Group Name: Violetgo
Name: Xinyu Wang
Email: xw3080@nyu.edu
Country: United States
College: New York University
Specialization: NLP

Problem description: This project involves constructing a sophisticated classifier utilizing Natural Language Processing (NLP) techniques. Its primary function is to scrutinize and identify offensive or provocative comments on Twitter, specifically those that contain hate speech or abusive language, with the ultimate aim of fostering healthier online interactions.

Data understanding:

```
df_train.head()
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation

```
[5] df_test.head()
```

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...

As can be seen, we have the data including both training set(31962*3) and test set(17197*2). The training set(1/3 of the whole data) has the id of each tweet, the label marking whether the tweet belongs to hate speech(0 means normal comment, 1 means hate speech), and the tweet is the content of each comment. Meanwhile the test only has the id and the tweets.

Type of data:

```
df_train.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
```

```
[ ] df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17197 entries, 0 to 17196
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      17197 non-null  int64 
 1   tweet   17197 non-null  object
dtypes: int64(1), object(1)
memory usage: 268.8+ KB
```

As for the types of the data of the training set, it has 31962 rows and 3 columns. The three columns are:

1.'id': This column consists of integer data, likely serving as unique identifiers for each entry in the dataset.

2.'label': This column is also comprised of integer data. Given the context of your project, this might be the target variable indicating whether a tweet is classified as hate speech (possibly indicated by 1) or not (possibly indicated by 0).

3.'tweet': This is an object datatype, which typically refers to text data in pandas. This column likely contains the actual content of the tweets that you are going to analyze.

As for the test set, it has 17197 rows and 2 columns (i.e. id and tweet). The data type is the same as the training set.

Problems in the data: The training and testing datasets are both complete and unique, devoid of any missing or duplicate entries. With regards to the content of the tweets, certain symbols such as "@" and "#" are present which are not considered as words in the traditional sense. These symbols, although part of Twitter's communication language, may require special handling during the text preprocessing stage of our analysis.

Approaches:

1. I first converted all text into lower case using the df.str.lower() function for consistency.
2. I removed numerical values from the tweets using df_train['tweet'].astype(str).str.replace(r"\d+", " ").
3. To get rid of punctuation marks, I used df_train['tweet'].apply(lambda tweet: re.sub(r, '', tweet)).
4. The df_train['tweet'].str.strip() function was utilized to eliminate leading and trailing white spaces.
5. I used df_train['tweet'].apply(lambda x: x.encode("ascii", errors="ignore").decode()) to remove non-ascii characters.
6. HTML characters were purged using df_train['tweet'].str.replace(r"<[^<>]*>", "", regex=True).
7. Text tokenization was performed via df_train['tweet'].apply(word_tokenize).
8. Stop words were discarded using df_train['tweet'].apply(lambda tokens: [token for token in tokens if token not in stop_words]).
9. To reduce words to their base form, I performed stemming with df_train['tweet'].apply(lambda tokens: [stemmer.stem(token) for token in tokens]).
10. Finally, the cleaned and processed tokens were reassembled into complete sentences with df_train['tweet'].str.join(" ").