```
from google.colab import files
upload1=files.upload()
```

```
import pandas as pd
import io
train_df=pd.read_csv(io.BytesIO(upload1['train.csv']))
```

```
upload2=files.upload()
```

```
test_df=pd.read_csv(io.BytesIO(upload2['test.csv']))
```

```
train_df.shape
```

```
test_df.shape
```

```
train_df.head()
```

```
test_df.head()
```

```
train_df.info()
```

```
test_df.info()
```

```
a=train_df.isnull().sum()
print(a)
print("Total no. of missing value in train data is",sum(a))
```

```
b=test_df.isnull().sum()
print(b)
print("Total no. of missing value in test data is",sum(b))
```

```
train_df.describe()
```

```
test_df.describe()
```

```
real_tweets=len(train_df[train_df["target"]==1])
real_tweets_percentage=real_tweets/train_df.shape[0]*100
fake_tweets_percentage=100-real_tweets_percentage
print("Real tweets percentage: ",real_tweets_percentage)
print("Fake tweets percentage: ",fake_tweets_percentage)
```

```
import matplotlib.pyplot as plt
train_df.isna().sum().plot(kind="bar")
plt.title("no of null values in train data")
plt.show()
```

```
import matplotlib.pyplot as plt
test_df.isna().sum().plot(kind="bar")
plt.title("no of null values in test data")
plt.show()
```

```
import seaborn as sns
sns.countplot(x='target',data=train_df)
```

```
length_train = train_df['text'].str.len()
length_test = test_df['text'].str.len()
plt.hist(length_train, label="train_tweets")
plt.hist(length_test, label="test_tweets")
plt.legend()
plt.show()
```

```
disaster_tweets = train_df[train_df['target']==1]['text']
for i in range(1,10):
    print(disaster_tweets[i])
```

```
non_disaster_tweets = train_df[train_df['target']==1]['text']
for i in range(1,10):
    print(non_disaster_tweets[i])
```

```
from wordcloud import WordCloud
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[20, 5])
wordcloud1 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(disaster_tweets))
ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Disaster Tweets',fontsize=40);

wordcloud2 = WordCloud( background_color='white',
                        width=600,
                        height=400).generate(" ".join(non_disaster_tweets))
ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Non Disaster Tweets',fontsize=40);
```

```
import re
import string
```

```
def clean_text(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text

# Applying the cleaning function to both test and training datasets
train_df['text'] = train_df['text'].apply(lambda x: clean_text(x))
test_df['text'] = test_df['text'].apply(lambda x: clean_text(x))

# Let's take a look at the updated text
train_df['text'].head()
```

```
import nltk
nltk.download('stopwords')
```

```
import nltk
from nltk.corpus import stopwords
tokenizer=nltk.tokenize.RegexpTokenizer(r'\w+')
train_df['text']=train_df['text'].apply(lambda x:tokenizer.tokenize(x))
test_df['text']=test_df['text'].apply(lambda x:tokenizer.tokenize(x))
train_df['text'].head()
```

```
from nltk.corpus import stopwords
def remove_stopwords(text):
    words=[w for w in text if w not in stopwords.words('english')]
    return words
train_df['text']=train_df['text'].apply(lambda x : remove_stopwords(x))
```

```
train_df['text']=train_df['text'].apply(lambda x : remove_stopwords(x))
test_df['text']=test_df['text'].apply(lambda x : remove_stopwords(x))
test_df.head()
```

```
stopwords.words('english')
```

```
len(stopwords.words('english'))
```

```
import nltk
nltk.download('wordnet')
```

```
from nltk.stem import WordNetLemmatizer
lem=WordNetLemmatizer()
def lem_word(x):
    return [lem.lemmatize(w) for w in x]
```

```
train_df['text']=train_df['text'].apply(lem_word)
test_df['text']=test_df['text'].apply(lem_word)
```

```
train_df['text'][:10]
```

```
     0    [deed, reason, earthquake, may, allah, forgive...
```

```
def combine_text(list_of_text):
    '''Takes a list of text and combines them into one large chunk of text.'''
    combined_text = ' '.join(list_of_text)
    return combined_text

train_df['text'] = train_df['text'].apply(lambda x : combine_text(x))
test_df['text'] = test_df['text'].apply(lambda x : combine_text(x))
train_df['text']
train_df.head()
```

```
     id  keyword  location                                          text  target
  0   1      NaN       NaN      deed reason earthquake may allah forgive u       1
  1   4      NaN       NaN              forest fire near la ronge sask canada       1
  2   5      NaN       NaN      resident asked shelter place notified officer ...       1
  3   6      NaN       NaN      people receive wildfire evacuation order calif...       1
  4   7      NaN       NaN   got sent photo ruby alaska smoke wildfire pour...       1
```

```
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
count_vectorizer=CountVectorizer()
train_vector=count_vectorizer.fit_transform(train_df['text'])
test_vector=count_vectorizer.transform(test_df['text'])
print(train_vector[0].todense())
```

```
[[0 0 0 ... 0 0 0]]
```

```
tfidf=TfidfVectorizer(min_df=2,max_df=0.5,ngram_range=(1,2))
train_tfidf=tfidf.fit_transform(train_df['text'])
test_tfidf=tfidf.transform(test_df['text'])
```

```
import xgboost as xgb
xgb_param=xgb.XGBClassifier(max_depth=5,n_estimators=300,colsample_bytree=0.8,nthread=10,lear
from sklearn import model_selection
```

```
scores=model_selection.cross_val_score(xgb_param,train_vector,train_df['target'],cv=5,scoring
scores
```

```
array([0.47905478, 0.33884298, 0.4247619 , 0.33072626, 0.48513902])
```

```
scorest=model_selection.cross_val_score(xgb_param,train_tfidf,train_df['target'],cv=5,scoring
scorest
```

```
array([0.47098214, 0.33790919, 0.43126177, 0.32959641, 0.50501002])
```

```
xgb_param.get_params()
```

```
{'base_score': 0.5,
 'booster': 'gbtree',
 'colsample_bylevel': 1,
 'colsample_bynode': 1,
 'colsample_bytree': 0.8,
 'gamma': 0,
 'learning_rate': 0.05,
 'max_delta_step': 0,
 'max_depth': 5,
 'min_child_weight': 1,
 'missing': None,
 'n_estimators': 300,
 'n_jobs': 1,
 'nthread': 10,
 'objective': 'binary:logistic',
 'random_state': 0,
 'reg_alpha': 0,
 'reg_lambda': 1,
 'scale_pos_weight': 1,
 'seed': None,
 'silent': None,
 'subsample': 1,
 'verbosity': 1}
```

```
from sklearn.naive_bayes import MultinomialNB
mnb=MultinomialNB(alpha=2.0)
scores=model_selection.cross_val_score(mnb,train_vector,train_df['target'],cv=10,scoring='f1'
print("score:",scores)
scorest=model_selection.cross_val_score(mnb,train_tfidf,train_df['target'],cv=10,scoring='f1'
print("score of tfidf:",scorest)
```

```
score: [0.69207317 0.5512605  0.58394161 0.53465347 0.66951567 0.62831858
 0.6735905  0.6029654  0.7107438  0.74614306]
score of tfidf: [0.6221374  0.47157895 0.58844765 0.46616541 0.59507042 0.50929368
 0.59107807 0.51639344 0.71890971 0.75409836]
```

```
mnb.get_params()
```

```
{'alpha': 1.0, 'class_prior': None, 'fit_prior': True}
```

```
from sklearn.linear_model import LogisticRegression
lg = LogisticRegression(C=1.0)
scores = model_selection.cross_val_score(lg, train_vector, train_df["target"], cv=5, scoring=
print("score:",scores)
scorest = model_selection.cross_val_score(lg, train_tfidf, train_df["target"], cv=5, scoring=
print("score of tfidf:",scorest)
```

```
score: [0.61904762 0.53401361 0.58340181 0.53521127 0.69756481]
score of tfidf: [0.58502024 0.50644567 0.54725473 0.48190279 0.66840731]
```

```
lg.get_params()
```

```
{'C': 1.0,
 'class_weight': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 100,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

```
mnb.fit(train_tfidf, train_df["target"])
y_pred=mnb.predict(test_tfidf)
```

```
y_pred
```

```
array([1, 0, 1, ..., 1, 1, 1])
```

```
submission_df2=pd.DataFrame({'Id':test_df['id'],'target':y_pred})
```

```
submission_df2.to_csv('submission_df2.csv',index=False)
```

```
submission_df2=pd.read_csv('submission_df2.csv')
```

```
submission_df2.head()
```

	Id	target
0	0	1
1	2	0
2	3	1
3	9	1
4	11	1