Sentiment Analysis for Marketing

PHASE -2

DATE	10 October 2023
TEAM ID	Proj-212173-Team-1
PROJECT NAME	Sentiment analysis for Marketing

DATA PREPROCESSING:

Data preprocessing is a crucial step in sentiment analysis, as it helps clean and prepare the text data for analysis. It involves the cleaning and transformation of raw data into a format that is suitable for analysis or for training machine learning models. The goal of data preprocessing is to enhance the quality of the data, making it more reliable and easier to work with.

Here are the steps you can follow to perform data preprocessing for sentiment analysis:

1.DATA CLEANING:

This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.

- ✓ **Handling missing values:** Identifying and filling in or removing missing data points.
- ✓ **Outlier detection and treatment:** Identifying and handling data points that are significantly different from the majority of the data.
- ✓ **Noise reduction:** Reducing random variations and errors in the data.

2.DATA INTEGRATION:

This involves combining data from multiple sources to create a unified dataset. Data integration can be challenging as it requires handling data with different formats, structures, and semantics. Techniques such as record linkage and data fusion can be used for data integration.

3.DATA TRANSFORMATION:

This involves converting the data into a suitable format for analysis. Common techniques used in data transformation include normalization, standardization, and discretization. Normalization is used to scale the data to a common range, while standardization is used to transform the data to have zero mean and unit variance. Discretization is used to convert continuous data into discrete categories.

4.DATA REDUCTION:

This involves reducing the size of the dataset while preserving the important information. Data reduction can be achieved through techniques such as feature selection and feature extraction. Feature selection involves selecting a subset of relevant features from the dataset, while feature extraction involves transforming the data into a lower-dimensional space while preserving the important information.

TEXT PREPROCESSING:

Text preprocessing in sentiment analysis is a crucial step that involves cleaning and transforming textual data to prepare it for analysis.

The goal of text preprocessing is to improve the quality of the text data, reduce noise, and make it suitable for sentiment analysis tasks.

TECHNIQUES USED IN TEXT PREPROCESSING :

- **Lowercasing:** Converting all text to lowercase helps ensure that the analysis is not case-sensitive. This way, "good" and "Good" are treated as the same word.

- **Tokenization:** Tokenization is the process of splitting text into individual words or tokens. It breaks down sentences or paragraphs into a list of words or sub-phrases, making it easier to analyze.
- **Removing Punctuation**: Removing punctuation marks like commas, periods, and exclamation points can help reduce noise and improve the accuracy of sentiment analysis.

- **Removing Stop Words:** Stop words are common words like "the," "and," "is," etc., that often do not carry significant sentiment information. Removing them can reduce the dimensionality of the data and improve processing speed.

- **Stemming and Lemmatization**: Stemming and lemmatization are techniques for reducing words to their root forms. For example, "running," "ran," and "runner" might be reduced to "run." This helps to group similar words together and reduce dimensionality.

- **Handling Emoticons and Emoji:** Sentiment analysis should take into account emoticons and emoji as they convey sentiment. You may choose to map them to sentiment labels.

- **Removing HTML Tags:** If dealing with text from web sources, it's common to encounter HTML tags. Removing these tags is essential to ensure that the analysis focuses on the text content.

- **Handling URLs and User Mentions:** URLs and user mentions (e.g., @username) are often irrelevant for sentiment analysis and can be removed or replaced.

- **Spell Checking and Correction:** Correcting spelling errors can improve the accuracy of sentiment analysis by ensuring that words are correctly recognized.

1)IMPORT MODULES OR LIBRARIES:

CODE AND OUTPUT:

```
[ ]  #import the required libraries
     import numpy as np
     import pandas as pd
     import nltk
     import string
     import re
     !pip install demoji
     import demoji

     Requirement already satisfied: demoji in /usr/local/lib/python3.10/dist-packages (1.1.0)
```

EXPLANATION:

Numpy: A library which is used for numerical operations and working with arrays or matrices.

Pandas: It is used for data manipulation and analysis.

Matplotlib: It is a data visualization library for creating various types of plots and charts.

Nltk: It is a library for natural language processing tasks.

Re: The **re** module is used for working with regular expressions to match and manipulate text patterns.

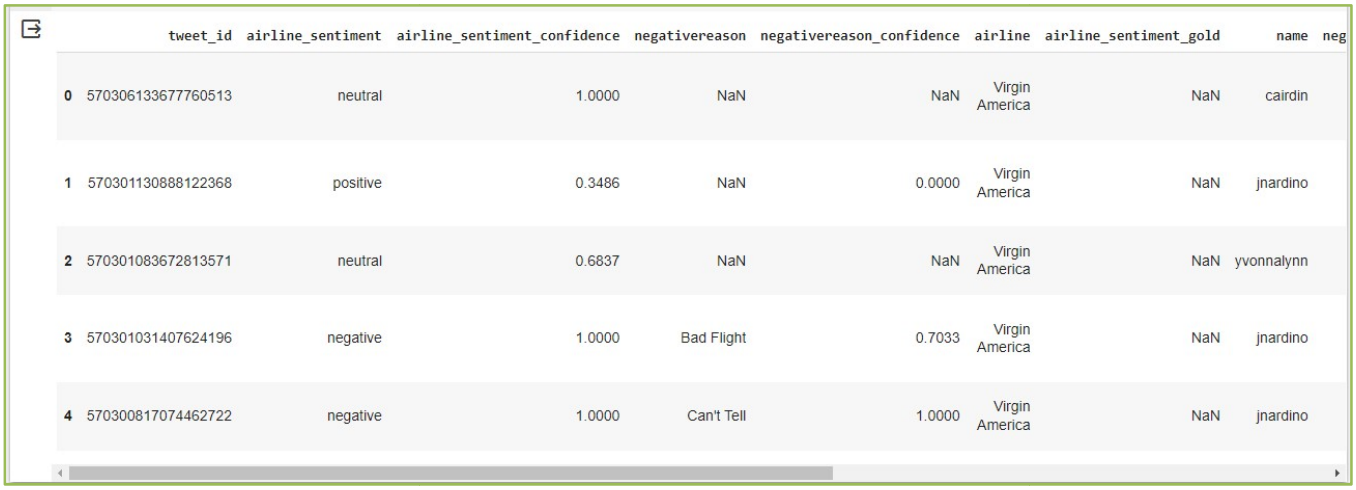
Demoji: It is a library for working with emojis in text data.

2) LOAD THE DATASET

CODE:

```
#Load the dataset
df=pd.read_csv('Tweets.csv')
#df.head() returns first five rows
df.head()
```

OUTPUT:



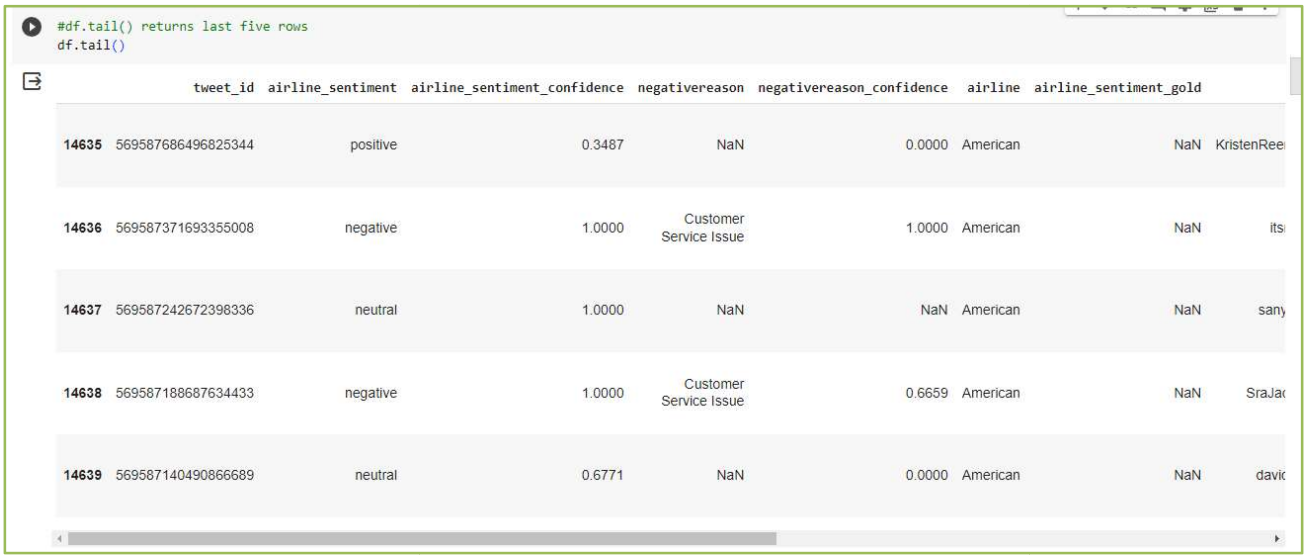
	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	neg
0	570306133677760513	neutral	1.0000	NaN	NaN	Virgin America	NaN	cairdin	
1	570301130888122368	positive	0.3486	NaN	0.0000	Virgin America	NaN	jnardino	
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	NaN	jnardino	
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	NaN	jnardino	

EXPLANATION:

pd.read_csv() - Used to read the data from csv file named “Tweets.csv” and stored it in a dataframe “df”.

df.head() – Display the first five rows of dataframe “df”

CODE AND OUTPUT:



```
#df.tail() returns last five rows
df.tail()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	
14635	569587686496825344	positive	0.3487	NaN	0.0000	American	NaN	KristenRee
14636	569587371693355008	negative	1.0000	Customer Service Issue	1.0000	American	NaN	its
14637	569587242672398336	neutral	1.0000	NaN	NaN	American	NaN	sany
14638	569587188687634433	negative	1.0000	Customer Service Issue	0.6659	American	NaN	SraJac
14639	569587140490866689	neutral	0.6771	NaN	0.0000	American	NaN	davic

EXPLANATION:

df.tail() – Display last five rows of dataframe “df”.

CODE AND OUTPUT:

```
[ ] #df.info() returns information about dataframe
    df.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 14640 entries, 0 to 14639
    Data columns (total 15 columns):
     #   Column                        Non-Null Count  Dtype
    ---  ------                        --------------  -----
     0   tweet_id                      14640 non-null  int64
     1   airline_sentiment             14640 non-null  object
     2   airline_sentiment_confidence  14640 non-null  float64
     3   negativereason                9178 non-null   object
     4   negativereason_confidence     10522 non-null  float64
     5   airline                       14640 non-null  object
     6   airline_sentiment_gold        40 non-null     object
     7   name                          14640 non-null  object
     8   negativereason_gold           32 non-null     object
     9   retweet_count                 14640 non-null  int64
     10  text                          14640 non-null  object
     11  tweet_coord                   1019 non-null   object
     12  tweet_created                 14640 non-null  object
     13  tweet_location                9907 non-null   object
     14  user_timezone                 9820 non-null   object
    dtypes: float64(2), int64(2), object(11)
    memory usage: 1.7+ MB
```

EXPLANATION:

df.info() –It gives information about the dataframe "df".

The typical output of `df.info()` includes:

- The total number of rows in the DataFrame.
- The total number of columns in the DataFrame.
- A list of column names.
- The count of non-null values in each column, which helps identify missing data.
- The data type of each column (e.g., integer, float, object).
- The approximate memory usage of the DataFrame.

CODE AND OUTPUT:

```
[ ] #df.isnull().sum() is used to count the missing values in each column of a dataframe
    df.isnull().sum()

    tweet_id                          0
    airline_sentiment                 0
    airline_sentiment_confidence      0
    negativereason                 5462
    negativereason_confidence      4118
    airline                           0
    airline_sentiment_gold        14600
    name                              0
    negativereason_gold           14608
    retweet_count                     0
    text                              0
    tweet_coord                   13621
    tweet_created                     0
    tweet_location                 4733
    user_timezone                  4820
    dtype: int64
```

EXPLANATION:

df.isnull().sum() - It counts the missing values in each column of a dataframe “df”.

df is a DataFrame containing your data.

df.isnull() creates a DataFrame of the same shape as `df`, where each element is `True` if the corresponding element in `df` is null (missing), and `False` otherwise.

.sum() is a method applied to the DataFrame resulting from `df.isnull()`. It sums the boolean values (`True` and `False`) along each column. Since `True` is treated as 1 and `False` as 0.

3) HANDLING MISSING VALUES:

CODE AND OUTPUT:

```
[ ] #df.fillna() is used to fill the missing values
    df['airline_sentiment_confidence'].fillna(df['airline_sentiment_confidence'].mean(), inplace=True)
    df['negativereason_confidence'].fillna(df['negativereason_confidence'].median(), inplace=True)
    df['negativereason'].fillna(df['negativereason'].mode(),inplace=True)
    df['user_timezone'].fillna(method='ffill', inplace=True)
    col=["negativereason_gold","airline_sentiment_gold","tweet_coord","tweet_location"]
    df.drop(col,axis=1,inplace=True)
    df['negativereason'].fillna('No text', inplace=True)
    #Recheck whether the dataframe has null values or not
    df.isnull().sum()

    tweet_id                      0
    airline_sentiment             0
    airline_sentiment_confidence  0
    negativereason                0
    negativereason_confidence     0
    airline                       0
    name                          0
    retweet_count                 0
    text                          0
    tweet_created                 0
    user_timezone                 0
    dtype: int64
```

EXPLANATION:

df.fillna() –is used to fill the missing columns in dataframe “df” with the means() or median() or mode().

df.fillna(method='ffill') carries the previous valid value forward to fill missing values.

df.fillna(df.mean()) fills missing values in each column with the mean of that column.

df.fillna(df.median()) fills missing values in each column with the median of that column.

df.fillna(df.mode()) fills missing values in each column with the mode of that column.

CODE AND OUTPUT:

```
[ ]  #Access the text column in a dataframe
     df['text']

     0                     @VirginAmerica What @dhepburn said.
     1        @VirginAmerica plus you've added commercials t...
     2        @VirginAmerica I didn't today... Must mean I n...
     3        @VirginAmerica it's really aggressive to blast...
     4        @VirginAmerica and it's a really big bad thing...
                                  ...
     14635    @AmericanAir thank you we got on a different f...
     14636    @AmericanAir leaving over 20 minutes Late Flig...
     14637    @AmericanAir Please bring American Airlines to...
     14638    @AmericanAir you have my money, you change my ...
     14639    @AmericanAir we have 8 ppl so we need 2 know h...
     Name: text, Length: 14640, dtype: object
```

EXPLANATION:

df[“text”] is used to access the column named ‘text’ from a dataframe “df”.

4) LOWERCASING:

CODE AND OUTPUT:

```
[ ]  #Text Preprocessing
     #Lowercasing the text
     df['new_text'] = df['text'].astype(str).str.lower()
     df['new_text']

     0                     @virginamerica what @dhepburn said.
     1        @virginamerica plus you've added commercials t...
     2        @virginamerica i didn't today... must mean i n...
     3        @virginamerica it's really aggressive to blast...
     4        @virginamerica and it's a really big bad thing...
                                  ...
     14635    @americanair thank you we got on a different f...
     14636    @americanair leaving over 20 minutes late flig...
     14637    @americanair please bring american airlines to...
     14638    @americanair you have my money, you change my ...
     14639    @americanair we have 8 ppl so we need 2 know h...
     Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

df[“text”].astype(str).str.lower() is used to convert the column ‘text’ in dataframe “df” into lowercase

5) REMOVING SPECIAL CHARACTERS AND HTML TAGS:

CODE AND OUTPUT:

```
[ ]
    def clean_txt(text):

        text=re.sub(r'@[a-zA-Z0-9]+','',text)#removes username
        text=re.sub(r'#\w+','',text)#removes hashtag
        text=re.sub(r'https?:/\/\s+','',text)#removes URL
        text=re.sub(r'RT[\s]+','',text)#removes retweet
        return text
    df['new_text']=df['new_text'].astype(str).apply(clean_txt)
    df['new_text']
```

```
0                                          what  said.
1         plus you've added commercials to the experien...
2         i didn't today... must mean i need to take an...
3         it's really aggressive to blast obnoxious "en...
4                 and it's a really big bad thing about it
                              ...
14635     thank you we got on a different flight to chi...
14636     leaving over 20 minutes late flight. no warni...
14637                   please bring american airlines to
14638     you have my money, you change my flight, and ...
14639     we have 8 ppl so we need 2 know how many seat...
Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

re.sub() is a method provided by the `re` module, which stands for regular expressions. It is used for performing regular expression-based substitutions in strings. This function allows you to find and replace patterns in text using regular expressions.

Syntax:

re.sub(pattern, replacement, text)

pattern: This is the regular expression pattern you want to search for in the text.

replacement: This is the string that you want to replace the matched pattern with.

text: This is the input text in which you want to perform the substitutions.

6) REMOVING PUNCTUATION:

CODE AND OUTPUT:

```
[ ]  #Removing Punctuation
     def remove_punctuation(text):
         return ''.join([char for char in text if char not in string.punctuation])

     df['new_text'] = df['new_text'].apply(remove_punctuation)
     df['new_text']
```

```
0                                          what  said
1         plus youve added commercials to the experienc...
2         i didnt today must mean i need to take anothe...
3         its really aggressive to blast obnoxious ente...
4                 and its a really big bad thing about it
                              ...
14635     thank you we got on a different flight to chi...
14636     leaving over 20 minutes late flight no warnin...
14637                   please bring american airlines to
14638     you have my money you change my flight and do...
14639     we have 8 ppl so we need 2 know how many seat...
Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

import string: This line imports the `string` module, which contains a string constant `string.punctuation` that holds all punctuation characters.

def remove_punctuation(text): This defines a function called `remove_punctuation` that takes a `text` input as a parameter.

return ''.join([char for char in text if char not in string.punctuation]): This line is the core of the function. It uses a list comprehension to iterate through each character (`char`) in the input `text`. It checks if the character is not in the `string.punctuation` constant (i.e., it's not a punctuation character) and adds it to the list. Then, it uses `join` to concatenate the characters in the list back into a single string without punctuation.

df['new_text'] = df['new_text'].apply(remove_punctuation): This line applies the `remove_punctuation` function to each element in the 'new_text' column of the DataFrame `df` using the `apply` method. It then assigns the result to a new column in the DataFrame, also named 'new_text'.

df['new_text']: This line prints the contents of the 'new_text' column after the punctuation removal has been applied.

As a result, the 'new_text' column in your DataFrame will contain the original text with all punctuation characters removed, making it cleaner and more suitable for text analysis or processing.

7) TOKENIZATION:

CODE AND OUTPUT:

```
[ ] #Tokenization

    nltk.download('punkt')
    from nltk.tokenize import word_tokenize
    def tokenize_text(text):

        tokens = word_tokenize(text)
        return tokens

    df['new_text'] = df['new_text'].astype(str).apply(word_tokenize)

    df['new_text']

    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
    0                                       [what, said]
    1        [plus, youve, added, commercials, to, the, exp...
    2        [i, didnt, today, must, mean, i, need, to, tak...
    3        [its, really, aggressive, to, blast, obnoxious...
    4        [and, its, a, really, big, bad, thing, about, it]
                                  ...
    14635    [thank, you, we, got, on, a, different, flight...
    14636    [leaving, over, 20, minutes, late, flight, no,...
    14637              [please, bring, american, airlines, to]
    14638    [you, have, my, money, you, change, my, flight...
    14639    [we, have, 8, ppl, so, we, need, 2, know, how,...
    Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

from nltk.tokenize import word_tokenize: This line imports the `word_tokenize` function from the NLTK library. This function is used for tokenizing text into words.

def tokenize_text(text): This defines a function called `tokenize_text` that takes a `text` input as a parameter.

tokens = word_tokenize(text): Inside the function, it uses `word_tokenize` to tokenize the input `text` into a list of words or tokens, and stores the result in the `tokens` variable.

df['new_text'] = df['new_text'].astype(str).apply(word_tokenize): This line applies the `word_tokenize` function to each element in the 'new_text' column of the DataFrame `df`. It first converts each element to a string using `astype(str)` (assuming the elements may not all be strings initially), and then applies `word_tokenize` to tokenize each text in the column. The result is assigned back to the 'new_text' column.

df['new_text']: This line prints the contents of the 'new_text' column after tokenization.

8) REMOVING STOPWORDS:

CODE:

```
#Removing stopwords
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words=stopwords.words('english')

def remove_stopwords(text):
    words = nltk.word_tokenize(text)
    filtered_words = [word for word in words if word.lower() not in stopwords.words('english')]
    return ' '.join(filtered_words)
df['new_text'] = df['new_text'].astype(str).apply(remove_stopwords)
df['new_text']
```

OUTPUT:

```
[ ]

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
0                                  [ 'what ' , 'said ' ]
1        [ 'plus ' , 'youve ' , 'added ' , 'commercials...
2        [ ' ' , 'didnt ' , 'today ' , 'must ' , 'mean ...
3        [ 'its ' , 'really ' , 'aggressive ' , 'to ' ,...
4        [ 'and ' , 'its ' , ' ' , 'really ' , 'big ' ,...
                              ...
14635    [ 'thank ' , 'you ' , 'we ' , 'got ' , 'on ' ,...
14636    [ 'leaving ' , 'over ' , '20 ' , 'minutes ' , ...
14637    [ 'please ' , 'bring ' , 'american ' , 'airlin...
14638    [ 'you ' , 'have ' , 'my ' , 'money ' , 'you '...
14639    [ 'we ' , 'have ' , ' 8 ' , 'ppl ' , 'so ' , '...
Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

nltk.download('stopwords'): This line downloads the NLTK stopwords dataset for the English language. It's a one-time operation to fetch the stopwords data.

from nltk.corpus import stopwords: This line imports the `stopwords` corpus from NLTK, which contains a list of common English stopwords.

stop_words = stopwords.words('english'): This line retrieves the list of English stopwords from NLTK and stores it in the `stop_words` variable.

def remove_stopwords(text): This defines a function called `remove_stopwords` that takes a `text` input as a parameter.

words = nltk.word_tokenize(text): Inside the function, it tokenizes the input `text` into words using `nltk.word_tokenize` and stores the result in the `words` variable.

filtered_words = [word for word in words if word.lower() not in stopwords.words('english')]: This line creates a list called `filtered_words` by iterating through the `words` list and including only those words that are not in the list of English stopwords (ignoring case). This effectively removes stopwords from the text.

return ' '.join(filtered_words): Finally, the function joins the `filtered_words` list back into a single string, with words separated by spaces, and returns the result.

df['new_text'] = df['new_text'].astype(str).apply(remove_stopwords): This line applies the `remove_stopwords` function to each element in the 'new_text' column of the DataFrame `df`. It first converts each element to a string using `astype(str)` and then applies `remove_stopwords` to remove stopwords from each text in the column. The result is assigned back to the 'new_text' column.

9) LEMMATIZATION:

CODE:

```
[ ]  #Lemmatization

     nltk.download('wordnet')
     nltk.download('punkt')
     from nltk.stem import WordNetLemmatizer
     lemmatizer = WordNetLemmatizer()
     def lemmatize_text(text):
         words = nltk.word_tokenize(text)
         lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
         return ' '.join(lemmatized_words)
     df['new_text'] = df['new_text'].astype(str).apply(lemmatize_text)
     df['new_text']
```

OUTPUT:

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
0                                  [ 'what ' , 'said ' ]
1        [ 'plus ' , 'youve ' , 'added ' , 'commercials...
2        [ ' ' , 'didnt ' , 'today ' , 'must ' , 'mean ...
3        [ 'its ' , 'really ' , 'aggressive ' , 'to ' ,...
4        [ 'and ' , 'its ' , ' ' , 'really ' , 'big ' ,...
                               ...
14635    [ 'thank ' , 'you ' , 'we ' , 'got ' , 'on ' ,...
14636    [ 'leaving ' , 'over ' , '20 ' , 'minutes ' , ...
14637    [ 'please ' , 'bring ' , 'american ' , 'airlin...
14638    [ 'you ' , 'have ' , 'my ' , 'money ' , 'you '...
14639    [ 'we ' , 'have ' , ' 8 ' , 'ppl ' , 'so ' , '...
Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

nltk.download('wordnet'): This line downloads the WordNet dataset for NLTK. WordNet is a lexical database that contains lemmas, synonyms, and other linguistic information.

from nltk.stem import WordNetLemmatizer: This line imports the `WordNetLemmatizer` class from NLTK, which is used for lemmatization.

lemmatizer = WordNetLemmatizer(): This line initializes a WordNet lemmatizer, creating an instance of the `WordNetLemmatizer` class.

def lemmatize_text(text): This defines a function called `lemmatize_text` that takes a `text` input as a parameter.

words = nltk.word_tokenize(text): Inside the function, it tokenizes the input `text` into words using `nltk.word_tokenize` and stores the result in the `words` variable.

lemmatized_words = [lemmatizer.lemmatize(word) for word in words]: This line creates a list called `lemmatized_words` by iterating through the `words` list and lemmatizing each word using the `lemmatizer.lemmatize` method.

return ' '.join(lemmatized_words): Finally, the function joins the `lemmatized_words` list back into a single string, with words separated by spaces, and returns the result.

df['new_text'] = df['new_text'].apply(lemmatize_text): This line applies the `lemmatize_text` function to each element in the 'new_text' column of the DataFrame `df`. It lemmatizes each text in the column and assigns the result back to the 'new_text' column.

10) REMOVING EMOJIS:

CODE AND OUTPUT:

```
#Removing emojis
demoji.download_codes()


def remove_emojis(text):
    return demoji.replace(text, '')
df['new_text'] = df['new_text'].apply(remove_emojis)
df['new_text']
```

```
<ipython-input-43-9336efb0d804>:2: FutureWarning: The demoji.download_codes attribute is deprecated and will be removed from demoji in a future version. It is an
  demoji.download_codes()
0                                [ 'what ' , 'said ' ]
1        [ 'plus ' , 'youve ' , 'added ' , 'commercials...
2        [ ' ' , 'didnt ' , 'today ' , 'must ' , 'mean ...
3        [ 'its ' , 'really ' , 'aggressive ' , 'to ' ,...
4        [ 'and ' , 'its ' , ' ' , 'really ' , 'big ' ,...
                               ...
14635    [ 'thank ' , 'you ' , 'we ' , 'got ' , 'on ' ,...
14636    [ 'leaving ' , 'over ' , '20 ' , 'minutes ' , ...
14637    [ 'please ' , 'bring ' , 'american ' , 'airlin...
14638    [ 'you ' , 'have ' , 'my ' , 'money ' , 'you '...
14639    [ 'we ' , 'have ' , ' 8 ' , 'ppl ' , 'so ' , '...
Name: new_text, Length: 14640, dtype: object
```

EXPLANATION:

demoji.download_codes(): This line downloads the emoji codes required by the `demoji` library. These codes are used to identify and replace emojis in text.

def remove_emojis(text): This defines a function called `remove_emojis` that takes a `text` input as a parameter.

return demoji.replace(text, ''): Inside the function, it uses the `demoji.replace` function to remove emojis from the input `text`. This function replaces emojis with an empty string, effectively removing them from the text.

df['new_text'] = df['new_text'].apply(remove_emojis): This line applies the `remove_emojis` function to each element in the 'new_text' column of the DataFrame `df`. It removes emojis from each text in the column and assigns the result back to the 'new_text' column.

11) TEXT LENGTH BASED OUTLIER DETECTION:

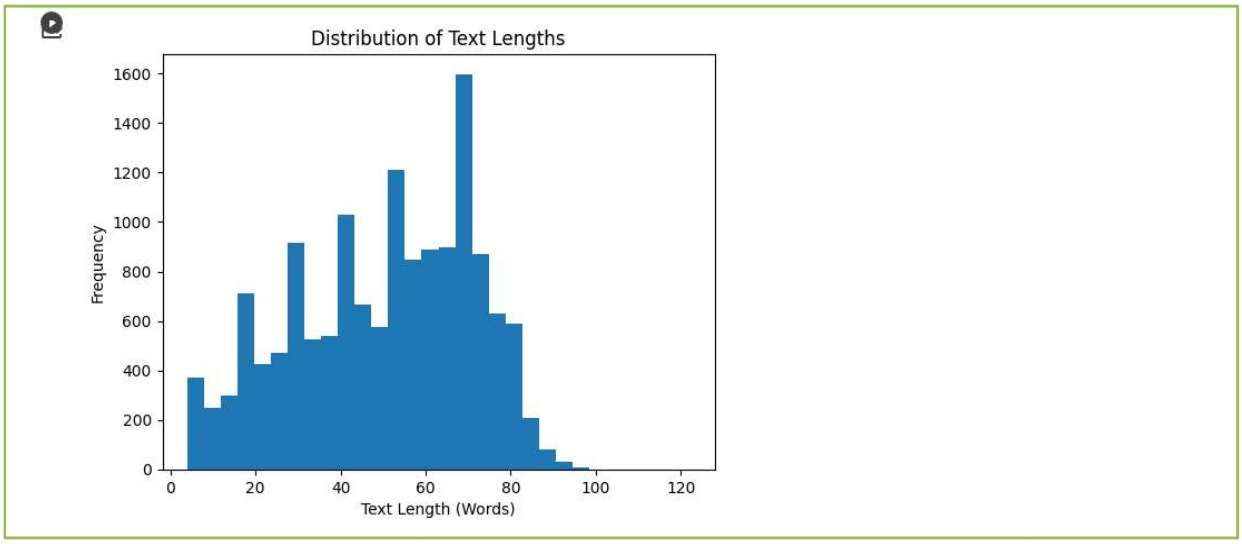
CODE:

```
[ ]  #Text length based outlier detection
     df['text_length_words'] = df['new_text'].apply(lambda x: len(x.split()))
```

```
[ ]  import matplotlib.pyplot as plt

     plt.hist(df['text_length_words'], bins=30)
     plt.xlabel('Text Length (Words)')
     plt.ylabel('Frequency')
     plt.title('Distribution of Text Lengths')
     plt.show()
```

OUTPUT:



EXPLANATION:

Import the pyplot from matplotlib library as plt.

Plt.hist(df[‘text_length_words’],bins=30) is used to create a histogram of data with 30 bins.

Plt.xlabel(“Text_lenght(words)”) – To set label for x-axis as “Text_lenght(words)”

Plt.ylabel(“Frequency)– To set label for y-axis as “Frequency”

Plt.title(“Distribution of text lengths”)–To set title for the plot as “Distribution of text lengths”

Plt.show()– To display plot with labeled x and y axis.

CODE AND OUTPUT:

```
threshold = 300
df['outlier_flag'] = False
df.loc[df['text_length_words'] > threshold, 'outlier_flag'] = True
df.head()
```

	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	name	retweet_count	tex
0	570306133677760513	neutral	1.0000	Customer Service Issue	0.6706	Virgin America	cairdin	0	@VirginAmeric Wh @dhepbu sa
1	570301130888122368	positive	0.3486	No text	0.0000	Virgin America	jnardino	0	@VirginAmeric plus you'v adde commercials t.
2	570301083672813571	neutral	0.6837	No text	0.6706	Virgin America	yvonnalynn	0	@VirginAmeric I didn't today. Must mean I n.
3	570301031407624196	negative	1.0000	Bad Flight	0.7033	Virgin America	jnardino	0	@VirginAmeric it's real aggressive blast.
4	570300817074462722	negative	1.0000	Can't Tell	1.0000	Virgin America	jnardino	0	@VirginAmeric and it's a real big bad thing.

EXPLANATION:

Set the threshold value as 300.

df['outlier_flag'] = False

It will add a new column "outlier_flag" to your DataFrame `df`, and each row in that column will have the value `False`.

df.loc[df['text_length_words'] > threshold, 'outlier_flag'] = True

.loc is used to locate rows in the DataFrame where a specific condition is met, and then we set the "outlier_flag" to `True` for those rows.