Dataset

- a set of tweets which have been divided into a training and a test set.
- The training set contains a target column identifying whether the tweet pertains to real disastor not.
- Binary Classification problem

1. Importing the necessary libraries

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.5.15)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.4)
```

```
import numpy as np
import pandas as pd

# text processing libraries
import re
import string
import nltk
from nltk.corpus import stopwords


# XGBoost
import xgboost as xgb
from xgboost import XGBClassifier

# sklearn
from sklearn import model_selection
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import f1_score
from sklearn import preprocessing, decomposition, model_selection,metrics, pipeline
from sklearn.model_selection import GridSearchCV, StratifiedKFold, RandomizedSearchCV


import matplotlib.pyplot as plt
import seaborn as sns

import os

import warnings
warnings.filterwarnings('ignore')
```

2. Reading the datasets

```
train = pd.read_csv('/content/drive/MyDrive/머신러닝 딥러닝/유런/train_nlp.csv')
print('Trainig data shape : ', train.shape)
train.head()
```

```
Trainig data shape :  (7613, 5)
```

	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

다음 단계: train변수로 코드 생성 추천 차트 보기

```
test = pd.read_csv('/content/drive/MyDrive/머신러닝 딥러닝/유런/test_nlp.csv')
print('Testing data shape : ', test.shape)
test.head()
```

```
Testing data shape :  (3263, 4)
```

	id	keyword	location	text
0	0	NaN	NaN	Just happened a terrible car crash
1	2	NaN	NaN	Heard about #earthquake is different cities, s...
2	3	NaN	NaN	there is a forest fire at spot pond, geese are...
3	9	NaN	NaN	Apocalypse lighting. #Spokane #wildfires
4	11	NaN	NaN	Typhoon Soudelor kills 28 in China and Taiwan

다음 단계: test변수로 코드 생성 추천 차트 보기

3. Basic EDA

Missing values

```
train.isnull().sum()
```

```
id             0
keyword       61
location    2533
text           0
target         0
dtype: int64
```

```
test.isnull().sum()
```

```
id             0
keyword       26
location    1105
text           0
dtype: int64
```

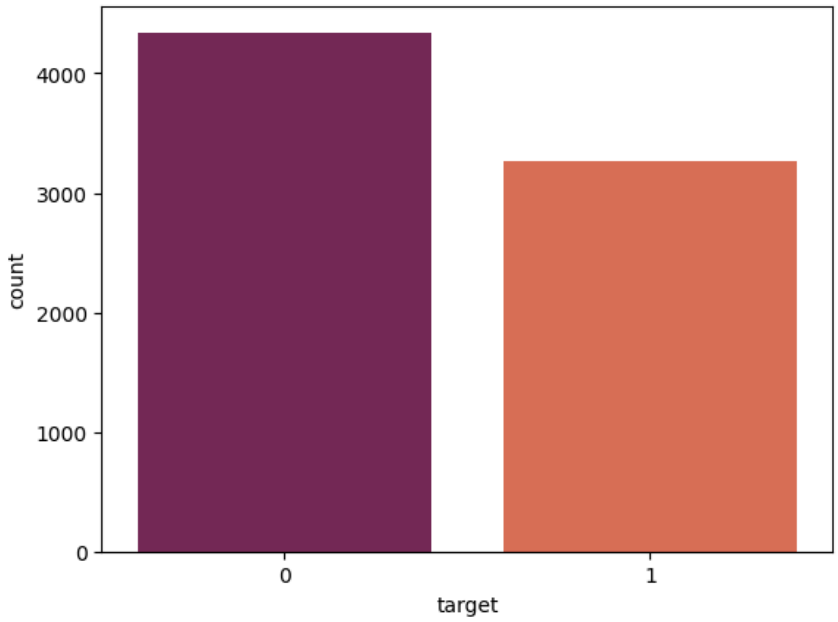
Exploring the Target Column

```
train['target'].value_counts()
```

```
target
0    4342
1    3271
Name: count, dtype: int64
```

```
sns.barplot(train['target'].value_counts(),
            palette='rocket')
```

```
<Axes: xlabel='target', ylabel='count'>
```



let's look at what the disater and the non disaster tweets look like

```
# a disaster tweet
disaster_tweets = train[train['target']==1]['text']
disaster_tweets.values[1]
```

```
'Forest fire near La Ronge Sask. Canada'
```

```
# not a disaster tweet
non_disaster_tweets = train[train['target']==0]['text']
non_disaster_tweets.values[1]
```

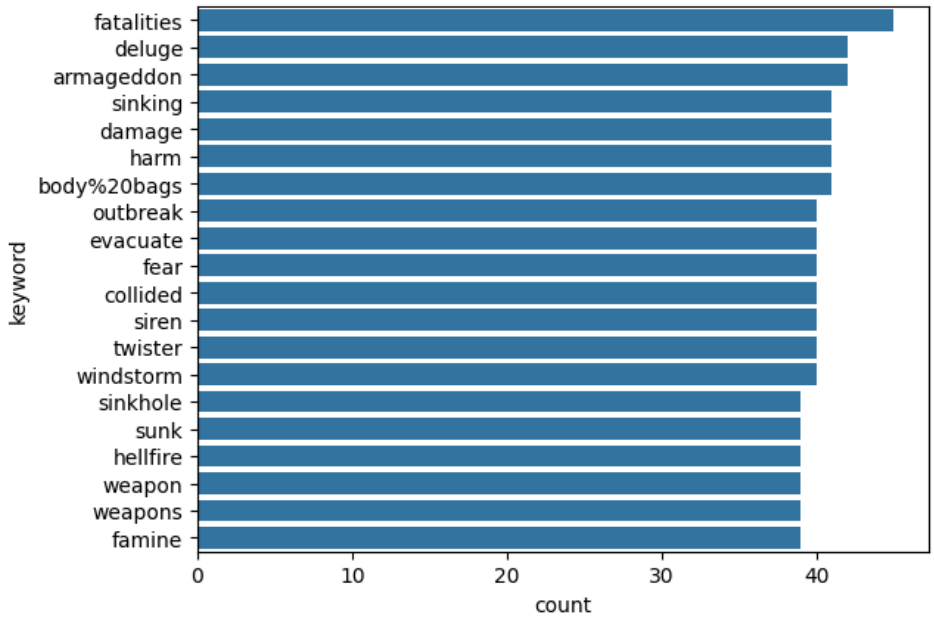
```
'I love fruits'
```

Exploring the 'keyword' column The keyword column denotes a keyword from the tweet

```
# top 20 keywords

sns.barplot(y=train['keyword'].value_counts()[:20].index,
            x=train['keyword'].value_counts()[:20],
            orient='h')
```

```
<Axes: xlabel='count', ylabel='keyword'>
```



How often the word 'disaster' come in the dataset and whether this help us in determining whether a tweet belongs to a disaster category or not

```
train.loc[train['text'].str.contains('disaster', na=False, case=False)].target.value_counts()
```

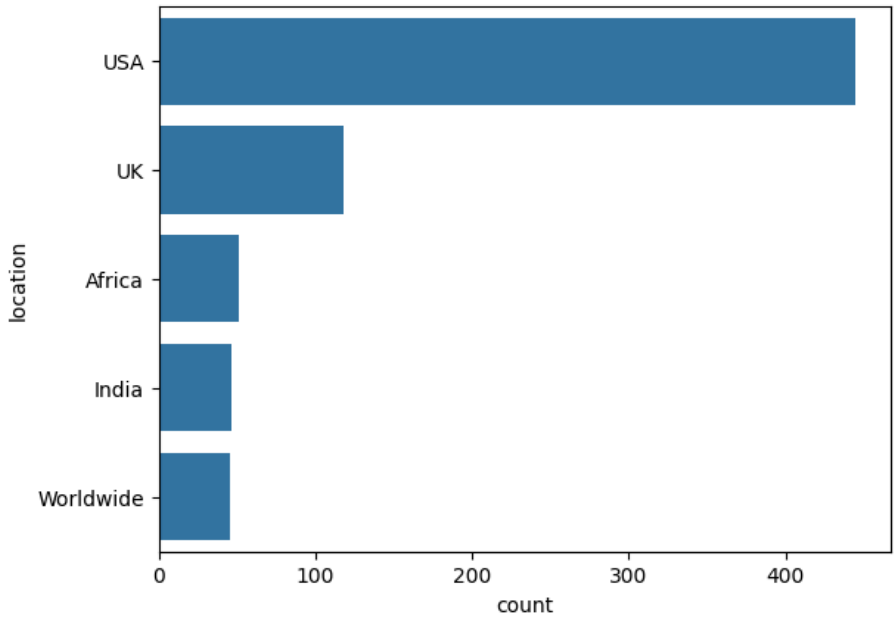
```
target
1    102
0     40
Name: count, dtype: int64
```

Exploring the 'location' column

```
# Replacing the ambigious locations name with Standard names
train['location'].replace({'United States':'USA',
                           'New York':'USA',
                            "London":'UK',
                            "Los Angeles, CA":'USA',
                            "Washington, D.C.":'USA',
                            "California":'USA',
                             "Chicago, IL":'USA',
                             "Chicago":'USA',
                            "New York, NY":'USA',
                            "California, USA":'USA',
                            "FLorida":'USA',
                            "Nigeria":'Africa',
                            "Kenya":'Africa',
                            "Everywhere":'Worldwide',
                            "San Francisco":'USA',
                            "Florida":'USA',
                            "United Kingdom":'UK',
                            "Los Angeles":'USA',
                            "Toronto":'Canada',
                            "San Francisco, CA":'USA',
                            "NYC":'USA',
                            "Seattle":'USA',
                            "Earth":'Worldwide',
                            "Ireland":'UK',
                            "London, England":'UK',
                            "New York City":'USA',
                            "Texas":'USA',
                            "London, UK":'UK',
                            "Atlanta, GA":'USA',
                            "Mumbai":"India"},inplace=True)

sns.barplot(y=train['location'].value_counts()[:5].index,x=train['location'].value_counts()[:5],
            orient='h')
```

```
<Axes: xlabel='count', ylabel='location'>
```



4. Text Data Preprocessing

1) Data Cleaning

```
# A quick glance over the existing data
train['text'][:5]
```

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

```
# Applying a first round of text cleaning techniques

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['text'] = train['text'].apply(lambda x: clean_text(x))
test['text'] = test['text'].apply(lambda x: clean_text(x))

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

```
0    our deeds are the reason of this earthquake ma...
1                forest fire near la ronge sask canada
2    all residents asked to shelter in place are be...
3     people receive wildfires evacuation orders in...
4    just got sent this photo from ruby alaska as s...
Name: text, dtype: object
```

2)Tokenization

```
text = "Are you coming , aren't you"
tokenizer1 = nltk.tokenize.WhitespaceTokenizer()
tokenizer2 = nltk.tokenize.TreebankWordTokenizer()
tokenizer3 = nltk.tokenize.WordPunctTokenizer()
tokenizer4 = nltk.tokenize.RegexpTokenizer(r'\w+')

print("Example Text: ",text)
print("----------------------------------------------------------------------------------------------------")
print("Tokenization by whitespace:- ",tokenizer1.tokenize(text))
print("Tokenization by words using Treebank Word Tokenizer:- ",tokenizer2.tokenize(text))
print("Tokenization by punctuation:- ",tokenizer3.tokenize(text))
print("Tokenization by regular expression:- ",tokenizer4.tokenize(text))
```

```
Example Text:  Are you coming , aren't you
----------------------------------------------------------------------------------------------------
Tokenization by whitespace:-  ['Are', 'you', 'coming', ',', "aren't", 'you']
Tokenization by words using Treebank Word Tokenizer:-  ['Are', 'you', 'coming', ',', 'are', "n't", 'you']
Tokenization by punctuation:-  ['Are', 'you', 'coming', ',', 'aren', "'", 't', 'you']
Tokenization by regular expression:-  ['Are', 'you', 'coming', 'aren', 't', 'you']
```

```
# Tokenizing the training and the test set
tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')
train['text'] = train['text'].apply(lambda x: tokenizer.tokenize(x))
test['text'] = test['text'].apply(lambda x: tokenizer.tokenize(x))
train['text'].head()
```

```
0    [our, deeds, are, the, reason, of, this, earth...
1        [forest, fire, near, la, ronge, sask, canada]
2    [all, residents, asked, to, shelter, in, place...
3    [people, receive, wildfires, evacuation, order...
4    [just, got, sent, this, photo, from, ruby, ala...
Name: text, dtype: object
```

3) Stopwords Removal

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

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
def remove_stopwords(text):
    """
    Removing stopwords belonging to english language

    """
    words = [w for w in text if w not in stopwords.words('english')]
    return words


train['text'] = train['text'].apply(lambda x : remove_stopwords(x))
test['text'] = test['text'].apply(lambda x : remove_stopwords(x))
train.head()
```

	id	keyword	location	text	target
0	1	NaN	NaN	[deeds, reason, earthquake, may, allah, forgiv...	1
1	4	NaN	NaN	[forest, fire, near, la, ronge, sask, canada]	1
2	5	NaN	NaN	[residents, asked, shelter, place, notified, o...	1
3	6	NaN	NaN	[people, receive, wildfires, evacuation, order...	1
4	7	NaN	NaN	[got, sent, photo, ruby, alaska, smoke, wildfi...	1

다음 단계: train변수로 코드 생성 추천 차트 보기

4) Token normailization

- Stemming : removing and replacing suffixes to get to the root form of the word, which is called the stem for instance cats - cat, wolves - wolv
- Lemmatization : Returns the base or dictionary form of a word, which is known as the lemma

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

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
# Stemming and Lemmatization examples
text = "feet cats wolves talked"

tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)

# Stemmer
stemmer = nltk.stem.PorterStemmer()
print("Stemming the sentence: ", " ".join(stemmer.stem(token) for token in tokens))

# Lemmatizer
lemmatizer=nltk.stem.WordNetLemmatizer()
print("Lemmatizing the sentence: ", " ".join(lemmatizer.lemmatize(token) for token in tokens))
```

```
Stemming the sentence:  feet cat wolv talk
Lemmatizing the sentence:  foot cat wolf talked
```

```
# After preprocessing, the text format
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['text'] = train['text'].apply(lambda x : combine_text(x))
test['text'] = test['text'].apply(lambda x : combine_text(x))
train['text']
train.head()
```

	id	keyword	location	text	target
0	1	NaN	NaN	deeds reason earthquake may allah forgive us	1
1	4	NaN	NaN	forest fire near la ronge sask canada	1
2	5	NaN	NaN	residents asked shelter place notified officer...	1
3	6	NaN	NaN	people receive wildfires evacuation orders cal...	1
4	7	NaN	NaN	got sent photo ruby alaska smoke wildfires pou...	1

다음 단계: train변수로 코드 생성 추천 차트 보기

A Text Preprocessing Function

```
# text preprocessing function
def text_preprocessing(text):
    """
    Cleaning and parsing the text.

    """
    tokenizer = nltk.tokenize.RegexpTokenizer(r'₩w+')

    nopunc = clean_text(text)
    tokenized_text = tokenizer.tokenize(nopunc)
    remove_stopwords = [w for w in tokenized_text if w not in stopwords.words('english')]
    combined_text = ' '.join(remove_stopwords)
    return combined_text
```

5. Transforming tokens to a vector

Bag of words

```
count_vectorizer = CountVectorizer()
train_vectors = count_vectorizer.fit_transform(train['text'])
test_vectors = count_vectorizer.transform(test["text"])

## Keeping only non-zero elements to preserve space
print(train_vectors[0].todense())
```

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

TFIDF Features

- Term Frequency: is a scoring of the frequency of the word in the current document.
- TF = (Number of times term t appears in a document)/(Number of terms in the document)
- Inverse Document Frequency: is a scoring of how rare the word is across documents.
- IDF = 1+log(N/n), where, N is the number of documents and n is the number of documents a term t has appeared in.

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

6. Building a Text Classification model

Logistic Regression Classifer

```
# Fitting a simple Logistic Regression on Counts
clf = LogisticRegression(C=1.0)
scores = model_selection.cross_val_score(clf, train_vectors, train["target"], cv=5, scoring="f1")
scores
```

```
array([0.59865255, 0.49611063, 0.57166948, 0.56290774, 0.68789809])
```

```
clf.fit(train_vectors, train["target"])
```

```
▾ LogisticRegression
LogisticRegression()
```

```
# Fitting a simple Logistic Regression on TFIDF
clf_tfidf = LogisticRegression(C=1.0)
scores = model_selection.cross_val_score(clf_tfidf, train_tfidf, train["target"], cv=5, scoring="f1")
scores
```

```
array([0.57229525, 0.49673203, 0.54277829, 0.46618106, 0.64768683])
```

Naives Bayes Classifer

```
# Fitting a simple Naive Bayes on Counts
clf_NB = MultinomialNB()
scores = model_selection.cross_val_score(clf_NB, train_vectors, train["target"], cv=5, scoring="f1")
scores
```

```
array([0.63149079, 0.60675773, 0.68575519, 0.64341085, 0.72505092])
```

```
clf_NB.fit(train_vectors, train["target"])
```

```
▾ MultinomialNB
MultinomialNB()
```

```
# Fitting a simple Naive Bayes on TFIDF
clf_NB_TFIDF = MultinomialNB()
scores = model_selection.cross_val_score(clf_NB_TFIDF, train_tfidf, train["target"], cv=5, scoring="f1")
scores
```

```
array([0.57590597, 0.57092511, 0.61135371, 0.5962963 , 0.7393745 ])
```