Social Media and Text Analytics - Industry Assignment 1

Importing Libraries

In [1]:
```
from flask import Flask, request, jsonify, render_template
import pandas as pd
import pickle
import nltk
import re
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
from nltk import sent_tokenize, word_tokenize
from nltk.stem.snowball import SnowballStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\khush\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Loading Dataset

In [2]:
```
app = Flask(__name__)

train = pd.read_csv("train.csv")
test = pd.read_csv("test.csv")
```

Loading Multi Output Classifier Model

In [3]:
```
with open('multi_output_classifier_model.pkl', 'rb') as file:
    loaded_model = pickle.load(file)
# Loading the CountVectorizer
with open('countvectorizer.pkl', 'rb') as file:
    loaded_countvectorizer = pickle.load(file)
# Loading the TfidfTransformer
with open('tfidftransformer.pkl', 'rb') as file:
    loaded_tfidftransformer = pickle.load(file)
```

Function to Preprocess Text Data

In [4]:
```
def preprocess_text(text):
    text = text.lower()
    text = re.sub('[^a-zA-Z]', ' ', text)
    tokens = nltk.word_tokenize(text)
    stop_words = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stop_words]
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]
    processed_text = ' '.join(tokens)
    return processed_text
```

Function to Make Predictions using Loaded Model

In [5]:
```
def make_prediction(text):
    preprocessed_text = preprocess_text(text)
    text_cv = loaded_countvectorizer.transform([preprocessed_text])
    text_tf = loaded_tfidftransformer.transform(text_cv)
    prediction = loaded_model.predict(text_tf)
    return prediction
```

In [ ]:
```
@app.route('/', methods=['GET'])
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if 'text' in request.form:
        user_text = request.form['text']
```

```
        prediction = make_prediction(user_text)
        return render_template('predict.html', prediction=prediction)
    else:
        return jsonify({'message': 'Text not provided'})

if __name__ == '__main__':
    app.run(debug=True)
```