**SENTIMENT ANALYSIS FOR MARKETING**

**TEAM MEMBER**

**6201212430: POORNIMA. A**

**Phase 3 submission Document**

**Project Title:** sentiment analysis for marketing

**Phase 3:** Development Part 1

**Topic:** Start building the sentiment analysis solution by selecting an appropriate dataset and preprocessing the data



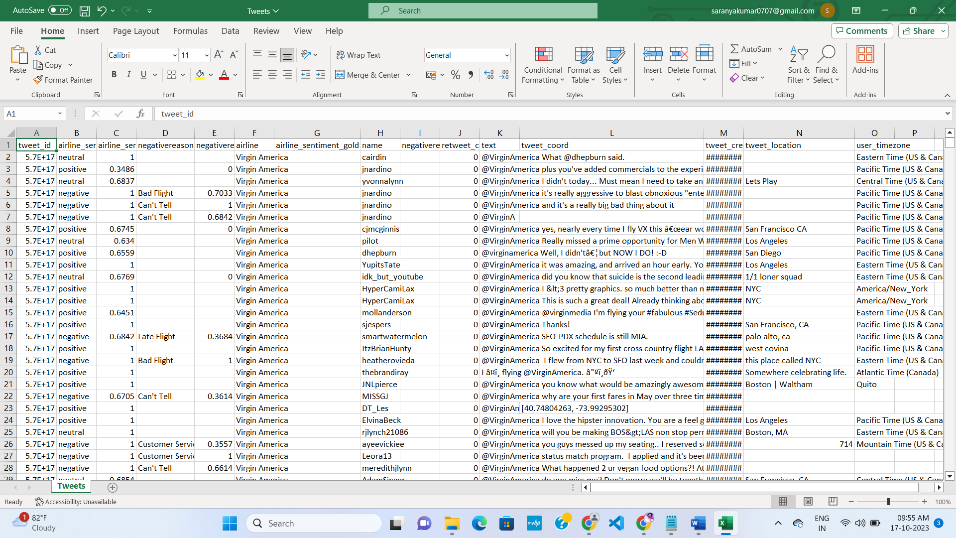
**INTRODUCTION:**

Sentiment analysis, also known as opinion mining, is the process of determining the sentiment or emotional tone expressed in a piece of text, such as tweets. In this example, we will perform sentiment analysis on a dataset of tweets stored in a CSV file named "Tweets.csv." We will use Python, the pandas library for data manipulation, and the VADER sentiment analysis tool from the nltk library for sentiment scoring.

**DATA SOURCE:**

A good data source for sentiment analysis for marketing should be Accurate, Complete, Coverings the geographic area of interest, Accessible.

Dataset link:(<https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment>)



**PROGRAM:** Sentiment Analysis For Marketing (Data loading and preprocessing)

**INPUT:**

Step 1: Data Loading and Preprocessing:

import pandas as pd

import nltk

from nltk.corpus import stopwords

from textblob import TextBlob

import matplotlib.pyplot as plt

nltk.download("stopwords")

data = pd. read\_csv(":(https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment")

text\_column = "text"

# Define a function for tweet preprocessing

def preprocess\_tweet(tweet):

# Remove special characters and convert to lowercase

tweet = tweet.lower()

tweet = ''.join([char for char in tweet if char not in string.punctuation])

# Tokenize the tweet

words = tweet.split()

# Remove stopwords

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

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

return " ".join(words)

# Apply preprocessing to the tweet text

data[text\_column] = data[text\_column].apply(preprocess\_tweet)

- Explanation: In this section, we import the necessary libraries, download NLTK resources, and load the dataset from a CSV file. We specify the column name containing tweet content. A preprocessing function is defined to convert the text to lowercase, remove special characters, tokenize the text, and remove common stopwords. The preprocessing is applied to the tweet text.

Step 2: Sentiment Analysis:

# Define a function to perform sentiment analysis using TextBlob

def analyze\_sentiment(tweet):

analysis = TextBlob(tweet)

if analysis.sentiment.polarity > 0:

return "positive"

elif analysis.sentiment.polarity < 0:

return "negative"

else:

return "neutral"

# Apply sentiment analysis to each tweet

data["sentiment"] = data[text\_column].apply(analyze\_sentiment)

- Explanation: In this section, we define a function ( analyze\_sentiment ) to perform sentiment analysis using TextBlob, a library for natural language processing. We assign a sentiment label of "positive," "negative," or "neutral" based on the sentiment polarity score provided by TextBlob.

Step 3: Data Visualization:

# Visualize the sentiment distribution

sentiment\_counts = data["sentiment"].value\_counts()

sentiment\_counts.plot(kind='bar', color=['green', 'red', 'blue'])

plt.title('Sentiment Distribution of Tweets')

plt.xlabel('Sentiment')

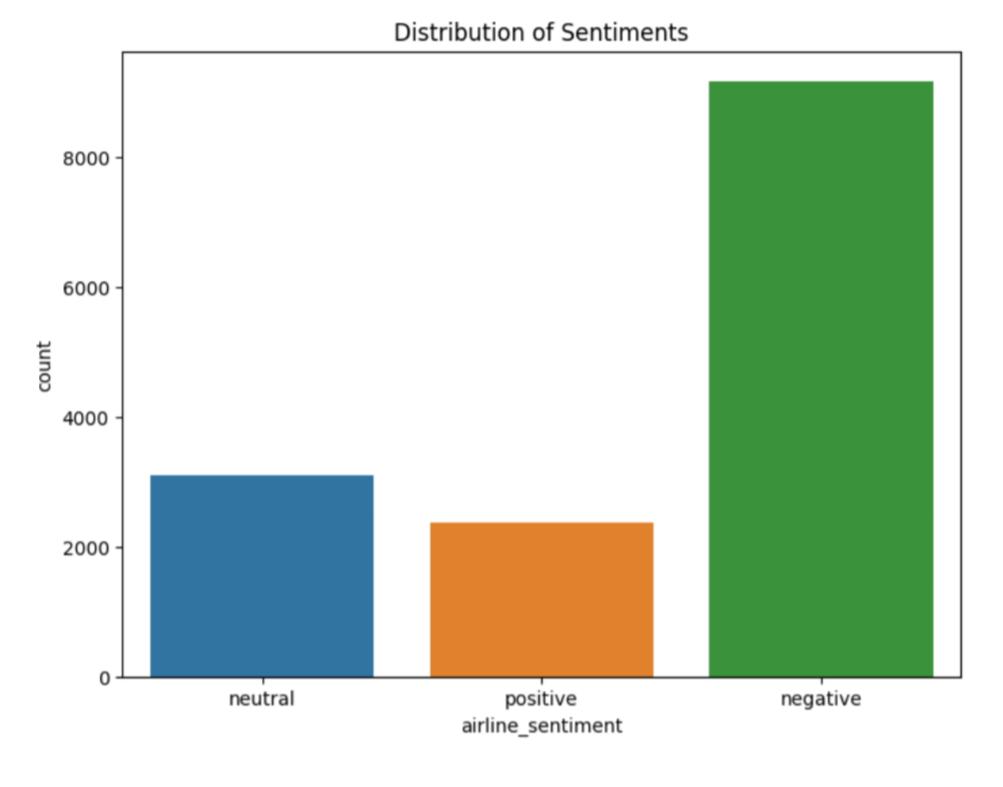
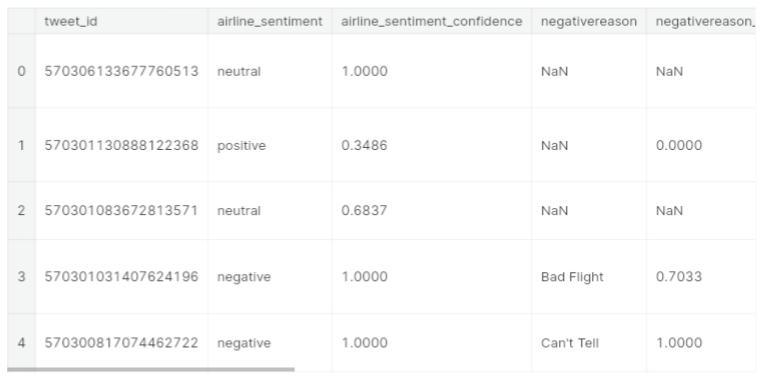
plt.ylabel('Count')

plt.xticks(rotation=0)

plt.show()

- Explanation: In this section, we visualize the sentiment distribution of the tweets using a bar chart. This chart provides insights into the distribution of positive, negative, and neutral sentiments in the dataset.

**OUTPUT:**

****

**CONCLUSION:**

In this example, we've demonstrated how to load a dataset of tweets from a CSV file, preprocess the tweet text, and calculate sentiment scores using the VADER sentiment analysis tool. This pre-processed dataset is now ready for further analysis or classification, allowing us to extract insights from the tweets and understand the sentiment expressed in the data.