Natural Language Processing Project

1-Problem Statement / Disaster Tweets Classification.

Twitter has become an important communication channel in times of emergency. The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

In this project, we are going to build a machine learning model that predicts which Tweets are about real disasters and which one's aren't. You'll have access to a dataset of more than 7,000 tweets that were hand classified

2-Data Description

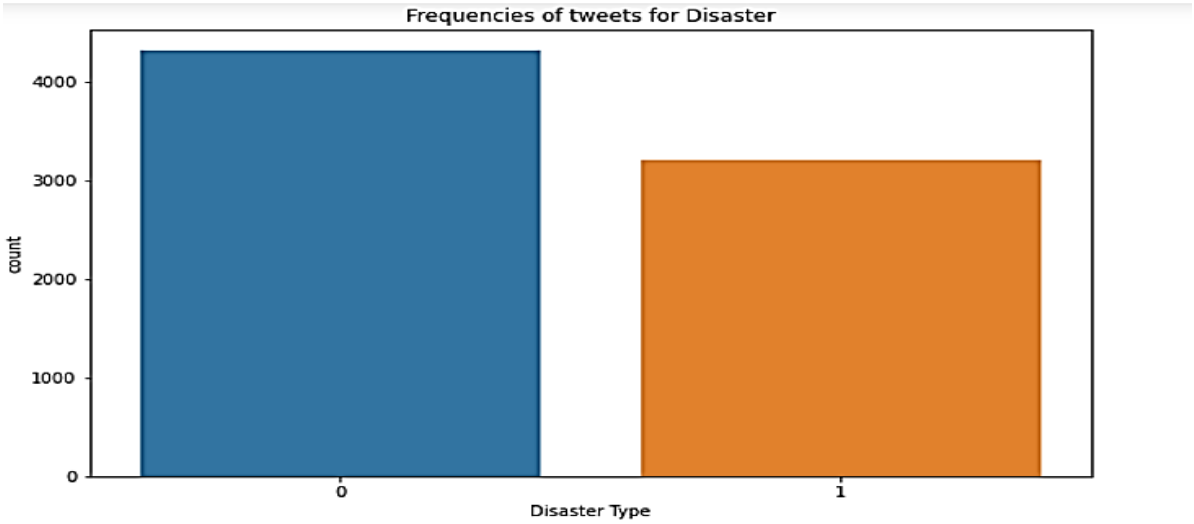
Data columns (total 5 columns), rows = 7613.

Column	**Description**
id	a unique identifier for each tweet (int64)
text	the text of the tweet (object)
location	he location the tweet was sent from (may be blank) (object)
keyword	a particular keyword from the tweet (may be blank) (object)
target	this denotes whether a tweet is about a real disaster (1) or not (0) (int64)

3-Data Visualization

Data visualization helps us to understand the data by see how the data looks like and what kind of correlation is held by the attributes of data and then determine the features correspond to the output.

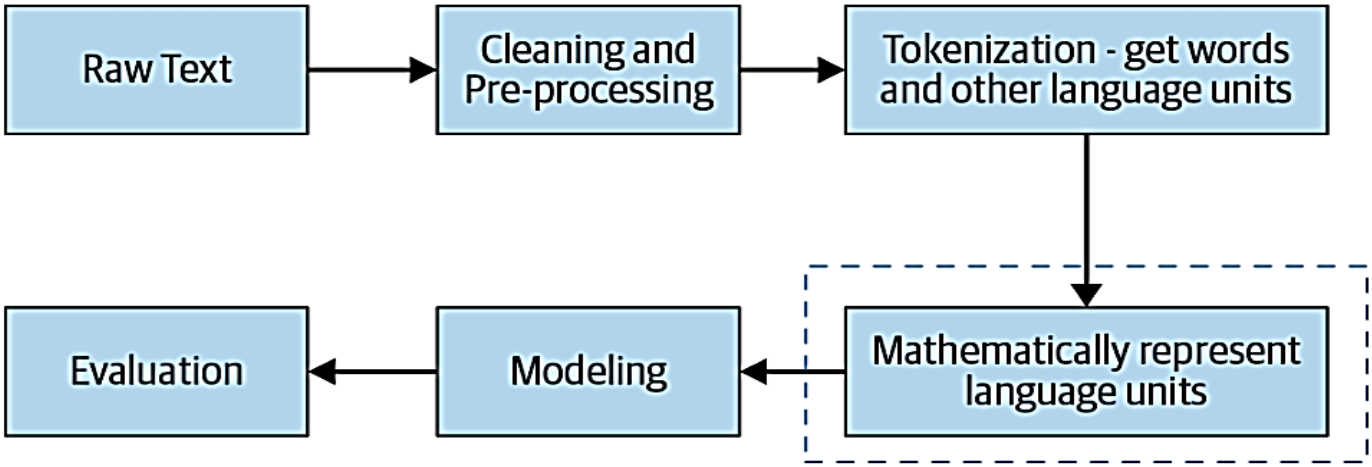
countplot is the most effective tools. A countplot allows us to show Frequencies of Subjects for true news.



4-Data Preprocessing

In Natural Language Processing (NLP) the conversion of raw-text to numerical form is called **Text Representation** and believe me this step is one of the most important steps in the NLP pipeline as if we feed in poor features in ML Model, we will get poor results. In computer science, this is often called “garbage in, garbage out.”

In this project, we will do text-representation to transform a given text into numerical form so that we can use it to build our Text classification model.



In this section, the focus will be on the dotted box in the figure.

But before moving on to the Text representation step first we have to get a cleaned dataset which then has to be preprocessed. In this Project, I will be using only a few basic steps to preprocess the text data:

Data Cleaning:

Loading the data

```
In [3]: Tweet_df = pd.read_csv("C:/Users/Ahmed Elsayed/Desktop/Tweet.csv")
```

```
In [4]: Tweet_df.head()
```

Out[4]:

	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

```
In [6]: Tweet_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        7613 non-null   int64
 1   keyword   7552 non-null   object
 2   location  5080 non-null   object
 3   text      7613 non-null   object
 4   target    7613 non-null   int64
dtypes: int64(2), object(3)
memory usage: 297.5+ KB
```

```
In [7]: Tweet_df['keyword'].isna().sum()
Out[7]: 61
```

```
In [8]: Tweet_df = Tweet_df.drop(['id', 'keyword', 'location'], axis = 1)
```

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

Out[9]: (7613, 2)

In [10]:
```
Tweet_df.columns
```

Out[10]: Index(['text', 'target'], dtype='object')

In [11]:
```
Tweet_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7613 entries, 0 to 7612
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    7613 non-null   object
 1   target  7613 non-null   int64 
dtypes: int64(1), object(1)
memory usage: 119.1+ KB
```

In [12]:
```
Tweet_df[Tweet_df["target"] == 1]["text"].values[0]
```

Out[12]: 'Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all'

In [13]:
```
Tweet_df[Tweet_df["target"] == 1]["text"].values[1]
```

Out[13]: 'Forest fire near La Ronge Sask. Canada'

In [14]:
```
print("Number of duplicates in data : {}".format(len(Tweet_df[Tweet_df.duplicated()])))
```

Number of duplicates in data : 92

[15]:
```
print("Duplicated rows before remove them : ")
Tweet_df[Tweet_df.duplicated(keep=False)].sort_values(by="text").head(8)
```

Duplicated rows before remove them :

[15]:

	text	target
4299	#Allah describes piling up #wealth thinking it...	0
4290	#Allah describes piling up #wealth thinking it...	0
6373	#Bestnaijamade: 16yr old PKK suicide bomber wh...	1
6363	#Bestnaijamade: 16yr old PKK suicide bomber wh...	1
6366	#Bestnaijamade: 16yr old PKK suicide bomber wh...	1
6377	#Bestnaijamade: 16yr old PKK suicide bomber wh...	1
6378	#Bestnaijamade: 16yr old PKK suicide bomber wh...	1
6392	#Bestnaijamade: 16yr old PKK suicide bomber wh...	1

In [16]:
```
#remove duplicated rows
Tweet_df.drop_duplicates(inplace=True)
```

In [17]:
```
print("Number of duplicates in data : {}".format(len(Tweet_df[Tweet_df.duplicated()])))
```

Number of duplicates in data : 0

In [18]:
```
Tweet_df['target'].value_counts()
```

Out[18]:
```
0    4315
1    3206
Name: target, dtype: int64
```

In [17]:
```
Real_Disaster_df = Tweet_df[Tweet_df['target'] == 1]
Real_Disaster_df.head()
```

Out[17]:

	text	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 [18]:
```
Not_Real_Disaster_df = Tweet_df[Tweet_df['target'] == 0]
Not_Real_Disaster_df.head()
```

Out[18]:

	text	target
15	What's up man?	0
16	I love fruits	0
17	Summer is lovely	0

Basic Text Pre-Processing:

Text preprocessing steps include a few essential tasks to further clean the available text data. It includes tasks like:-

1. Stop-Word Removal : In English words like a, an, the, as, in, on, etc. are considered as stop-words so according to our requirements we can remove them to reduce vocabulary size as these words don't have some specific meaning

2. Lower Casing: Convert all words into the lower case because the upper or lower case may not make a difference for the problem. And we are reducing vocabulary size by doing so.

3. Stemming: Stemming refers to the process of removing suffixes and reducing a word to some base form such that all different variants of that word can be represented by the same form (e.g., “walk” and “walking” are both reduced to “walk”).

4. Tokenization: NLP software typically analyzes text by breaking it up into words (tokens) and sentences.

Text Preprocessing

In [19]:
```
# take text and preprocess 'remove stopwords [a, the, and, thus, ... etc] and punctations[,%$ ..etc] and len of text less than 3'
def clean_text(text):
    """
        text: a string
        return: cleaned string
    """
    result = []
    for token in simple_preprocess(text):
        if token not in STOPWORDS and token not in punctation and  len(token) >= 3 :
            token = token.lower()
            result.append(token)
    return " ".join(result)
```

In [20]:
```
Tweet_df['text'] = Tweet_df['text'].map(clean_text)
Tweet_df.head()
```

Out[20]:

	text	target
0	deeds reason earthquake allah forgive	1
1	forest near ronge sask canada	1
2	residents asked shelter place notified officer...	1

Text-Representations - TF-IDF:

TF-IDF, or term frequency-inverse document frequency, addresses this issue. It aims to quantify the importance of a given word relative to other words in the document and in the corpus.

The intuition behind TF-IDF is as follows: if a word w appears many times in a sentence S1 but does not occur much in the rest of the Sentences Sn in the corpus, then the word w must be of great importance to the Sentence S1. The importance of w should increase in proportion to its frequency in S1 (how many times that word occurs in sentence S1), but at the same time, its importance should decrease in proportion to the word’s frequency in other Sentence Sn in the corpus.

Let's consider an example for explanation:

Sentence A = The Car is Driven on the Road
Sentence B = The Truck is Driven on the highway

Computation of TF-IDF scores are shown below

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

5-Train Test Split:

Splitting the data set into a training and a testing set. The test set is the 20% of the original data set

In [22]:
```
X = Tweet_df_shuffled['text']
y = Tweet_df_shuffled['target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state = 42, stratify = y)
```

6- Machine learning Algorithms for Text Classification.

We use Two algorithms:

- ✓ **NB_classifier**
- ✓ **SGD_Classifier**

In [23]:
```
from sklearn.model_selection import cross_val_score
nb_classifier = Pipeline([('vect', CountVectorizer()),
                          ('tfidf', TfidfTransformer()),
                          ('clf', MultinomialNB()),])

nb_classifier.fit(X_train, y_train)

y_pred = nb_classifier.predict(X_test)
print('accuracy {}'.format(accuracy_score(y_pred, y_test)))
```

```
accuracy 0.8019933554817276
```

In [24]:
```
sgd = Pipeline([('vect', CountVectorizer()),
                ('tfidf', TfidfTransformer()),
                ('clf', SGDClassifier(loss='epsilon_insensitive', penalty='l2',alpha=1e-3, random_state=42, max_iter=1000, tol=None)),])


sgd.fit(X_train, y_train)
y_pred = sgd.predict(X_test)
print('accuracy {}'.format(accuracy_score(y_pred, y_test)))
```

```
accuracy 0.7667774086378738
```

7-Evaluating Model Performance

- ✓ All results in The Notebook

8- Another solution

In this solution, we use **Keras model with word embeddings**

- ✓ **Function for making the embeddings:**
  - **We need first to load the large model to get the vectors**

```
In [32]: # Need to load the large model to get the vectors
         nlp = spacy.load('en_core_web_lg')
```

- ✓ **The Model:**

```
]: keras_clf = tf.keras.Sequential([
     tf.keras.layers.Flatten(input_shape=(2, 300)),
     tf.keras.layers.Dropout(0.2),
    # tf.keras.layers.Dense(300, activation='relu'),
    # tf.keras.layers.Dropout(0.5),
     tf.keras.layers.Dense(128, activation='relu'),
     tf.keras.layers.Dropout(0.6),
     tf.keras.layers.Dense(64, activation='relu'),
     tf.keras.layers.Dropout(0.4),
     tf.keras.layers.Dense(2, activation="softmax")
   ])

   keras_clf.compile(optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
             metrics=['accuracy'])
```

- ✓ **The Result:**

```
]: keras_train_preds = keras_model_predict(train_data)
   print(classification_report(y_train, keras_train_preds, target_names=['not_disaster', 'disaster']))
```

2021-12-12 14:41:17.186324: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

	precision	recall	f1-score	support
not_disaster	0.63	0.03	0.05	3904
disaster	0.43	0.98	0.60	2947
accuracy			0.44	6851
macro avg	0.53	0.50	0.33	6851
weighted avg	0.55	0.44	0.29	6851

9- Link of NoteBook deployed on Github

- ✓ https://github.com/sohairkilany/NLP_Project