Week 8: Project Deliverables

Project Topic: Hate Speech Detection using Transformers (Deep Learning)

Group Name: Data Defenders

Batch Code: LISUM34

Team Members' Details:

Name	Malika Hafiza Pasha	Biswadip Bhattacharyya	Oluseun Omobulejo
Email	malikahafizap@gmail.com	bbhattac@syr.edu	Seun.omobulejo@outlook.com
Country	United States	United States	United Kingdom
College/Company	California State University – Dominguez Hills	Syracuse University	University of Westminster
Specialization	Data Science		

Problem Description:

Any verbal, written, or behavioral communication that targets or uses derogatory or discriminatory language against an individual or group on the basis of who they are—that is, based on their religion, ethnicity, nationality, race, color, ancestry, sex, or another identity factor—is referred to as hate speech. We will walk you through a Python and machine learning hate speech detection model in this problem.

Sentiment categorization is often the process involved in hate speech detection. Therefore, training a model on data that is often used to classify attitudes can result in a model that can identify hate speech from a given text passage. Therefore, in order to complete the objective of developing a hate speech recognition model, we will use Twitter to find tweets that include hate speech.

Business Understanding:

The rise of social media and online communication platforms has increased the dissemination of hate speech. It is crucial for businesses, social media platforms, and communities to identify and address hate speech to

maintain a safe and inclusive environment. Effective detection and moderation can enhance user experience, comply with regulations, and protect brand reputation.

Data Collection:

Dataset Details:	
1. Dataset Name	train_E6oV3lV
2. Dataset storage location	Twitter hate speech (kaggle.com)
3. Base format of the file	csv
4. Size of the data	2.95 MB
5. Total number of observations	31962
6. Total number of files	1
7. Total number of features	3
8. Proposed Approach	There are no missing vales in this dataset

Data Information and Data Preprocessing

A. Data Information

1. **Importing Libraries**:

The necessary libraries for data preprocessing are imported. This typically includes pandas, numpy, re (regular expressions), and string libraries.

2. Loading the Dataset:

The dataset is loaded into a pandas DataFrame. The specific dataset and its location are not specified in the extracted content, but typically, it would involve using pd.read_csv or similar functions.

3. Initial Data Inspection:

Basic data inspection techniques are used, such as displaying the first few rows of the dataset using df.head() and getting the summary of the dataset with df.info().

In [5]:
```python
# total number of observations and features
print(f'Number of Observations: {df.shape[0]}')
print(f'Number of Features: {df.shape[1]}')
```

```
Number of Observations: 31962
Number of Features: 3
```

In [6]:
```python
# features that exists in this data
df.columns
```

Out[6]:
```
Index(['id', 'label', 'tweet'], dtype='object')
```

In [7]:
```python
# type of data in the dataset
df.dtypes
```

Out[7]:
```
id         int64
label      int64
tweet     object
dtype: object
```

In [8]:
```python
# information about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      31962 non-null  int64
 1   label   31962 non-null  int64
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

In [9]:
```python
# size of the data
df.size
```

Out[9]:
```
95886
```

In [10]:
```python
# Predictor Attribute
text = df.iloc[:, 1:]
text.tail()
```

Out[10]:

	label	tweet
31957	0	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð□□□ð...
31958	0	to see nina turner on the airwaves trying to...
31959	0	listening to sad songs on a monday morning otw...
31960	1	@user #sikh #temple vandalised in in #calgary,...
31961	0	thank you @user for you follow

In [11]:
```python
# target Attribute
label = df.iloc[:, 0:1]
label.tail()
```

Out[11]:

	id
31957	31958
31958	31959
31959	31960
31960	31961
31961	31962

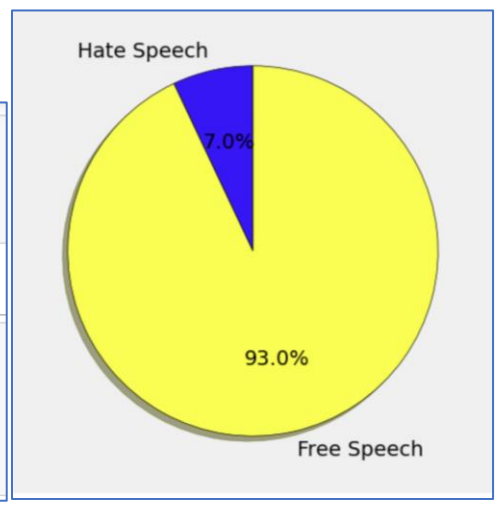
In [12]:
```python
# checking the missing values
df.isnull().sum()
```

Out[12]:
```
id       0
label    0
tweet    0
dtype: int64
```

In [13]:
```python
# Extract the Label Feature for Each Class
hate_speech = df[df['label'] == 1].shape[0]
free_speech = df[df['label'] == 0].shape[0]
print('Hate Speech =', hate_speech)
print('Free Speech =', free_speech)
```

```
Hate Speech = 2242
Free Speech = 29720
```

In [14]:
```python
# Visualize the Label Class
speech = [hate_speech, free_speech]
label = ["Hate Speech", "Free Speech"]
plt.pie(speech, labels = label, shadow = True, wedgeprops = {'edgecolor': 'black'},
        autopct = '%1.1f%%', startangle= 90, colors=['blue', 'yellow'])
plt.tight_layout()
plt.show()
```



2. Data Preprocessing - Text Cleaning:

a. Removing Punctuation: This function removes all punctuation from the text data.

```
# Text Cleaning: Removing Punctuation
def remove_punct(text):
    return text.translate(str.maketrans('','',string.punctuation))
df['tweet'] = df['tweet'].apply(remove_punct)
```

b. Removing URLs: This function removes URLs from the text data.

```
# Text Cleaning: Removing URLs
def remove_punct(text):
    return text.translate(str.maketrans('','',string.punctuation))
df['tweet'] = df['tweet'].apply(remove_punct)
```

c. Removing Tags: This function removes tags (e.g., @username) from the text data.

```
# Text Cleaning: Removing Tags
def remove_tag(text):
    newtext= re.sub(r'(@[A-Za-z0-9]+)',"",text)
    return newtext
df['tweet'] = df['tweet'].apply(remove_tag)
```

d. Removing Special Characters: This function removes special characters from the text data, keeping only alphanumeric characters.

```
# Text Cleaning: Removing Special Characters
def remove_special(text):
    return " ".join(e for e in text.split() if e.isalnum())
df['tweet'] = df['tweet'].apply(remove_special)
```

e. Tokenization: Involves splitting text into individual words or tokens.

```python
# tokenizing
from nltk.tokenize import sent_tokenize, word_tokenize
def tokenize(text):
    text = word_tokenize(text)
    return text
df['tweet'] = df['tweet'].apply(tokenize)
```

f. Removing Stopwords: It is usually removed to focus on the meaningful words in the text.

```python
# removing stopwords
from nltk.corpus import stopwords
def remove_stop(text):
    text  = [i for i in text if not i in stopwords.words('english')]
    return text
df['tweet'] = df['tweet'].apply(remove_stop)
```

g. Lemmatization: It reduce words to their root forms. Lemmatization considers the context and converts words to their base forms.

```python
# Lemmatization
from nltk.stem import WordNetLemmatizer
def Lemmatize(text):
    word_lem = WordNetLemmatizer()
    text = [word_lem.lemmatize(token) for token in text]
    return text
df['tweet'] = df['tweet'].apply(Lemmatize)
```

GitHub Repository Link:

https://github.com/malikahafizap/Data_Glacier_Internship/tree/main/Week%208%20(19%20July%20%E2%80%93%2025%20July%202024)