In [3]:
```
import pandas as pd
import numpy as np
import re
import string
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 classification_report, confusion_matrix, accuracy_score
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
nltk.download('stopwords')
nltk.download('wordnet')
```

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

Out[3]: True

In [4]:
```
df = pd.read_csv('disaster_tweets_data(DS).csv')
print(df.head())
```

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

In [5]:
```
df.isnull().sum()
df.dropna(inplace=True)
```

In [9]:
```
def preprocess_text(text):
    text = text.lower()  # Lowercase
    text = re.sub(r'@\w+', '', text)  # Remove handles
    text = re.sub(r'http\S+|www\S+|https\S+', '', text)  # Remove links
    text = text.translate(str.maketrans('', '', string.punctuation))  # Remove punc
    tokens = text.split()
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]  # Lemmatization
    return ' '.join(tokens)

df['cleaned_tweets'] = df['tweets'].apply(preprocess_text)
```

In [11]:
```
# Choose one: CountVectorizer or TfidfVectorizer
vectorizer = TfidfVectorizer()
```

```
X = vectorizer.fit_transform(df['cleaned_tweets'])
y = df['target']
```

In [13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [15]:
```
# a) Multinomial Naive Bayes
nb = MultinomialNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict(X_test)

# b) Logistic Regression
lr = LogisticRegression()
lr.fit(X_train, y_train)
lr_pred = lr.predict(X_test)

# c) KNN
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_test)
```

In [17]:
```
models = {'Naive Bayes': nb_pred, 'Logistic Regression': lr_pred, 'KNN': knn_pred}
for name, pred in models.items():
    print(f"\n{name} Results:")
    print("Confusion Matrix:\n", confusion_matrix(y_test, pred))
    print("Classification Report:\n", classification_report(y_test, pred))
    print("Accuracy:", accuracy_score(y_test, pred))
```

```
Naive Bayes Results:
Confusion Matrix:
 [[786  88]
 [217 432]]
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.90      0.84       874
           1       0.83      0.67      0.74       649

    accuracy                           0.80      1523
   macro avg       0.81      0.78      0.79      1523
weighted avg       0.80      0.80      0.80      1523

Accuracy: 0.7997373604727511

Logistic Regression Results:
Confusion Matrix:
 [[790  84]
 [233 416]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.90      0.83       874
           1       0.83      0.64      0.72       649

    accuracy                           0.79      1523
   macro avg       0.80      0.77      0.78      1523
weighted avg       0.80      0.79      0.79      1523

Accuracy: 0.7918581746552856

KNN Results:
Confusion Matrix:
 [[745 129]
 [226 423]]
Classification Report:
              precision    recall  f1-score   support

           0       0.77      0.85      0.81       874
           1       0.77      0.65      0.70       649

    accuracy                           0.77      1523
   macro avg       0.77      0.75      0.76      1523
weighted avg       0.77      0.77      0.76      1523

Accuracy: 0.7669074195666448
```

In [ ]: