In [ ]:

```
#GitHub link for this assigment - https://github.com/ass2testing/week4.git
```

Step 1: Import Necessary Libraries First, import the libraries you'll need for data manipulation, visualization, and model building.

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Step 2: Load the Data Load the training and test data from the Kaggle dataset

In [2]:

```
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')
```

Step 3: Explore the Data Get a sense of what the data looks like.

In [3]:

```
print(train_data.head())
print(train_data.info())
print(train_data.describe())
```

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

   target
0       1
1       1
2       1
3       1
4       1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   id        7613 non-null   int64 
 1   keyword   7552 non-null   object
 2   location  5080 non-null   object
 3   text      7613 non-null   object
 4   target    7613 non-null   int64 
dtypes: int64(2), object(3)
memory usage: 297.5+ KB
None
                 id      target
count   7613.000000  7613.00000
mean    5441.934848     0.42966
std     3137.116090     0.49506
min        1.000000     0.00000
25%     2734.000000     0.00000
50%     5408.000000     0.00000
75%     8146.000000     1.00000
max    10873.000000     1.00000
```

Step 4: Preprocess the Text Data Clean and preprocess the text data by removing stopwords, punctuation, and applying other necessary transformations.

In [4]:

```python
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'\d+', '', text)  # Remove numbers
    text = re.sub(r'https?://\S+|www\.\S+', '', text)  # Remove URLs
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove punctuation
    text = text.split()
    text = [word for word in text if word not in stop_words]
    text = ' '.join(text)
    return text

train_data['cleaned_text'] = train_data['text'].apply(preprocess_text)
test_data['cleaned_text'] = test_data['text'].apply(preprocess_text)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Gyadav\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Step 5: Vectorize the Text Data Convert the text data into numerical features using TF-IDF Vectorizer.

In [5]:

```
vectorizer = TfidfVectorizer(max_features=1000)
X = vectorizer.fit_transform(train_data['cleaned_text']).toarray()
y = train_data['target']

X_test = vectorizer.transform(test_data['cleaned_text']).toarray()
```

Step 6: Train-Test Split Split the training data into training and validation sets.

In [6]:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state
```

Step 7: Build and Train the Model Use a simple Logistic Regression model to start.

In [7]:

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[7]:

```
▾ LogisticRegression
LogisticRegression()
```

Step 8: Evaluate the Model Check the accuracy and other metrics on the validation set.

In [9]:

```
y_pred = model.predict(X_val)
print('Accuracy:', accuracy_score(y_val, y_pred))
print('Confusion Matrix:\n', confusion_matrix(y_val, y_pred))
print('Classification Report:\n', classification_report(y_val, y_pred))
```

```
Accuracy: 0.7984241628365069
Confusion Matrix:
 [[766 108]
 [199 450]]
Classification Report:
               precision    recall  f1-score   support

           0       0.79      0.88      0.83       874
           1       0.81      0.69      0.75       649

    accuracy                           0.80      1523
   macro avg       0.80      0.78      0.79      1523
weighted avg       0.80      0.80      0.80      1523
```

Step 9: Make Predictions on the Test Set Finally, use the trained model to make predictions on the test set.

In [10]:

```
test_predictions = model.predict(X_test)
submission = pd.DataFrame({'id': test_data['id'], 'target': test_predictions})
submission.to_csv('submission.csv', index=False)
```

In [ ]: