Create a classification model to predict the sentiment either (1 or 0) based on Disaster tweets

In [1]:
```
import pandas as pd
```

In [2]:
```
import nltk
```

In [3]:
```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report
```

In [4]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\HP\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

In [5]:
```
nltk.download('stopwords')
```

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

Out[5]:
```
True
```

In [6]:
```
df=pd.read_csv('disaster_tweets_data(DS).csv')
```

In [7]:
```
df.head(5)
```

Out[7]:

	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 [8]:
```
df.tail(5)
```

Out[8]:

	tweets	target
7608	Two giant cranes holding a bridge collapse int...	1
7609	@aria_ahrary @TheTawniest The out of control w...	1
7610	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	Police investigating after an e-bike collided ...	1
7612	The Latest: More Homes Razed by Northern Calif...	1

In [9]:
```
df.shape
```

Out[9]:
```
(7613, 2)
```

1)Remove handle null values (if any).

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
tweets    0
target    0
dtype: int64
```

Preprocess

In [11]:
```
def preprocess_text(text):
    # Tokenizing words
    words = nltk.word_tokenize(text)

    # Convert words to lowercase
    words = [word.lower() for word in words]

    # Removing punctuations
    words = [word for word in words if word.isalnum()]

    # Removing stop words
    stop_words = set(stopwords.words('english'))
    words = [word for word in words if word not in stop_words]

    # Stemming using Porter Stemmer (
    stemmer = PorterStemmer()
    words = [stemmer.stem(word) for word in words]

    return ' '.join(words)

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

split the dataset into training and testing sets

In [12]:
```
X=df['preprocessed_tweets']
y=df['target']
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.5,random_state=4
```

TF-IDF vectorization

In [13]:
```
tfidf_vectorizer=TfidfVectorizer(max_features=5000)
X_train_tfidf=tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf=tfidf_vectorizer.transform(X_test)
```

classification model(Multinomail naive bayes)

In [14]:
```
classifier=MultinomialNB()
```

In [15]:
```
classifier.fit(X_train_tfidf,y_train)
```

Out[15]:
```
MultinomialNB
MultinomialNB()
```

In [16]:
```
y_pred=classifier.predict(X_test_tfidf)
```

In [17]:
```
accuracy=accuracy_score(y_test,y_pred)
```

In [18]:
```
classification_rep=classification_report(y_test,y_pred)
```

In [19]:
```
print("accuracy",accuracy,"\n classification_report:",classification_rep)
```

```
accuracy 0.7972156553716837
 classification_report:               precision    recall  f1-score   support

           0       0.78      0.90      0.84      2185
           1       0.83      0.66      0.74      1622

    accuracy                           0.80      3807
   macro avg       0.80      0.78      0.79      3807
weighted avg       0.80      0.80      0.79      3807
```

LogisticRegression

In [20]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
```

In [21]:
```
logistic_regression = LogisticRegression()
```

In [22]:
```
logistic_regression.fit(X_train_tfidf,y_train)
```

Out[22]:
```
LogisticRegression
LogisticRegression()
```

In [23]:
```
y_pred_logistic_regression=logistic_regression.predict(X_test_tfidf)
```

In [24]:
```
confusion_matrix_logistic_regression=confusion_matrix(y_test,y_pred_logistic_re
classification_report_logistic_regression=classification_report(y_test,y_pred_l
```

In [25]:
```
confusion_matrix_logistic_regression
```

Out[25]:
```
array([[1954,  231],
       [ 532, 1090]], dtype=int64)
```

In [26]:
```
classification_report_logistic_regression
```

Out[26]:
```
'              precision    recall  f1-score   support\n\n           0
0.79      0.89      0.84      2185\n           1       0.83      0.67      0.
74      1622\n\n    accuracy                           0.80      3807\n   mac
ro avg       0.81      0.78      0.79      3807\nweighted avg       0.80
0.80      0.80      3807\n'
```

In [27]:
```
accuracy_lr=accuracy_score(y_test,y_pred_logistic_regression)
```

In [28]:
```
accuracy_lr
```

Out[28]:
```
0.7995797215655371
```

KNN Classifier

In [29]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [30]:
```
knn_classifier=KNeighborsClassifier()
```

In [31]:
```
knn_classifier.fit(X_train_tfidf,y_train)
```

Out[31]:
```
KNeighborsClassifier
KNeighborsClassifier()
```

In [32]:
```
y_pred_knn_classifier=knn_classifier.predict(X_test_tfidf)
```

In [33]:
```
confusion_matrix_knn_classifier=confusion_matrix(y_test,y_pred_knn_classifier)
classification_report_knn_classifier=classification_report(y_test,y_pred_knn_cl
```

In [34]:
```
accuracy_knn=accuracy_score(y_test,y_pred_knn_classifier)
```

In [35]:
```
confusion_matrix_knn_classifier
```

Out[35]:
```
array([[2169,   16],
       [1329,  293]], dtype=int64)
```

In [36]:
```
classification_report_knn_classifier
```

Out[36]:
```
'              precision    recall  f1-score   support\n\n           0
0.62      0.99      0.76      2185\n           1       0.95      0.18      0.
30      1622\n\n    accuracy                           0.65      3807\n   mac
ro avg       0.78      0.59      0.53      3807\nweighted avg       0.76
0.65      0.57      3807\n'
```

In [37]:
```
accuracy_knn
```

Out[37]:
```
0.6467034410296821
```

In [38]:
```
best_accuracy = 0
best_model = ""
```

In [39]:
```
models = [
    ("Logistic Regression", y_pred_logistic_regression),
    ("K-Nearest Neighbors", y_pred_knn_classifier),
    ("Multinomail naive bayes",y_pred)]
```

In [40]:
```
for model_name, predictions in models:
    if accuracy > best_accuracy:
        best_accuracy = accuracy
        best_model = model_name
```

best accuracy

In [41]:
```
print("The model with the best accuracy is",best_model,"with an accuracy of",be
```

```
The model with the best accuracy is Logistic Regression with an accuracy of
0.7972156553716837
```