DINESH KUMAR

225229108

In [1]:

```
import pandas as pd
import re
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
import string
import numpy as np
```

```
[nltk_data] Downloading package punkt to /usr/share/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [2]:

```
df = pd.read_csv('train.csv')
df_test = pd.read_csv('test.csv')
```

In [3]:

```
df
```

Out[3]:

	id	keyword	location	text	target
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1
...	...	...	...	...	...
7608	10869	NaN	NaN	Two giant cranes holding a bridge collapse int...	1
7609	10870	NaN	NaN	@aria_ahrary @TheTawniest The out of control w...	1
7610	10871	NaN	NaN	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	10872	NaN	NaN	Police investigating after an e-bike collided ...	1
7612	10873	NaN	NaN	The Latest: More Homes Razed by Northern Calif...	1

7613 rows × 5 columns

Preprocessing - 1:

```
1. convert sentence to lower case.
2. Remove numbers if any.
3. Remove HTML tags
4. Remove URLS in a sentence.
5. Remove emojis and other symbols is any.
6. Remove Punctuation marks.
```

Apply the function preprocess for both train and test data.

In [4]:

```
def preprocess1(text):
    text=str(text).lower() #Converts text to lowercase
    text=re.sub('\d+', '', text) #removes numbers
    text=re.sub('\[.*?\]', '', text) #removes HTML tags
    text=re.sub('https?://\S+|www\.\S+', '', text) #removes url
    text=re.sub(r"["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U000027B0"
                           u"\U000024C2-\U0001F251"
                           "]+", "", text) #removes emojis
    text=re.sub('[%s]' % re.escape(string.punctuation),'',text) #removes punctuation
    return text
```

In [5]:

```
df['clean_text']=df['text'].apply(preprocess1)
df_test['clean_text']=df_test['text'].apply(preprocess1)

df.head()
```

Out[5]:

	id	keyword	location	text	target	clean_text
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	our deeds are the reason of this earthquake ma...
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	forest fire near la ronge sask canada
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	all residents asked to shelter in place are be...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	people receive wildfires evacuation orders in...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	just got sent this photo from ruby alaska as s...

Stop Words:

In [6]:

```
from nltk.corpus import stopwords
nltk.download('stopwords')
stop=set(stopwords.words('english'))
stop.remove('not')
```

```
[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Stemming:

In [7]:

```
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize
ps = PorterStemmer()
```

In [8]:

```
def stemming(text):
    stem_strings=list(map(lambda y: [ps.stem(word) for word in word_tokenize(y) if word not
    return stem_strings
```

In [9]:

```
text_after_stemming=stemming(df['clean_text'])
text_after_stemming[1:5]
```

Out[9]:

```
[['forest', 'fire', 'near', 'la', 'rong', 'sask', 'canada'],
 ['resid',
  'ask',
  'shelter',
  'place',
  'notifi',
  'offic',
  'evacu',
  'shelter',
  'place',
  'order',
  'expect'],
 ['peopl', 'receiv', 'wildfir', 'evacu', 'order', 'california'],
 ['got',
  'sent',
  'photo',
  'rubi',
  'alaska',
  'smoke',
  'wildfir',
  'pour',
  'school']]
```

Preprocess - 2

In [10]:

```
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package wordnet to /usr/share/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [11]:

```
lemma=WordNetLemmatizer()
def preprocess2(text):
    final_text=text.apply(lambda x: ' '.join(lemma.lemmatize(word) for word in x.split(' '))
    return final_text
```

In [12]:

```
df['final']=preprocess2(df['clean_text'])
df_test['final']=preprocess2(df_test['clean_text'])
```

In [13]:

```
df.head()
```

Out[13]:

	id	keyword	location	text	target	clean_text	final
0	1	NaN	NaN	Our Deeds are the Reason of this #earthquake M...	1	our deeds are the reason of this earthquake ma...	deed reason earthquake may allah forgive u
1	4	NaN	NaN	Forest fire near La Ronge Sask. Canada	1	forest fire near la ronge sask canada	forest fire near la ronge sask canada
2	5	NaN	NaN	All residents asked to 'shelter in place' are ...	1	all residents asked to shelter in place are be...	resident asked shelter place notified officer ...
3	6	NaN	NaN	13,000 people receive #wildfires evacuation or...	1	people receive wildfires evacuation orders in...	people receive wildfire evacuation order cali...
4	7	NaN	NaN	Just got sent this photo from Ruby #Alaska as ...	1	just got sent this photo from ruby alaska as s...	got sent photo ruby alaska smoke wildfire pour...

Type *Markdown* and LaTeX: α^2

Words to vectors:

In [14]:

```
global dis_freq, ndis_freq
dis_freq=df.loc[df['target']==1, 'final'].str.split(expand=True).stack().value_counts().to_
ndis_freq=df.loc[df['target']==0, 'final'].str.split(expand=True).stack().value_counts().to
```

In [15]:

```
def create_vector(tweet):
    total_dis =0
    total_ndis =0
    for word in tweet.split(' '):
        total_dis+=dis_freq[word] if word in dis_freq.keys() else 0
        total_ndis+=ndis_freq[word] if word in ndis_freq.keys() else 0
    return [total_dis, total_ndis]
```

In [16]:

```
vector=df['final'].apply(create_vector)
vector2=df_test['final'].apply(create_vector)
```

In [17]:

```
df1 = pd.DataFrame(vector.values.tolist()).add_prefix('data')
df2 = pd.DataFrame(vector2.values.tolist()).add_prefix('data')
print(df1)
```

```
      data0  data1
0       220    217
1       392    119
2       159     72
3       366    126
4       192    208
...     ...    ...
7608    318    118
7609    567    210
7610     60     11
7611    295    181
7612    604    118

[7613 rows x 2 columns]
```

In [18]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

In [19]:

```
def train_model(model,X,y, test):
    X_train,X_test, y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    print(classification_report(y_test,y_pred))
    return model.predict(test)
```

In [20]:

```
X=df1
y=df['target']
```

In [21]:

```
lr = LogisticRegression()
y_pred=train_model(lr,X,y,df2)
```

```
              precision    recall  f1-score   support

           0       0.80      0.87      0.83      1326
           1       0.79      0.70      0.74       958

    accuracy                           0.80      2284
   macro avg       0.80      0.78      0.79      2284
weighted avg       0.80      0.80      0.79      2284
```

In [22]:

```
from sklearn.feature_extraction.text import CountVectorizer
cv= CountVectorizer(max_features = 2500, binary=True)
# Max-features - vector length
X = cv.fit_transform(df['final']).toarray()
X_test = cv.transform(df_test['final']).toarray()
```

In [23]:

```
X
```

Out[23]:

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

In [24]:

```
y_pred=train_model(lr,X,y,X_test)
```

```
              precision    recall  f1-score   support

           0       0.80      0.86      0.83      1326
           1       0.78      0.69      0.74       958

    accuracy                           0.79      2284
   macro avg       0.79      0.78      0.78      2284
weighted avg       0.79      0.79      0.79      2284
```

In [25]:

```
#TD_IDF
from sklearn.feature_extraction.text import TfidfVectorizer
cv = TfidfVectorizer()
X_tdidf = cv.fit_transform(df['final'])
X_tdidf_test = cv.transform(df_test['final'])
```

In [26]:

```
X_tdidf[0].toarray()
```

Out[26]:

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

In [27]:

```
y_pred=train_model(lr,X_tdidf,y,X_tdidf_test)
```

```
              precision    recall  f1-score   support

           0       0.78      0.92      0.84      1326
           1       0.85      0.64      0.73       958

    accuracy                           0.80      2284
   macro avg       0.81      0.78      0.79      2284
weighted avg       0.81      0.80      0.79      2284
```

In [28]:

```
from sklearn.naive_bayes import MultinomialNB
mnb=MultinomialNB()
y_pred=train_model(mnb,X_tdidf,y,X_tdidf_test)
```

```
              precision    recall  f1-score   support

           0       0.78      0.92      0.84      1326
           1       0.85      0.65      0.74       958

    accuracy                           0.80      2284
   macro avg       0.82      0.78      0.79      2284
weighted avg       0.81      0.80      0.80      2284
```

In [33]:

```
submission = df_test[['id']].reset_index(drop=True)
submission['target'] = y_pred
```

In [34]:

```
y_pred
```

Out[34]:

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

In [31]:

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

In [32]:

```
submission
```

Out[32]:

```
         id  target
0         0       1
1         2       1
2         3       1
3         9       1
4        11       1
...     ...     ...
3258  10861       1
3259  10865       1
3260  10868       1
3261  10874       1
3262  10875       1

3263 rows × 2 columns
```

In [ ]:

```

```

In [ ]:

```

```