Theory Activity No. 1

Name: Vinay Bhure

PRN:202401070031

Roll no: ET2-12

Batch: ET-21

Dataset: Text Classification

Dataset Link: https://www.kaggle.com/datasets/vstepanenko/disaster-tweets

20 Problem statements:

1. Find the total number of tweets in the dataset.

2. Find the number of real disaster and non-disaster tweets.

3. Calculate the percentage of real disaster tweets using NumPy.

4. Calculate the percentage of non-disaster tweets using NumPy.

5. Find the number of missing values in keyword and location.

6. Replace missing keywords with "unknown" using NumPy.

7. Replace missing locations with "unknown" using NumPy.

8. Add a new column text_length representing the number of characters in the tweet.

9. Add a new column word_count representing the number of words in the tweet.

10. Calculate the average word count of disaster tweets using NumPy.

11. Calculate the average word count of non-disaster tweets using NumPy.

12. Find the most common keyword among real disaster tweets.

13. Find the most common keyword among non-disaster tweets.

14. Find the tweet with the maximum number of characters.

15. Find the tweet with the minimum number of characters.

16. Remove duplicate tweets and display the new shape of the dataset.

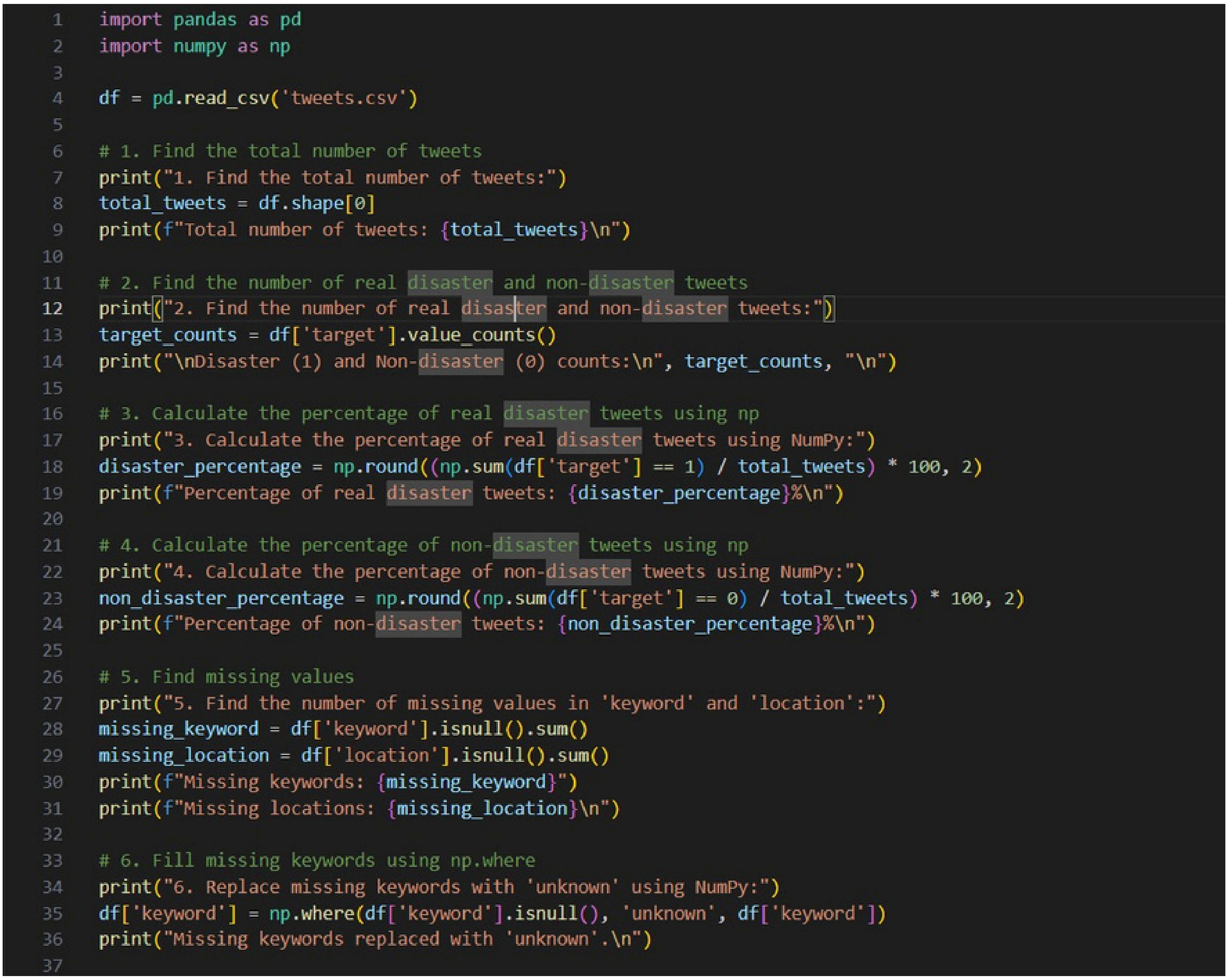
17. Calculate the correlation between text length and target using NumPy.

18. Group tweets by location and count disaster tweets.

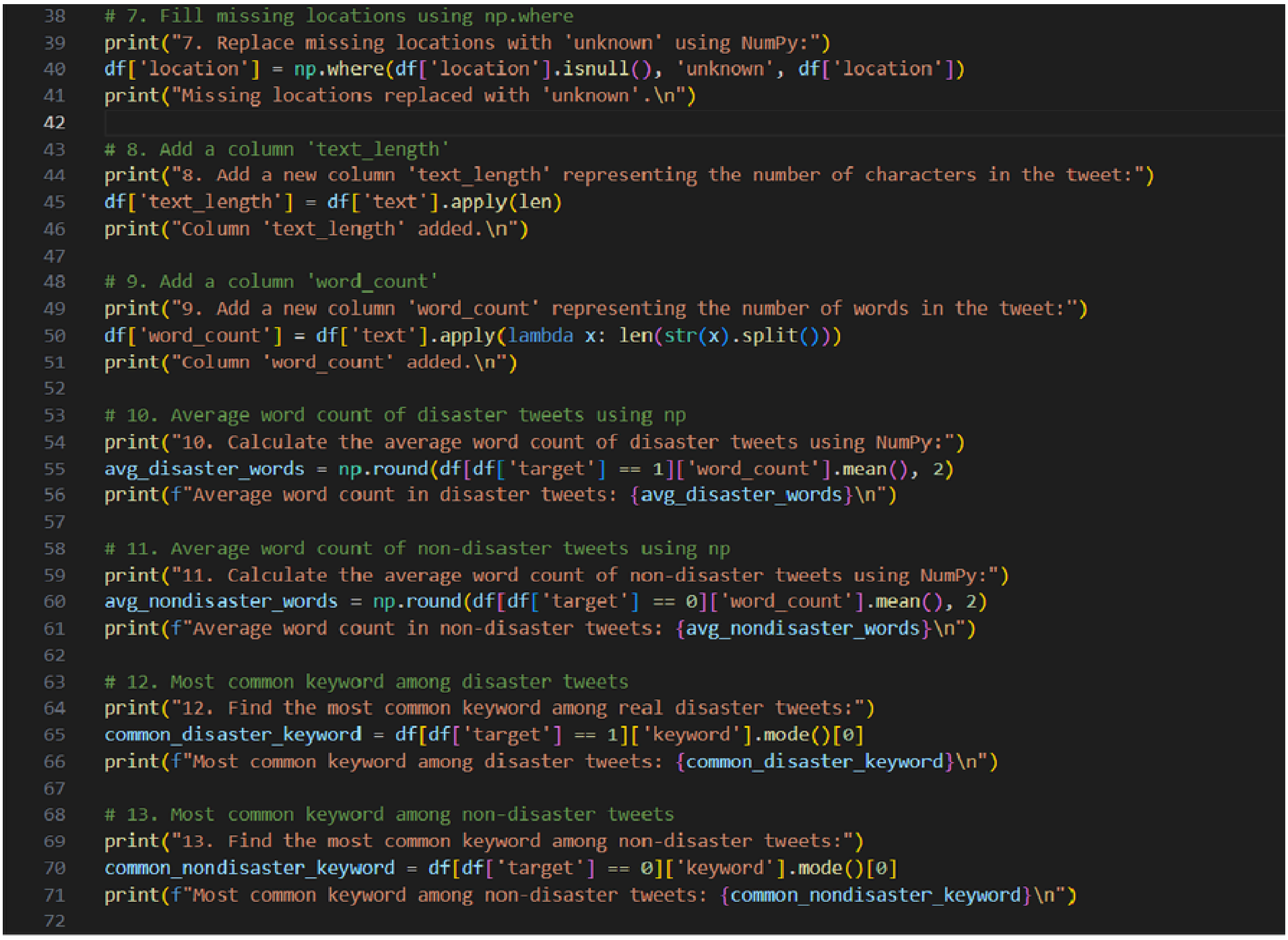
19. Find the top 5 locations with the highest number of disaster tweets.

20. Find how many disaster tweets contain the word "help" using Numpy and String Operations.

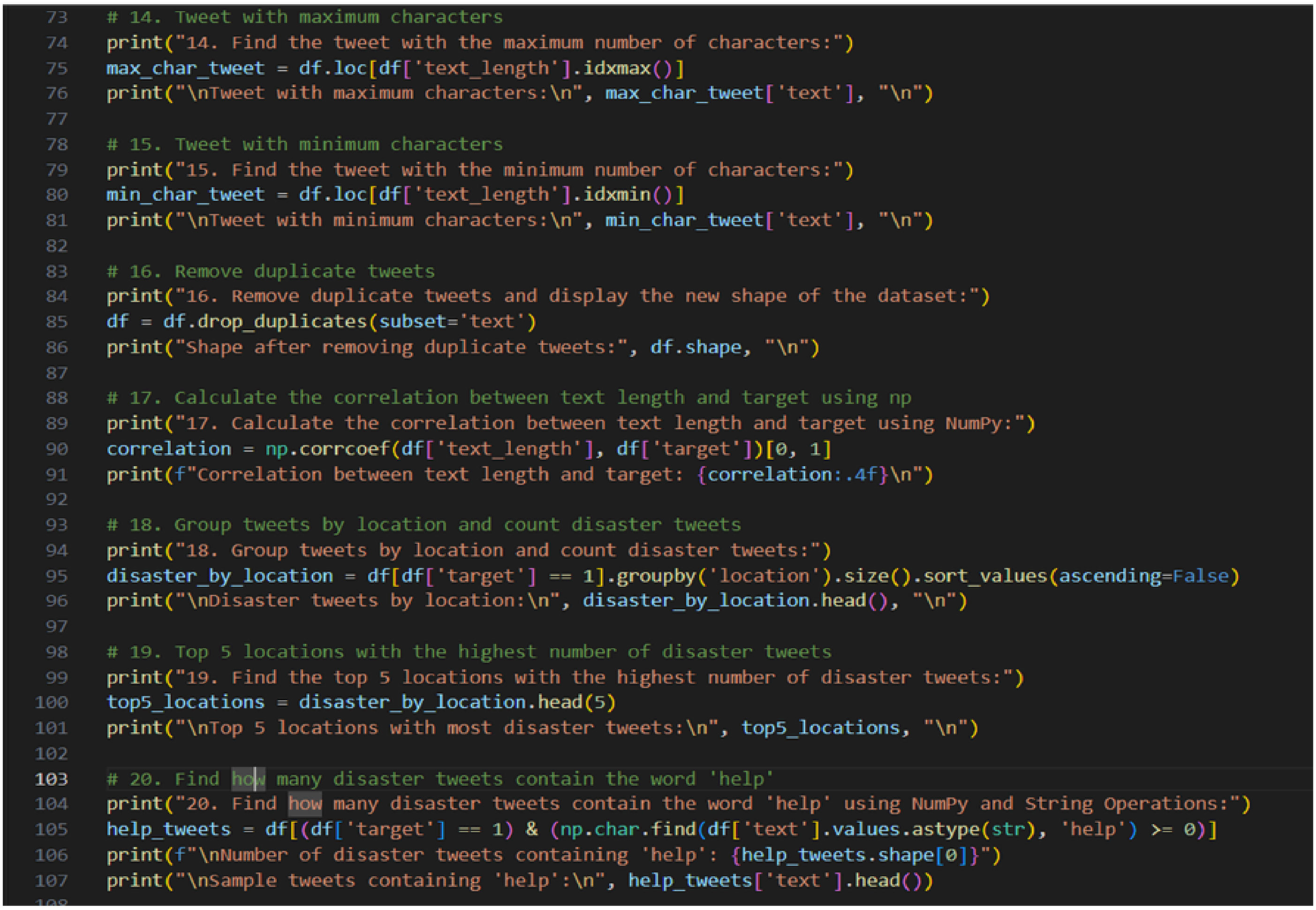
Code



```python
import pandas as pd
import numpy as np

df = pd.read_csv('tweets.csv')

# 1. Find the total number of tweets
print("1. Find the total number of tweets:")
total_tweets = df.shape[0]
print(f"Total number of tweets: {total_tweets}\n")

# 2. Find the number of real disaster and non-disaster tweets
print("2. Find the number of real disaster and non-disaster tweets:")
target_counts = df['target'].value_counts()
print("\nDisaster (1) and Non-disaster (0) counts:\n", target_counts, "\n")

# 3. Calculate the percentage of real disaster tweets using np
print("3. Calculate the percentage of real disaster tweets using NumPy:")
disaster_percentage = np.round((np.sum(df['target'] == 1) / total_tweets) * 100, 2)
print(f"Percentage of real disaster tweets: {disaster_percentage}%\n")

# 4. Calculate the percentage of non-disaster tweets using np
print("4. Calculate the percentage of non-disaster tweets using NumPy:")
non_disaster_percentage = np.round((np.sum(df['target'] == 0) / total_tweets) * 100, 2)
print(f"Percentage of non-disaster tweets: {non_disaster_percentage}%\n")

# 5. Find missing values
print("5. Find the number of missing values in 'keyword' and 'location':")
missing_keyword = df['keyword'].isnull().sum()
missing_location = df['location'].isnull().sum()
print(f"Missing keywords: {missing_keyword}")
print(f"Missing locations: {missing_location}\n")

# 6. Fill missing keywords using np.where
print("6. Replace missing keywords with 'unknown' using NumPy:")
df['keyword'] = np.where(df['keyword'].isnull(), 'unknown', df['keyword'])
print("Missing keywords replaced with 'unknown'.\n")

```

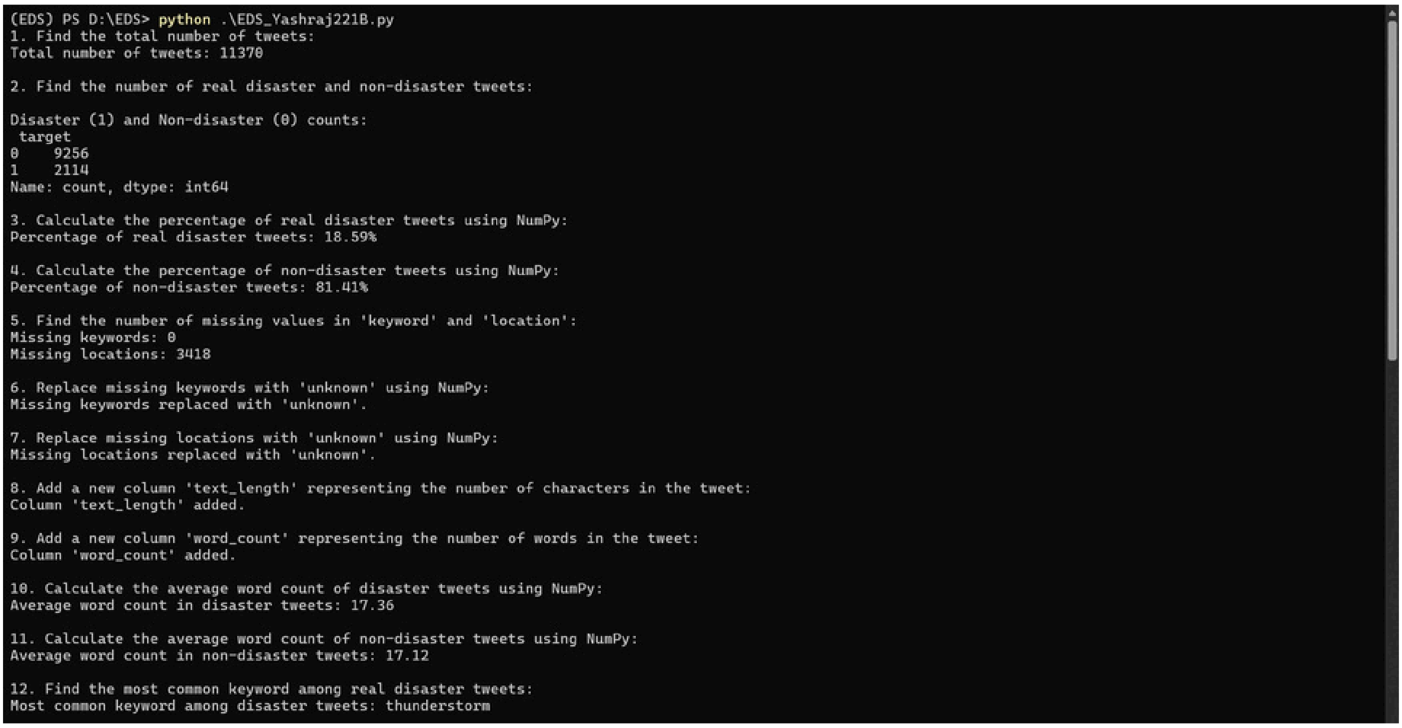


```python
# 7. Fill missing locations using np.where
print("7. Replace missing locations with 'unknown' using NumPy:")
df['location'] = np.where(df['location'].isnull(), 'unknown', df['location'])
print("Missing locations replaced with 'unknown'.\n")

# 8. Add a column 'text_length'
print("8. Add a new column 'text_length' representing the number of characters in the tweet:")
df['text_length'] = df['text'].apply(len)
print("Column 'text_length' added.\n")

# 9. Add a column 'word_count'
print("9. Add a new column 'word_count' representing the number of words in the tweet:")
df['word_count'] = df['text'].apply(lambda x: len(str(x).split()))
print("Column 'word_count' added.\n")

# 10. Average word count of disaster tweets using np
print("10. Calculate the average word count of disaster tweets using NumPy:")
avg_disaster_words = np.round(df[df['target'] == 1]['word_count'].mean(), 2)
print(f"Average word count in disaster tweets: {avg_disaster_words}\n")

# 11. Average word count of non-disaster tweets using np
print("11. Calculate the average word count of non-disaster tweets using NumPy:")
avg_nondisaster_words = np.round(df[df['target'] == 0]['word_count'].mean(), 2)
print(f"Average word count in non-disaster tweets: {avg_nondisaster_words}\n")

# 12. Most common keyword among disaster tweets
print("12. Find the most common keyword among real disaster tweets:")
common_disaster_keyword = df[df['target'] == 1]['keyword'].mode()[0]
print(f"Most common keyword among disaster tweets: {common_disaster_keyword}\n")

# 13. Most common keyword among non-disaster tweets
print("13. Find the most common keyword among non-disaster tweets:")
common_nondisaster_keyword = df[df['target'] == 0]['keyword'].mode()[0]
print(f"Most common keyword among non-disaster tweets: {common_nondisaster_keyword}\n")

```

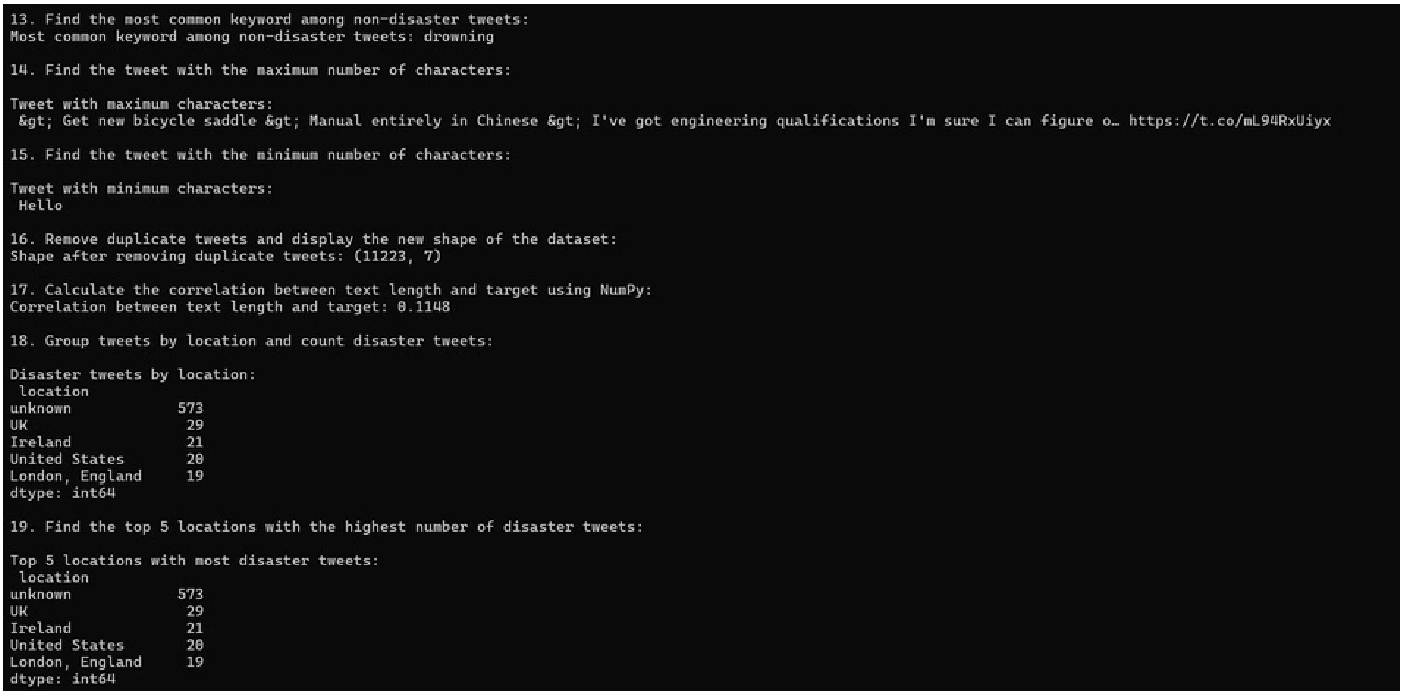


```python
# 14. Tweet with maximum characters
print("14. Find the tweet with the maximum number of characters:")
max_char_tweet = df.loc[df['text_length'].idxmax()]
print("\nTweet with maximum characters:\n", max_char_tweet['text'], "\n")

# 15. Tweet with minimum characters
print("15. Find the tweet with the minimum number of characters:")
min_char_tweet = df.loc[df['text_length'].idxmin()]
print("\nTweet with minimum characters:\n", min_char_tweet['text'], "\n")

# 16. Remove duplicate tweets
print("16. Remove duplicate tweets and display the new shape of the dataset:")
df = df.drop_duplicates(subset='text')
print("Shape after removing duplicate tweets:", df.shape, "\n")

# 17. Calculate the correlation between text length and target using np
print("17. Calculate the correlation between text length and target using NumPy:")
correlation = np.corrcoef(df['text_length'], df['target'])[0, 1]
print(f"Correlation between text length and target: {correlation:.4f}\n")

# 18. Group tweets by location and count disaster tweets
print("18. Group tweets by location and count disaster tweets:")
disaster_by_location = df[df['target'] == 1].groupby('location').size().sort_values(ascending=False)
print("\nDisaster tweets by location:\n", disaster_by_location.head(), "\n")

# 19. Top 5 locations with the highest number of disaster tweets
print("19. Find the top 5 locations with the highest number of disaster tweets:")
top5_locations = disaster_by_location.head(5)
print("\nTop 5 locations with most disaster tweets:\n", top5_locations, "\n")

# 20. Find how many disaster tweets contain the word 'help'
print("20. Find how many disaster tweets contain the word 'help' using NumPy and String Operations:")
help_tweets = df[(df['target'] == 1) & (np.char.find(df['text'].values.astype(str), 'help') >= 0)]
print(f"\nNumber of disaster tweets containing 'help': {help_tweets.shape[0]}")
print("\nSample tweets containing 'help':\n", help_tweets['text'].head())
```

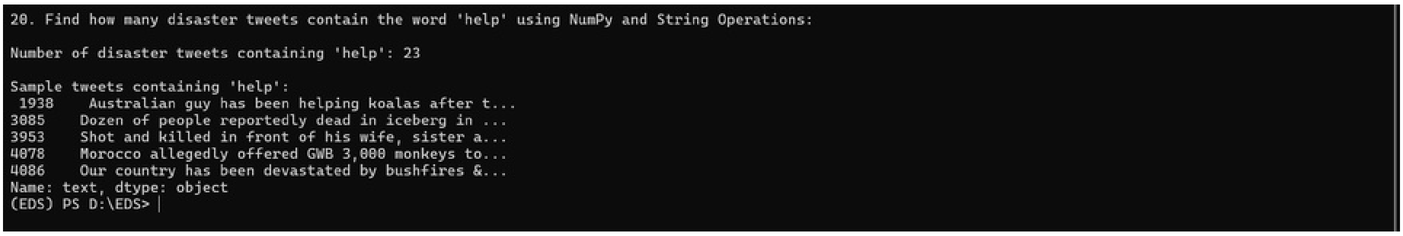
Output



```
(EDS) PS D:\EDS> python .\EDS_Yashraj221B.py
1. Find the total number of tweets:
Total number of tweets: 11370

2. Find the number of real disaster and non-disaster tweets:

Disaster (1) and Non-disaster (0) counts:
 target
0    9256
1    2114
Name: count, dtype: int64

3. Calculate the percentage of real disaster tweets using NumPy:
Percentage of real disaster tweets: 18.59%

4. Calculate the percentage of non-disaster tweets using NumPy:
Percentage of non-disaster tweets: 81.41%

5. Find the number of missing values in 'keyword' and 'location':
Missing keywords: 0
Missing locations: 3418

6. Replace missing keywords with 'unknown' using NumPy:
Missing keywords replaced with 'unknown'.

7. Replace missing locations with 'unknown' using NumPy:
Missing locations replaced with 'unknown'.

8. Add a new column 'text_length' representing the number of characters in the tweet:
Column 'text_length' added.

9. Add a new column 'word_count' representing the number of words in the tweet:
Column 'word_count' added.

10. Calculate the average word count of disaster tweets using NumPy:
Average word count in disaster tweets: 17.36

11. Calculate the average word count of non-disaster tweets using NumPy:
Average word count in non-disaster tweets: 17.12

12. Find the most common keyword among real disaster tweets:
Most common keyword among disaster tweets: thunderstorm
```



```
13. Find the most common keyword among non-disaster tweets:
Most common keyword among non-disaster tweets: drowning

14. Find the tweet with the maximum number of characters:

Tweet with maximum characters:
 &gt; Get new bicycle saddle &gt; Manual entirely in Chinese &gt; I've got engineering qualifications I'm sure I can figure o… https://t.co/mL94RxUiyx

15. Find the tweet with the minimum number of characters:

Tweet with minimum characters:
 Hello

16. Remove duplicate tweets and display the new shape of the dataset:
Shape after removing duplicate tweets: (11223, 7)

17. Calculate the correlation between text length and target using NumPy:
Correlation between text length and target: 0.1148

18. Group tweets by location and count disaster tweets:

Disaster tweets by location:
 location
unknown            573
UK                  29
Ireland             21
United States       20
London, England     19
dtype: int64

19. Find the top 5 locations with the highest number of disaster tweets:

Top 5 locations with most disaster tweets:
 location
unknown            573
UK                  29
Ireland             21
United States       20
London, England     19
dtype: int64
```



```
20. Find how many disaster tweets contain the word 'help' using NumPy and String Operations:

Number of disaster tweets containing 'help': 23

Sample tweets containing 'help':
 1938    Australian guy has been helping koalas after t...
3085    Dozen of people reportedly dead in iceberg in ...
3953    Shot and killed in front of his wife, sister a...
4078    Morocco allegedly offered GWB 3,000 monkeys to...
4086    Our country has been devastated by bushfires &...
Name: text, dtype: object
(EDS) PS D:\EDS> |
```