Required Libraries

In [22]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package stopwords to C:\Users\s monisi
[nltk_data]     prabha\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to C:\Users\s monisi
[nltk_data]     prabha\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[22]:

```
True
```

Load dataset

In [23]:

```
file_path = r'C:\Users\s monisi prabha\Downloads\disaster_tweets_data(DS).csv'
data = pd.read_csv(file_path)
```

Handle missing values

In [24]:

```
data = data.dropna()
```

Preprocess tweets

In [25]:

```
def preprocess_text(text):
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)  # Remov
    text = re.sub(r'\@w+|\#','', text)  # Remove mentions and hashtags
    text = re.sub(r'[^\w\s]', '', text)  # Remove punctuations
    text = text.lower()  # Convert to lowercase
    tokens = text.split()  # Tokenize the text
    tokens = [word for word in tokens if word not in stopwords.words('english')]  #
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Lemmatize words
    return ' '.join(tokens)
```

Apply preprocessing to the dataset

In [26]:

```
data['tweets'] = data['tweets'].apply(preprocess_text)
```

Convert tweets into vectors using CountVectorizer or TF-IDF

Using TF-IDF

In [27]:

```
vectorizer = TfidfVectorizer(max_features=5000)
x = vectorizer.fit_transform(data['tweets']).toarray()
```

Split data into training and testing

In [28]:

```
y = data['target']
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_stat
```

Multinomial Naive Bayes

In [29]:

```
nb_model = MultinomialNB()
nb_model.fit(x_train, y_train)
y_pred_nb = nb_model.predict(x_test)
```

Logistic Regression

In [30]:

```
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(x_train, y_train)
y_pred_lr = lr_model.predict(x_test)
```

K-Nearest Neighbors

In [31]:

```
knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(x_train, y_train)
y_pred_knn = knn_model.predict(x_test)
```

Evaluate the models

In [32]:

```
def evaluate_model(y_test, y_pred, model_name):
    print(f"Model: {model_name}")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))
    print("Accuracy: ", accuracy_score(y_test, y_pred))
    print("\n")
```

Evaluate NB

In [33]:

```
evaluate_model(y_test, y_pred_nb, "Multinomial Naive Bayes")
```

```
Model: Multinomial Naive Bayes
Confusion Matrix:
 [[772 102]
 [205 444]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.88      0.83       874
           1       0.81      0.68      0.74       649

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

Accuracy:  0.7984241628365069
```

Evaluate LR

In [34]:

```
evaluate_model(y_test, y_pred_lr, "Logistic Regression")
```

```
Model: Logistic Regression
Confusion Matrix:
 [[776  98]
 [211 438]]
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.89      0.83       874
           1       0.82      0.67      0.74       649

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

Accuracy:  0.7971109652002626
```

Evaluate KNN

In [35]:

```
evaluate_model(y_test, y_pred_knn, "KNN")
```

```
Model: KNN
Confusion Matrix:
 [[858  16]
 [462 187]]
Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.98      0.78       874
           1       0.92      0.29      0.44       649

    accuracy                           0.69      1523
   macro avg       0.79      0.63      0.61      1523
```

```
weighted avg       0.77      0.69      0.64      1523

Accuracy:  0.6861457649376231
```

Compare accuracies and select the best model

In [36]:

```
models = {
    "Multinomial Naive Bayes": accuracy_score(y_test, y_pred_nb),
    "Logistic Regression": accuracy_score(y_test, y_pred_lr),
    "KNN": accuracy_score(y_test, y_pred_knn)
}

best_model = max(models, key=models.get)
print(f"Best model: {best_model} with accuracy {models[best_model]:.4f}")
```

```
Best model: Multinomial Naive Bayes with accuracy 0.7984
```