Pipeline

- Load Dataset
- Clean the Dataset
- Text Pre-Processing
- Text Vectorization
- Build and Train Model
- Plot Classification Report

Import Library

In [1]:
```
# Import Library

import re
import nltk
import numpy as np
import pandas as pd

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

Load Dataset

In [2]:
```
# Load Dataset

def load_dataset(file_name):

    df = pd.read_csv(file_name)

    df = df[["tweet", "class"]]
    df.drop_duplicates(inplace = True)
    df.dropna(inplace = True)

    return df

df = load_dataset("Dataset/labeled_data.csv")

print("Train Shape :", df.shape)

df.head()
```

```
Train Shape : (24783, 2)
```

Out[2]:

	tweet	class
0	!!! RT @mayasolovely: As a woman you shouldn't...	2
1	!!!!! RT @mleew17: boy dats cold...tyga dwn ba...	1
2	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80sbaby...	1
3	!!!!!!!!! RT @C_G_Anderson: @viva_based she lo...	1
4	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you...	1

Text - Preprocessing

In [3]:
```
# Tweet Preprocessing

def pre_processing(tweet: str):

    # Remove Leading Blank Spaces
    tweet = tweet.strip()

    # Lower Case
    tweet = tweet.lower()

    # Remove URLS
    url_pattern = re.compile(r"https?://\S+|www\.\S+")
    tweet = re.sub(url_pattern, "", tweet)

    # Remove UserName
    username_pattern = re.compile(r"@\w+")
    tweet = re.sub(username_pattern, "", tweet)

    # Remove Hashtags
    hashtag_pattern = re.compile(r"#\w+")
    tweet = re.sub(hashtag_pattern, "", tweet)

    # Character normalization // todaaaaay -> today
    tweet = re.sub(r"([a-zA-Z])\1{2,}", r'\1', tweet)

    # Remove Special Characters
    tweet = re.sub(r'[^a-zA-Z\s]', "", tweet)

    # Word Tokenizer
    tweet = nltk.word_tokenize(tweet)

    # Remove Stop Words
    stop_words = set([re.sub(r'[^a-zA-Z\s]', "", word) for word in nltk.corpus.stopwords.words("english")])
    tweet = [word for word in tweet if word not in stop_words]

    # lemmatization
    def get_pos(word):
        tag = nltk.pos_tag([word])[0][1][0].upper()
        tag_dict = {"N": "n", "V": "v", "R": "r", "J": "a"}
        return tag_dict.get(tag, "n")

    lemma = nltk.stem.WordNetLemmatizer()
    tweet = [lemma.lemmatize(word, pos=get_pos(word)) for word in tweet]

    return tweet

df["pre-tweet"] = df["tweet"].apply(pre_processing)

pre_processing("I loveeeee NLP, @rahul_appu, www.rahul_appu.com, #NLP ")
```

Out[3]:
```
['love', 'nlp']
```

Train Test Split

In [4]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df["pre-tweet"].values, df["class"].values, train_size=0.8)
```

Vocab

In [5]:
```
vocab = set()

for words in x_train:
    for word in words:
        vocab.add(word)

print("Vocab Size :", len(vocab))
```

```
Vocab Size : 14569
```

Vectorization

Word2Vec

In [6]:
```
from gensim.models import Word2Vec

g_model = Word2Vec(vector_size=200, window=5, workers=5)
g_model.build_vocab(x_train)
g_model.train(x_train, total_examples=g_model.corpus_count, epochs=500)
```

Out[6]:
```
(52714421, 76588000)
```

In [7]:
```
def in_vocab(word_l):
    for word in word_l:
        if word not in g_model.wv:
            return False
    else:
        return True

train_vec = [g_model.wv[x].sum(axis = 0) if len(x) and in_vocab(x) else np.zeros((200)) for x in x_train]
test_vec  = [g_model.wv[x].sum(axis = 0) if len(x) and in_vocab(x) else np.zeros((200)) for x in x_test]
```

In [8]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter = 1000)
model.fit(train_vec, y_train)

from sklearn.metrics import classification_report, accuracy_score

predict = model.predict(test_vec)
print("Accuracy Score :", accuracy_score(y_test, predict), end='\n\n')
print(classification_report(y_true = y_test, y_pred = predict))
```

```
Accuracy Score : 0.8053258018963082

              precision    recall  f1-score   support

           0       0.44      0.08      0.14       238
           1       0.81      0.98      0.89      3880
           2       0.74      0.20      0.31       839

    accuracy                           0.81      4957
   macro avg       0.66      0.42      0.45      4957
weighted avg       0.78      0.81      0.76      4957
```

fastText

In [9]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df["tweet"].values, df["class"].values, train_size=0.8)

with open("fastText/text.txt", "w") as f:
    for text in x_train:
        f.write(text + "\n")
```

In [10]:
```
import fasttext

f_model = fasttext.train_unsupervised("fastText/text.txt", "cbow")
```

In [11]:
```
train_vec = [f_model[sent] for sent in x_train]
test_vec  = [f_model[sent] for sent in x_test]
```

In [12]:
```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(max_iter = 1000)
model.fit(train_vec, y_train)

from sklearn.metrics import classification_report, accuracy_score

predict = model.predict(test_vec)
print("Accuracy Score :", accuracy_score(y_test, predict), end='\n\n')
print(classification_report(y_true = y_test, y_pred = predict))
```

```
Accuracy Score : 0.7847488400242082

              precision    recall  f1-score   support

           0       0.00      0.00      0.00       297
           1       0.78      1.00      0.88      3866
           2       0.88      0.04      0.07       794

    accuracy                           0.78      4957
   macro avg       0.55      0.34      0.32      4957
weighted avg       0.75      0.78      0.70      4957
```