department of Computer Science and Engineering
Graphic Era (Deemed to be University)
Dehradun, Uttarakhand
January-2024

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the project report entitled **"Sentiment Analysis "** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineering of the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Dr. Gulshan Dhasmana , Assistant Professor** , Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name: Sumit Kumar University Roll no: 2023936

Table of Contents

Chapter 1

Introduction

- **Introduction**

Sarcasm is a form of expression often used to convey the opposite of the literal meaning of words, typically to mock or convey contempt. Detecting sarcasm in text is critical for accurate sentiment analysis in fields such as social media monitoring, customer feedback analysis, and conversational AI. Unlike straightforward text sentiment analysis, sarcasm detection requires a deeper understanding of linguistic nuances and context, making it a challenging computational task.

- **Problem Statement**

Despite advancements in natural language processing (NLP), sarcasm detection remains a challenging task due to its subjective nature, dependency on context, and cultural nuances. Existing sentiment analysis systems often fail to detect sarcasm, leading to incorrect interpretations and insights.

- **How it works :**
    - **Preprocessing and analyzing the dataset.**
    - **Training the Model.**
    - **Model's performance and Output.**

- **Why do we need it?**

This study focuses on textual sarcasm detection in English language datasets. The project uses machine learning models trained on structured datasets, excluding multimodal inputs like images or voice.

Chapter 2

Literature Survey

2.1 What is Machine learning?

Machine learning (ML) is a fascinating field within artificial intelligence (AI) that focuses on creating systems that can learn from data and improve their performance over time, without being explicitly programmed for every task. Imagine a computer that can analyze huge amounts of data, uncover hidden patterns, and then use these insights to make predictions or even generate new content! That's essentially what machine learning is all about.

Key features:

- **Data-driven:** ML algorithms are trained on large datasets of examples, such as images, text, or numbers. As they process more data, they learn to identify patterns and relationships within that data.
- **Prediction and decision making**: The ultimate goal of ML is to use these learned patterns to make predictions on new, unseen data. This could involve tasks like classifying emails as spam or not spam, recommending products based on your purchase history, or even translating languages on the fly.
- **Different types of learning:** There are various types of machine learning, each with its own strengths and weaknesses. Some common types include supervised learning, unsupervised learning, and reinforcement learning.
- **Impact across industries:** Machine learning is already having a profound impact on various industries, from healthcare and finance to entertainment and transportation. Its applications are constantly growing as researchers and developers push the boundaries of what's possible.

2.2 Natural Language Processing (NLP)

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) and linguistics that focuses on enabling machines to understand, interpret, generate, and respond to human language in a way that is both meaningful and useful. It bridges the gap between human communication and computer systems by analyzing and processing text and speech data using computational techniques.

What makes Uniqueness in NLP:

- **Diverse Applications:** NLP powers a wide range of applications, from chatbots and voice assistants to text summarization, sentiment analysis, and machine translation.
- **Human-like Understanding:** NLP enables machines to process, understand, and generate language in ways that mimic human behavior.
- **Contextual Mastery:** Modern NLP excels at capturing context through techniques like attention mechanisms, enabling it to understand nuances, idioms, and even sarcasm in some cases.
- **Personalization:** NLP can tailor responses based on user behavior, preferences, or history, making it highly adaptable for personal or industry-specific use cases.
- **Cross-Language Adaptability:** NLP models like multilingual transformers can handle multiple languages, facilitating cross-cultural communication and understanding.

But here's the catch:

- **Ambiguity in Language:** Words can have multiple meanings based on context. For example, "bank" could refer to a financial institution or the side of a river.
- **Bias in Data:** NLP models are only as unbiased as the data they're trained on. If the training data contains biases (e.g., gender, racial), the model may perpetuate them in its outputs.

- **Resource Intensive:** Training state-of-the-art NLP models requires significant computational resources and large datasets, which can be a barrier for smaller organizations.
- **Cultural Nuances and Low-Resource Languages:** NLP struggles with understanding idiomatic expressions, cultural subtleties, and languages with limited training data (low-resource languages).
- **Understanding Beyond Text:** While NLP can process textual data well, it might fail to account for non-verbal cues, tone, or visual context, which are essential in human communication.
- **Ethical Concerns:** Misuse of NLP technologies, such as generating fake news, deepfake texts, or exploiting conversational AI for scams, poses ethical dilemmas.

NLP's uniqueness lies in its ability to bridge human and machine communication, but its true potential can only be realized by addressing its inherent challenges with innovation, responsibility, and ethical foresight.

2.3 Logistic Regression from sklearn.linear_model:

Logistic Regression is widely used in various fields for binary classification tasks, and its process involves several steps from data preparation to prediction. In this notebook, it is used to predict whether a comment is sarcastic (1) or not (0).

How Logistic Regression is Used:

1. **Define the Problem:** Identify the binary classification task (e.g., spam vs. not spam, disease vs. no disease, sarcasm vs. no sarcasm).

2. **Data Preparation:** Gather labeled data for training (inputs and corresponding binary outcomes).

3. **Feature Selection:** Select or engineer the most relevant features to improve the model's performance and interpretability.

4. **Model Training:** Apply the logistic regression algorithm to the training dataset. Use an optimization algorithm like Gradient Descent to minimize the log-loss (cost function) and estimate the weights (β).

5. **Model Evaluation:** Proportion of correct predictions. Ratio of true positives to predicted positives.

6. **Model Deployment:** Use the trained model to predict probabilities on new data. Classify outcomes by setting a threshold (default is 0.5 but can be adjusted based on the application).

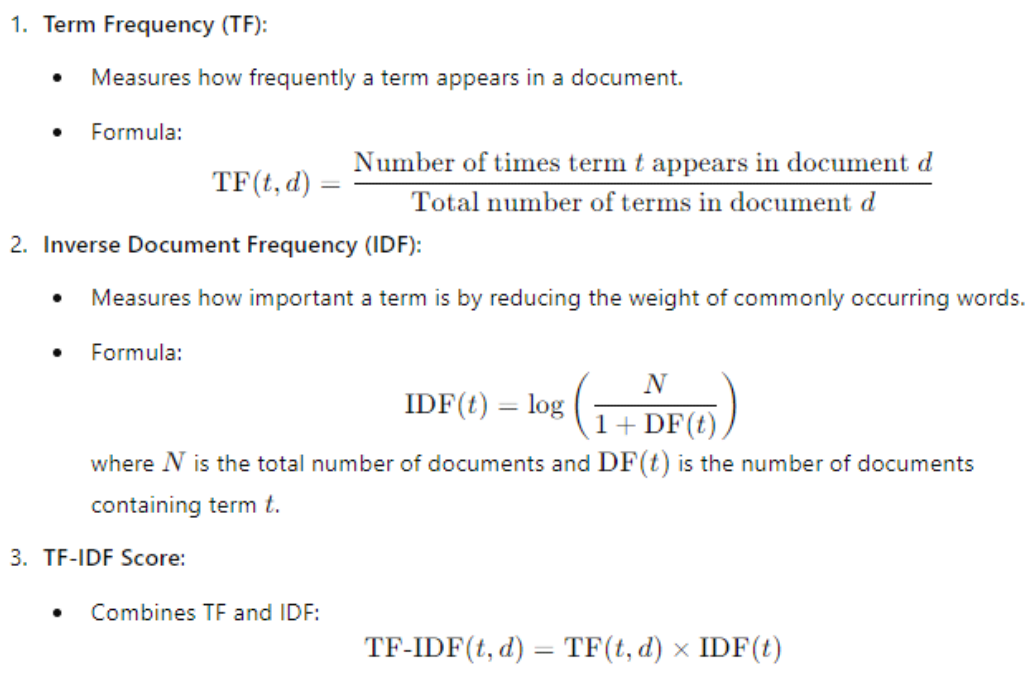
7. **Interpret Results:** Analyze the model coefficients to understand feature importance. Use the probabilities to make informed decisions (e.g., flag emails as spam, recommend additional testing in medical diagnostics).

2.4 TfidfVectorizer from sklearn.feature_extraction.text :

TfidfVectorizer is a powerful tool for converting raw text data into numerical representations (feature vectors) that can be used for machine learning tasks like text classification, clustering, or sentiment analysis. It combines two concepts: Term Frequency (TF) and Inverse Document Frequency (IDF).

What is TF-IDF?

TF-IDF stands for Term Frequency-Inverse Document Frequency. It measures the importance of words in a document relative to a collection of documents (corpus).



1. **Term Frequency (TF):**
   - Measures how frequently a term appears in a document.
   - Formula:

$$\mathrm{TF}(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

2. **Inverse Document Frequency (IDF):**
   - Measures how important a term is by reducing the weight of commonly occurring words.
   - Formula:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{1+\mathrm{DF}(t)}\right)$$

   where N is the total number of documents and $\mathrm{DF}(t)$ is the number of documents containing term t.

3. **TF-IDF Score:**
   - Combines TF and IDF:

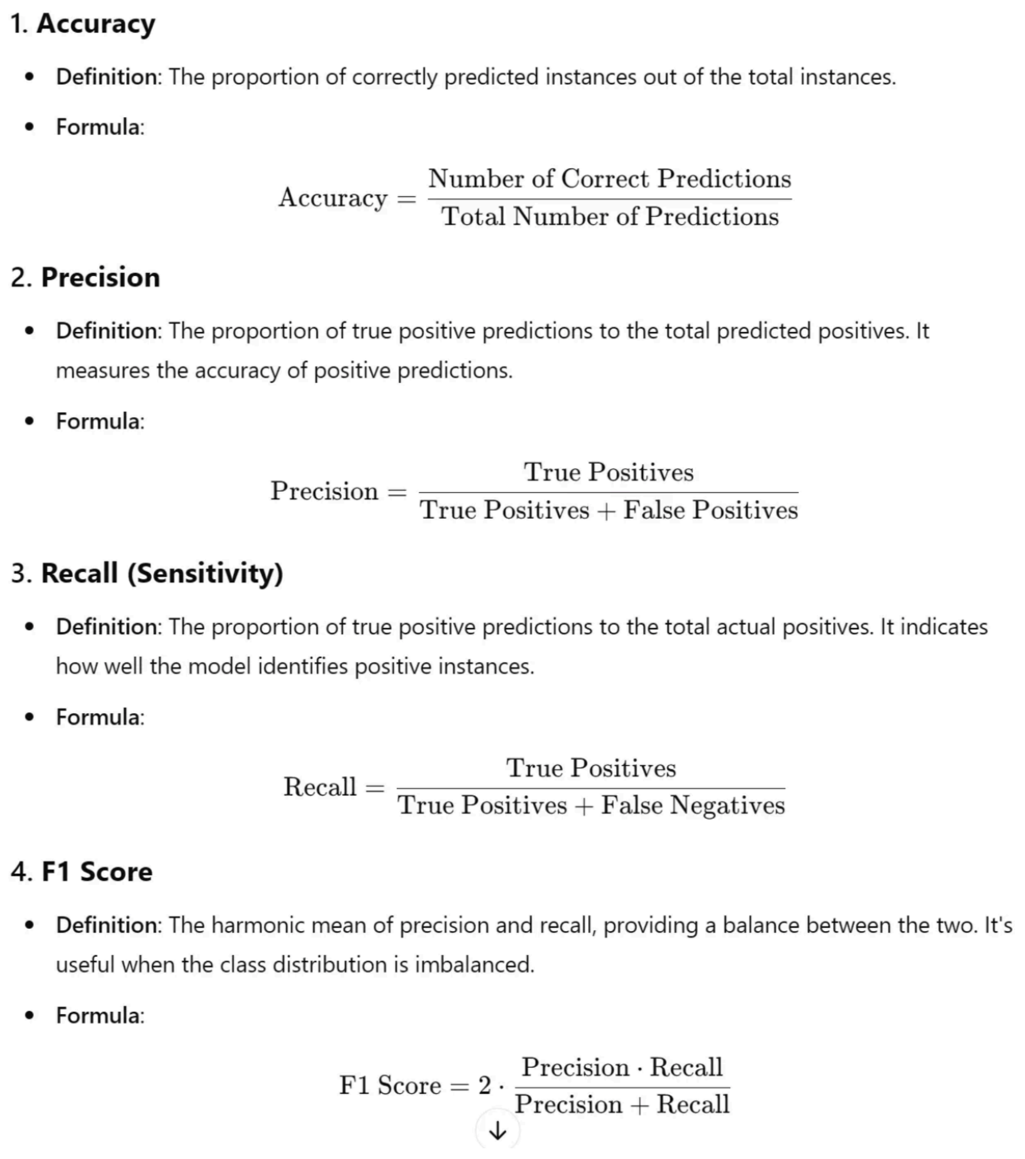
$$\text{TF-IDF}(t,d) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

2.5 Pipeline from sklearn.pipeline

A pipeline in machine learning is a structured workflow that automates the sequence of data preprocessing, feature extraction, model training, and evaluation steps. In scikit-learn, the Pipeline class is often used to streamline and automate these processes, ensuring consistency and reproducibility.

2.6 confusion_matrix from sklearn.metrics

A confusion matrix is a performance evaluation tool for classification models. It summarizes the model's predictions by comparing the predicted and actual class labels. The matrix provides insights into the types of errors made by the classifier.



1. **Accuracy**
   - **Definition**: The proportion of correctly predicted instances out of the total instances.
   - **Formula**:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

2. **Precision**
   - **Definition**: The proportion of true positive predictions to the total predicted positives. It measures the accuracy of positive predictions.
   - **Formula**:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

3. **Recall (Sensitivity)**
   - **Definition**: The proportion of true positive predictions to the total actual positives. It indicates how well the model identifies positive instances.
   - **Formula**:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

4. **F1 Score**
   - **Definition**: The harmonic mean of precision and recall, providing a balance between the two. It's useful when the class distribution is imbalanced.
   - **Formula**:

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chapter 3

Methodology

3.1 Requirement Analysis

a) Basic knowledge of python

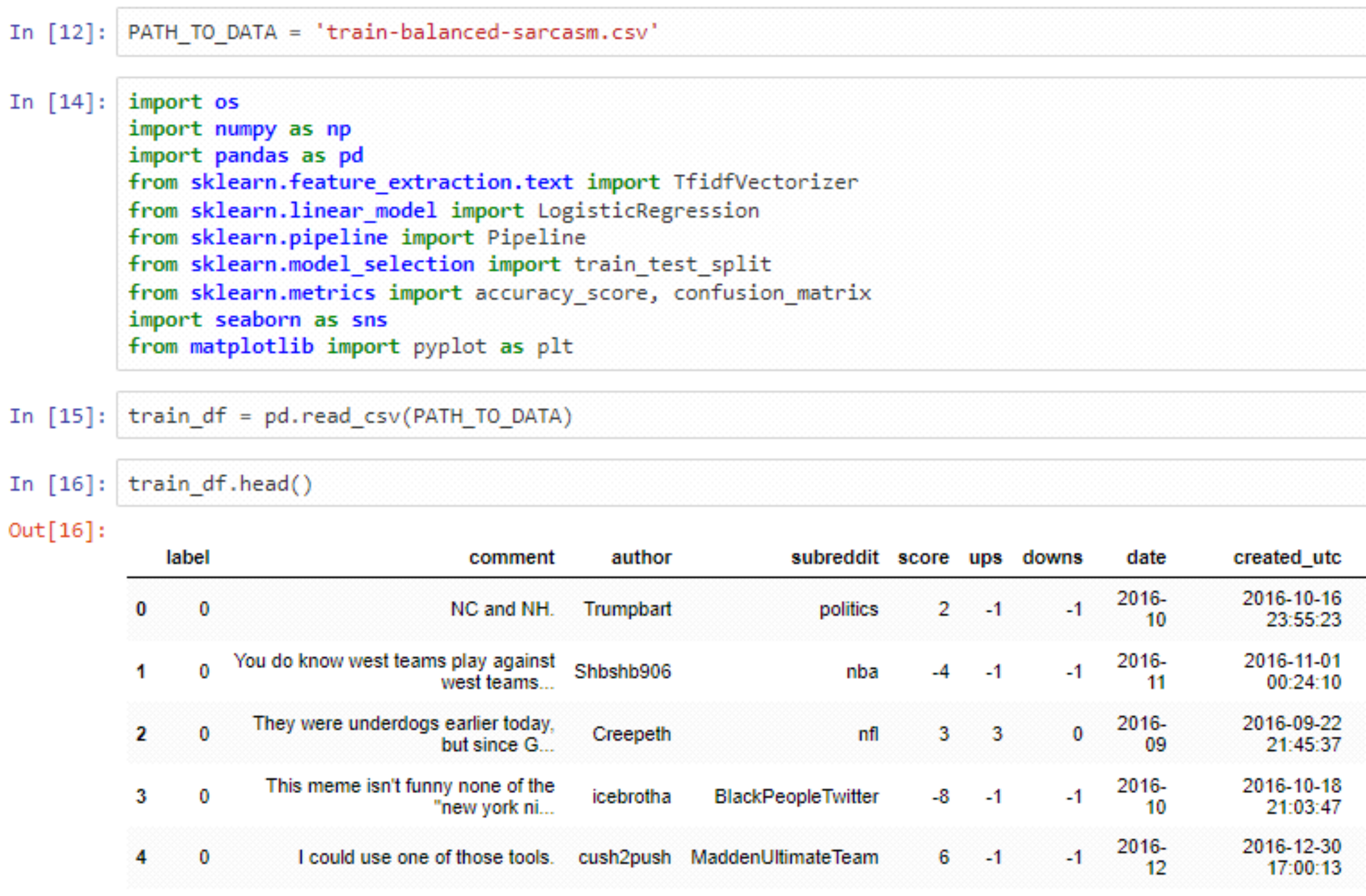
b) Dataset

c) An IDE

3.2 Start

First of all we got basic knowledge of python and wrote a code.

Importing the required header files



```
In [12]: PATH_TO_DATA = 'train-balanced-sarcasm.csv'
```

```
In [14]: import os
         import numpy as np
         import pandas as pd
         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn.linear_model import LogisticRegression
         from sklearn.pipeline import Pipeline
         from sklearn.model_selection import train_test_split
         from sklearn.metrics import accuracy_score, confusion_matrix
         import seaborn as sns
         from matplotlib import pyplot as plt
```

```
In [15]: train_df = pd.read_csv(PATH_TO_DATA)
```

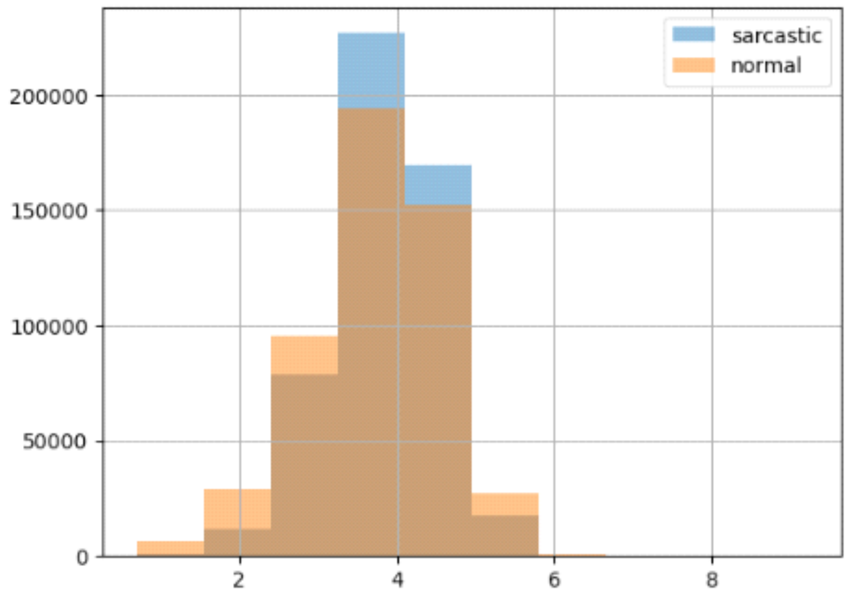
```
In [16]: train_df.head()
```

Out[16]:

	label	comment	author	subreddit	score	ups	downs	date	created_utc
0	0	NC and NH.	Trumpbart	politics	2	-1	-1	2016-10	2016-10-16 23:55:23
1	0	You do know west teams play against west teams...	Shbshb906	nba	-4	-1	-1	2016-11	2016-11-01 00:24:10
2	0	They were underdogs earlier today, but since G...	Creepeth	nfl	3	3	0	2016-09	2016-09-22 21:45:37
3	0	This meme isn't funny none of the "new york ni...	icebrotha	BlackPeopleTwitter	-8	-1	-1	2016-10	2016-10-18 21:03:47
4	0	I could use one of those tools.	cush2push	MaddenUltimateTeam	6	-1	-1	2016-12	2016-12-30 17:00:13

3.3 Code

```
In [19]: train_df.dropna(subset = ['comment'], inplace = True)

In [24]: train_df['label'].value_counts()

Out[24]: label
         0    505403
         1    505368
         Name: count, dtype: int64

In [26]: train_texts, valid_texts, y_train, y_valid = \
                 train_test_split(train_df['comment'], train_df['label'], random_state = 17)

In [28]: train_df.loc[train_df['label'] == 1, 'comment'].str.len().apply(np.log1p).hist(label = 'sarcastic', alpha = .5)
         train_df.loc[train_df['label'] == 0, 'comment'].str.len().apply(np.log1p).hist(label = 'normal', alpha = .5)
         plt.legend();
```



In [50]:

```
%%time
tfidf_logit_pipeline.fit(train_texts, y_train)
```

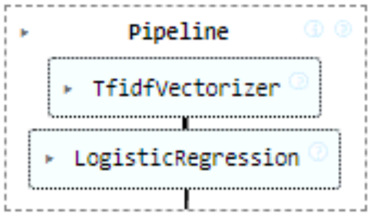
```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =        50001     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  6.93147D-01    |proj g|=  1.75475D-03
```

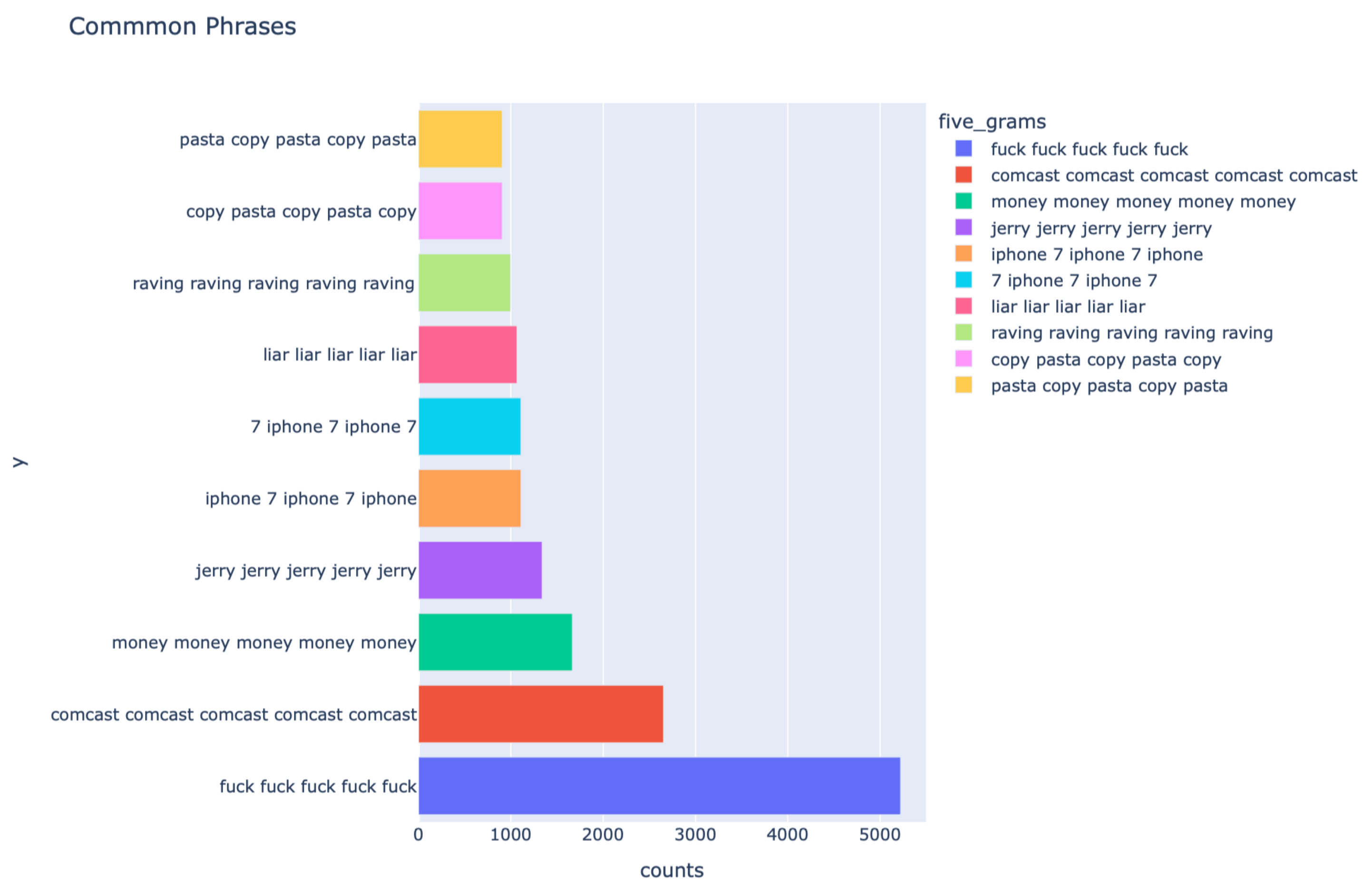
```
 This problem is unconstrained.
```

```
At iterate   50    f=  5.46377D-01    |proj g|=  4.22580D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
50001     87    104      1     0     0   7.497D-05   5.371D-01
  F =  0.53711600554808547

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
CPU times: user 15.1 s, sys: 499 ms, total: 15.6 s
Wall time: 25.1 s
```

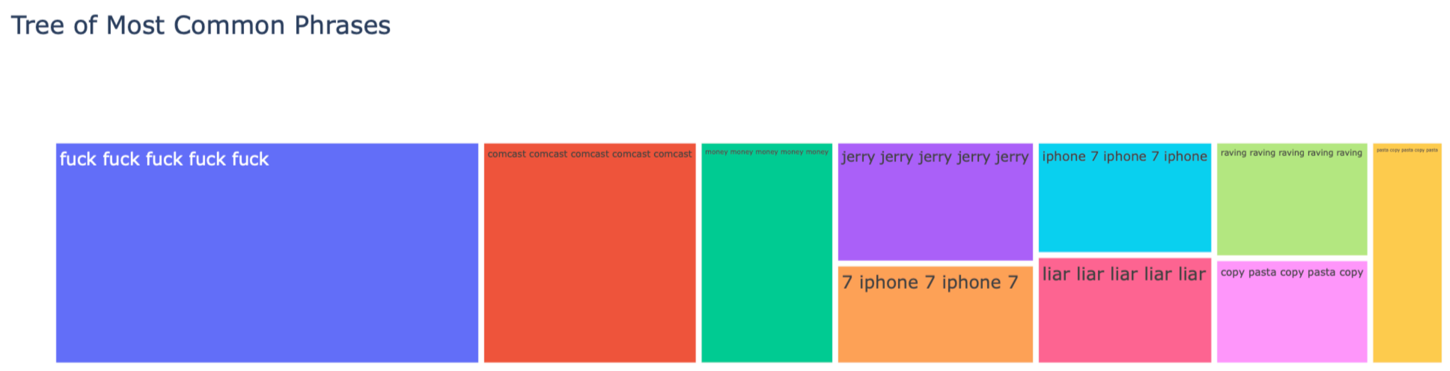
Out[50]:



```
[683]: import plotly.express as px
       fig = px.bar(df , x='counts', y=five_grams, title='Commmon Phrases', orientation = 'h',
                    width=1000, height=700,color='five_grams')
       fig.show()
```

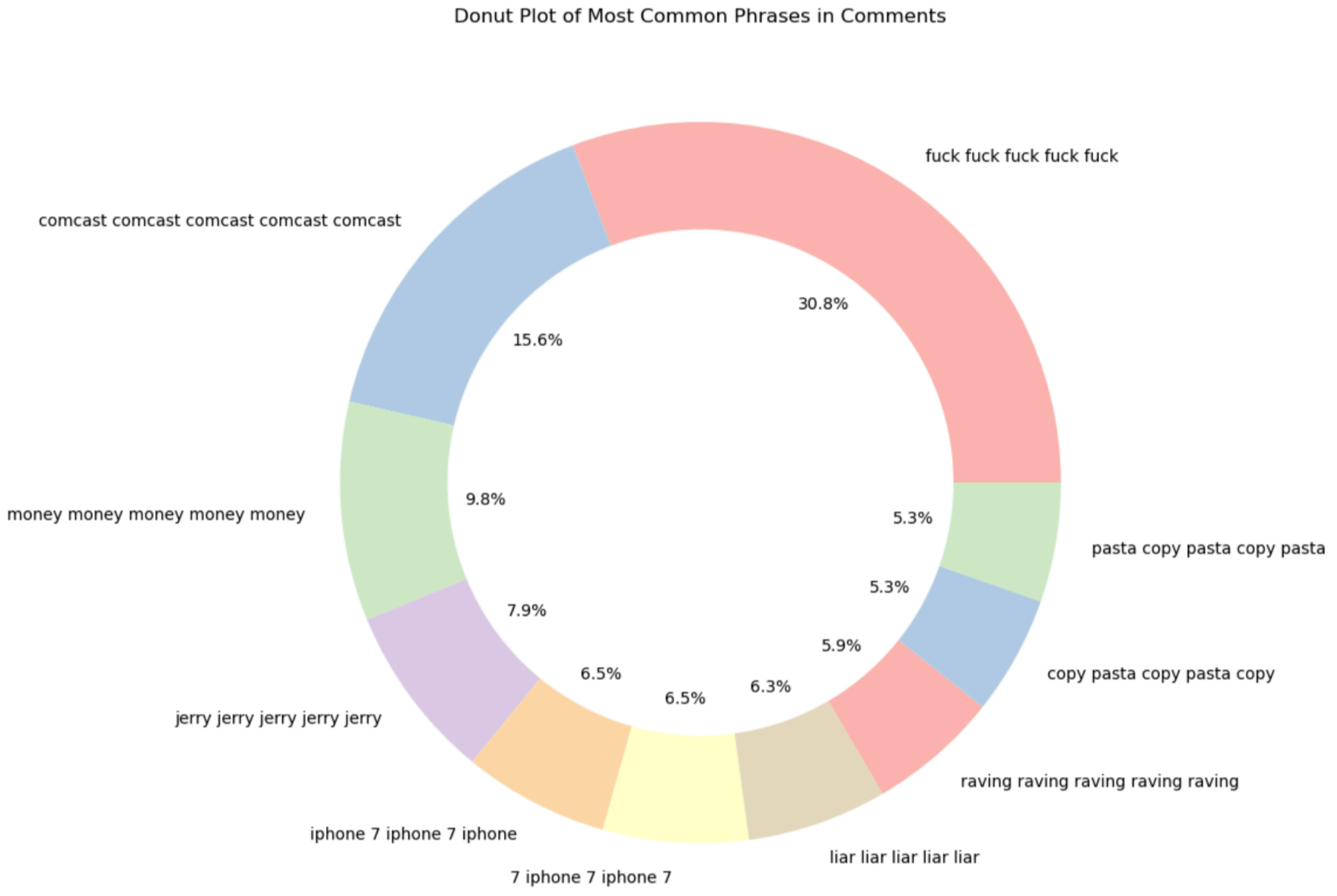


```
[685]: fig = px.treemap(df, path=['five_grams'], values='counts',title='Tree of Most Common Phrases')
       fig.show()
```



[687]:

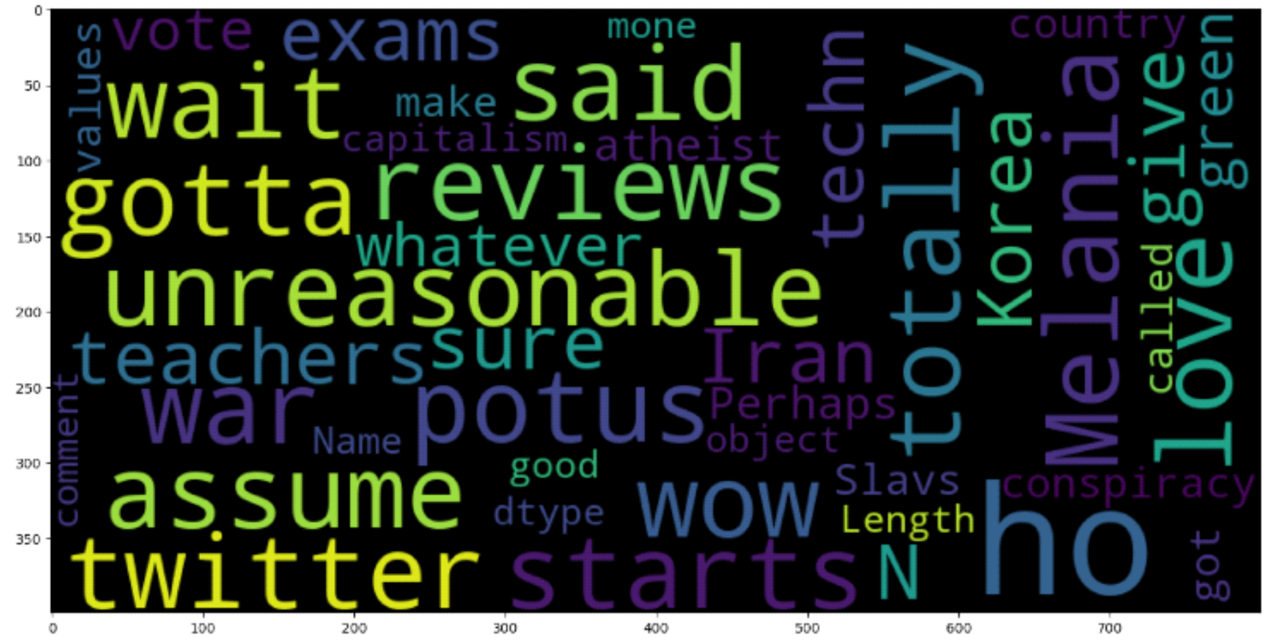
```
plt.figure(figsize=(16,10))
my_circle = plt.Circle((0, 0), 0.7, color='white')
plt.pie(counts, labels=five_grams, colors=Pastel1_7.hex_colors[:len(five_grams)], autopct='%1.1f%%')
p = plt.gcf()
p.gca().add_artist(my_circle)
plt.title('Donut Plot of Most Common Phrases in Comments')
plt.show()
```



```
In [30]: from wordcloud import WordCloud, STOPWORDS
```

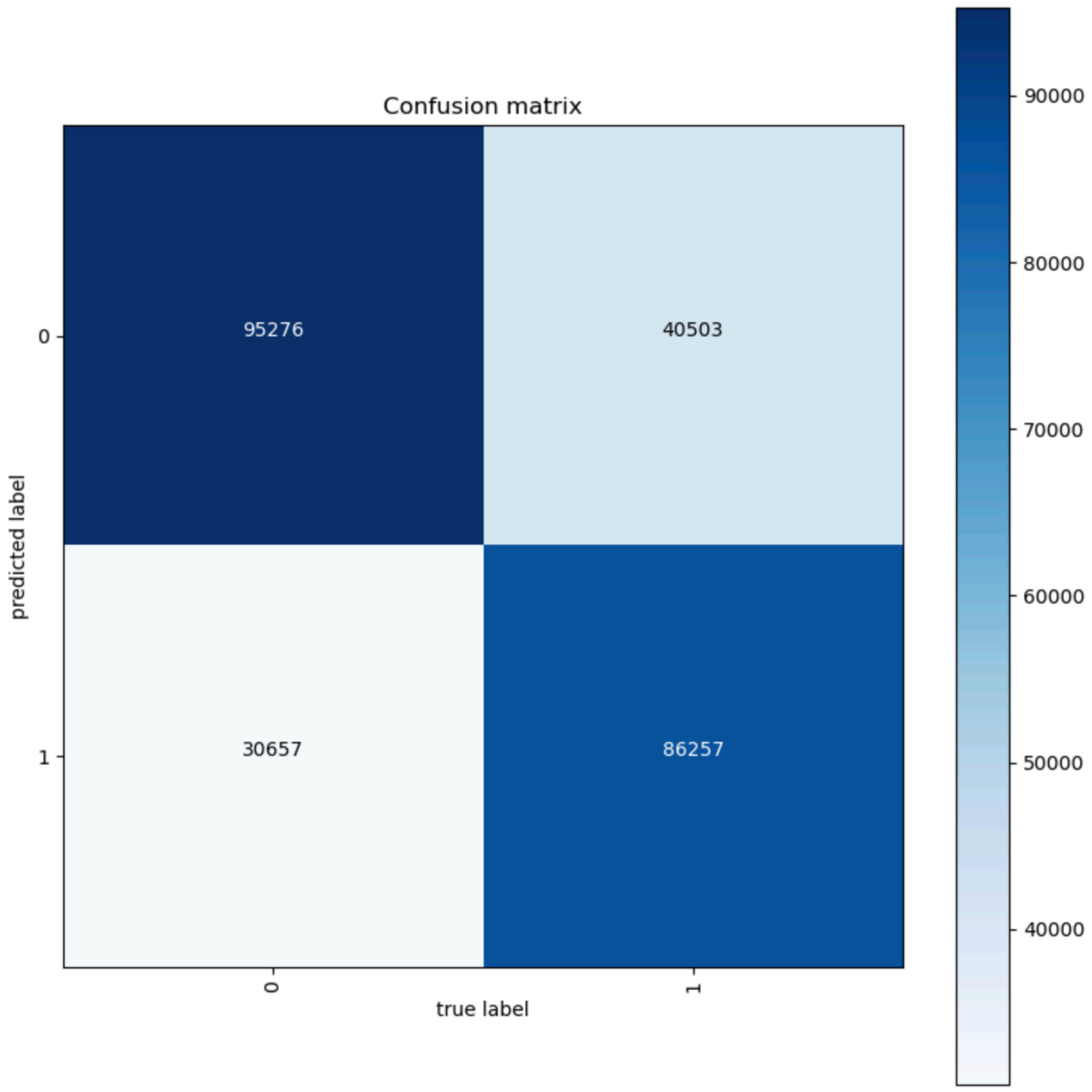
```
In [32]: wordcloud = WordCloud(
             background_color = 'black',
             stopwords = STOPWORDS,
             max_words = 200,
             max_font_size = 100,
             random_state = 17,
             width = 800,
             height = 400)
```

```
In [34]: plt.figure(figsize = (16, 12))
         wordcloud.generate(str(train_df.loc[train_df['label'] == 1, 'comment']))
         plt.imshow(wordcloud);
```



Chapter 4

Result and Discussion



The accuracy using the model after training is around 71%.

Chapter 5

Conclusion and Future Work

5.1 Conclusion :

a) Successfully implemented sarcasm detection using TF-IDF vectorization and logistic regression.

b) Achieved reasonable accuracy with interpretable and meaningful feature importance.

c) Feature engineering, such as bigrams and metadata, enhanced model performance.

d) Challenges persist in detecting context-dependent and nuanced sarcasm.

5.2 Future Work :

The future models will be able to distinguish sarcastic photographs, emojis and audio notes.

References

[1] Natural Language Processing with Python - **Edward Loper, Ewan Klein, and Steven Bird**

[2] Deep Learning for Natural Language Processing by Palash Goyal, Sumit Pandey, and Karan Jain

[3] Speech and Language Processing by Daniel Jurafsky and James H. Martin

[4] Dataset -https://www.kaggle.com/code/kashnitsky/a4-demo-sarcasm-detection-with-logit-solution

[5]Dataset-https://www.kaggle.com/code/shailaja4247/sentiment-analysis-of-tweets-wordclouds-textblob

[6] GFG guided project–https://www.geeksforgeeks.org/sarcasm-detection-using-neural-networks/

[7] NLP-Tutorials -https://github.com/laxmimerit/NLP-Tutorials-with-HuggingFace

[8] "A Survey on Sarcasm Detection" by Shmuel E. Shapira, Yona L. H. Koren

[9] "Sarcasm Detection in Twitter: The Role of Contextual Information" by Soujanya Poria, Erik Cambria, and Alessandro Gelbukh

[10] Towards Data Science - Sarcasm Detection (https://towardsdatascience.com)