**SENTIMENT ANALYSIS FOR MARKETING**

Phase-3

# 1.Acquire the Dataset:

Obtain the dataset containing text data and corresponding sentiment labels. Datasets for sentiment analysis can be collected from various sources, such as social media, customer reviews, or surveys. Ensure that the dataset has a label or target variable (e.g., positive, negative, neutral) for each text sample.

# 2.Import Libraries:

Import the necessary libraries, including pandas for data manipulation, scikit-learn for preprocessing and machine learning, and any other specific libraries you may need.

## python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

from sklearn.preprocessing import LabelEncoder

# 1.Load the Dataset:

Read the dataset into a pandas DataFrame.

## python

data = pd.read\_csv('marketing\_sentiment\_dataset.csv')

# 1.Data Exploration:

Explore the dataset to understand its structure and contents. You can use functions like head(), info(), and describe() to get a sense of the data.

## python

print(data.head())

print(data.info())

# 1.Text Preprocessing:

Clean and preprocess the text data. This may include:

\*Removing special characters, punctuation, and numbers.

\*Converting text to lowercase.

\*Tokenization (splitting text into individual words or tokens).

\*Removing stop words (common words like "the," "and," "is" that don't provide much information).

\*Lemmatization or stemming to reduce words to their root form.

Here's an example of some text preprocessing using the nltk library:

## python

import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer

nltk.download('stopwords')

nltk.download('wordnet')

stop\_words = set(stopwords.words('english'))

lemmatizer = WordNetLemmatizer()

def preprocess\_text(text):

text = text.lower()

words = text.split()

words = [word for word in words if word not in stop\_words]

words = [lemmatizer.lemmatize(word) for word in words]

return ' '.join(words)

data['text'] = data['text'].apply(preprocess\_text)

# 1.Label Encoding:

If your sentiment labels are in text format (e.g., "positive," "negative," "neutral"), you'll need to encode them into numerical values. You can use LabelEncoder from scikit-learn for this purpose.

## python

label\_encoder = LabelEncoder()

data['sentiment'] = label\_encoder.fit\_transform(data['sentiment'])

# 1.Split the Data:

Split your dataset into training and testing sets. This allows you to evaluate the model's performance later.

## python

X = data['text']

y = data['sentiment']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 1.Vectorization:

Convert the text data into numerical form by using techniques like Count Vectorization or TF-IDF Vectorization. These methods convert text into feature vectors that machine learning models can understand.

## python

tfidf\_vectorizer = TfidfVectorizer(max\_features=5000)

X\_train\_tfidf = tfidf\_vectorizer.fit\_transform(X\_train)

X\_test\_tfidf = tfidf\_vectorizer.transform(X\_test)

# 1.Model Training:

Train a sentiment analysis model using your vectorized data. You can use various machine learning algorithms or deep learning models for this task.

# 2.Model Evaluation:

Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score on the test data.

# 3.Inference:

Use the trained model to perform sentiment analysis on new marketing data to gain insights and make data-driven decisions.