```
# Data manipulation and analysis
import numpy as np
import pandas as pd

# Visualization libraries
import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt
from matplotlib.ticker import MaxNLocator
import matplotlib.gridspec as gridspec
import matplotlib.patches as mpatches
%matplotlib inline

# Text processing and visualization
import re
from wordcloud import WordCloud
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Deep learning framework
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Embedding, LSTM, GRU, Bidirectional
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Machine learning utilities
from sklearn.model_selection import train_test_split

# Model training utilities
from keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint

# Additional utilities
from collections import Counter
import missingno as msno

# Suppress warnings
import warnings
warnings.filterwarnings("ignore")
```

```
# Load the training dataset from CSV file
training_data = pd.read_csv("training_set.csv")

# Import the test dataset from CSV file
evaluation_set = pd.read_csv("evaluation_set.csv")
```

```
# Display summary statistics for the evaluation dataset
print('Evaluation Dataset Statistics:')
display(evaluation_set.describe(include='all'))

# Show the first few rows of the training dataset
print('Training Dataset Preview:')
display(training_data.head())

# Present summary statistics for the training dataset
print('Training Dataset Statistics:')
display(training_data.describe(include='all'))

# Exhibit the first few rows of the evaluation dataset
print('Evaluation Dataset Preview:')
display(evaluation_set.head())
```

0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1

count	7613.000000	7552	5080	7613	7613.00000
unique	NaN	221	3341	7503	NaN
top	NaN	fatalities	USA	11-Year-Old Boy Charged With Manslaughter of T...	NaN
freq	NaN	45	104	10	NaN
mean	5441.934848	NaN	NaN	NaN	0.42966
std	3137.116090	NaN	NaN	NaN	0.49506
min	1.000000	NaN	NaN	NaN	0.00000
25%	2734.000000	NaN	NaN	NaN	0.00000
50%	5408.000000	NaN	NaN	NaN	0.00000
75%	8146.000000	NaN	NaN	NaN	1.00000
max	10873.000000	NaN	NaN	NaN	1.00000

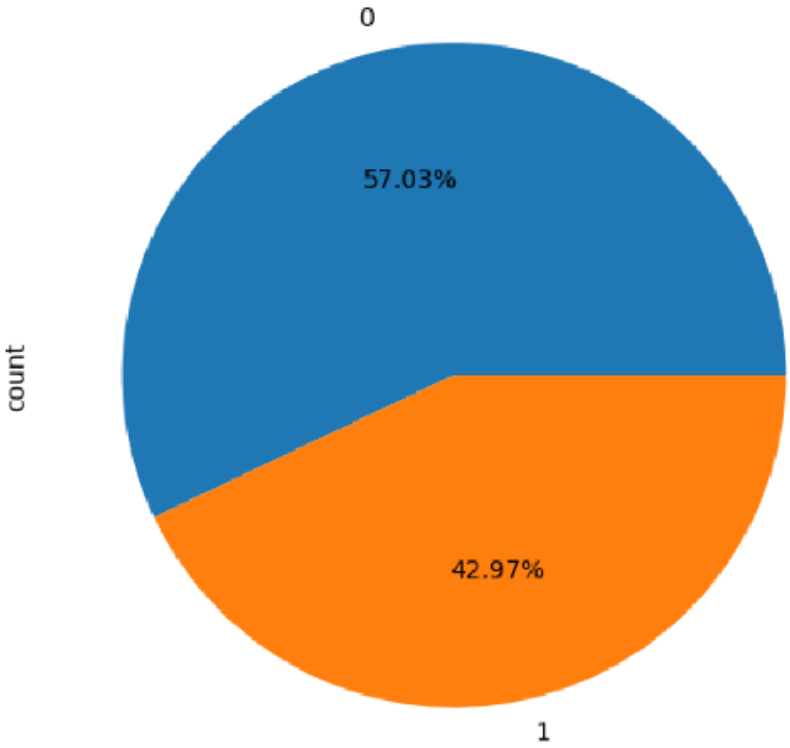
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan

count	3263.000000	3237	2158	3263
unique	NaN	221	1602	3243
top	NaN	deluged	New York	11-Year-Old Boy Charged With Manslaughter of T...
freq	NaN	23	38	3
mean	5427.152927	NaN	NaN	NaN
std	3146.427221	NaN	NaN	NaN
min	0.000000	NaN	NaN	NaN
25%	2683.000000	NaN	NaN	NaN
50%	5500.000000	NaN	NaN	NaN
75%	8176.000000	NaN	NaN	NaN
max	10875.000000	NaN	NaN	NaN

```
# Check for missing values in the training dataset
print('Null values in Train set')
display(training_data.isna().sum())

# Check for missing values in the evaluation dataset
print('Null values in Train set')
display(evaluation_set.isna().sum())
```

```
Null values in Train set:
id             0
keyword       61
location    2533
text           0
target         0
dtype: int64
Null values in Test set:
id             0
keyword       26
location    1105
text           0
dtype: int64
```

```python
# Set up the figure size
plt.figure(figsize=(8,8))

# Create a donut chart of the outcome variable distribution
outcome_counts = training_data['label'].value_counts()
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99']
explode = (0.05, 0, 0, 0)  # To emphasize the first slice

outcome_counts.plot(kind='pie', autopct='%1.1f%%', startangle=90, colors=colors,
                    explode=explode, pctdistance=0.85, wedgeprops=dict(width=0.5))

# Customize the plot
plt.title('Distribution of Outcome Labels', fontsize=16)
plt.ylabel('')  # Remove y-label
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle

# Add a circle at the center to create a donut chart
center_circle = plt.Circle((0,0), 0.70, fc='white')
fig = plt.gcf()
fig.gca().add_artist(center_circle)

# Display the plot
plt.tight_layout()
plt.show()
```




```python
def visualize_frequent_terms(dataframe, dataset_type, color_scheme, top_n=20):
    # Set the plot style
    sns.set_style('darkgrid')

    # Create the figure and axes
    fig, ax = plt.subplots(figsize=(10, 10))

    # Get the top N most frequent terms
    frequent_terms = dataframe['term'].value_counts()[:top_n]

    # Create a horizontal bar plot
    bars = sns.barplot(x=frequent_terms, y=frequent_terms.index, palette=color_scheme, ax=ax)

    # Add value labels to the bars
    for container in bars.containers:
        ax.bar_label(container, padding=5)

    # Customize x-axis labels
    ax.set_xticklabels([f'{int(i):,}' for i in ax.get_xticks()], fontsize=9)

    # Customize y-axis labels
    ax.set_yticklabels([t.get_text() for t in ax.get_yticklabels()], fontsize=11)

    # Set title and labels
    plt.title(f'Top {top_n} Frequent Terms in {dataset_type} Dataset', fontsize=16, pad=20)
    plt.xlabel('Frequency', fontsize=12)
    plt.ylabel('Terms', fontsize=12)

    # Adjust layout and display the plot
    plt.tight_layout()
    plt.show()

# Visualize frequent terms for training and evaluation datasets
visualize_frequent_terms(training_data, 'Training', 'viridis')
visualize_frequent_terms(evaluation_set, 'Evaluation', 'magma')
```