Sentiment Analysis Project.

Business Overview

Introduction

In today's digital age, social media platforms such as Twitter have become powerful sources of real-time customer feedback and opinions. Understanding the sentiment expressed by customers toward specific brands and products is essential for businesses to make informed decisions, enhance customer satisfaction, and maintain a positive brand reputation.

The goal of this project aims to develop a sentimental analysis model specifically tailored to analyze Twitter data related to Google, Apple, and other products.

Business Problem Statement

As a consulting firm, Twitter has assigned us the task of building a model that can rate the sentiment of a Tweet based on its content that can correctly categorize Twitter sentiment about Apple and Google products into positive, negative, or neutral categories and gain valuable insights into public perception, that will be used for informed decision-making in business strategies and customer satisfaction enterprise.

Data Understanding

For this project, we utilized a dataset from CrowdFlower via Data.world, consisting of approximately 9,000 tweet sentiments about Apple and Google products. The dataset includes columns such as tweet_text, emotion_in_tweet_is_directed_at, and

is_there_an_emotion_directed_at_a_brand_or_product. Our objective is to develop a sentiment analysis model that should accurately classify tweets into positive, negative, or neutral sentiment categories.

Objectives for the project:

1. To develop a binary classifier that can classify tweets into positive or negative sentiment categories. This will serve as a proof of concept and provide a foundation for further analysis. The classifier will be a Logistic regression model and the benchmark accuracy will be 85%.
2. To expand to a multiclass classifier, thereby incorporating the neutral tweets to create a multiclass classifier that can accurately classify tweets as positive, negative, or neutral. This will provide a more comprehensive sentiment analysis of the tweets. The classifier will be a XGBoost model and MultinomialNB the benchmark accuracy will be 70%.
3. To compare sentiment between Apple and Google products by analyzing the sentiment distribution of tweets mentioning Apple, Google, and other products.

Data Preparation

To prepare the data, we performed various preprocessing steps.

These included; removing duplicates, handling missing values by filling them with "none" for the emotion_in_tweet_is_directed_at column, and dropping the row with a missing tweet_text. We also employed tokenization, lowercase conversion, stopword removal, and lemmatization to refine the text data.

Modeling

The packages/libraries utilized for data preparation and analysis included NLTK, sklearn, and pandas.

- NLTK was used for tokenization, stopword removal, and lemmatization.
- Sklearn's CountVectorizer was employed for vectorization.
- Pandas was used for data manipulation and handling missing values.

For modeling, we employed logistic regression as the binary classifier to classify tweets into positive or negative sentiment categories. As a benchmark, we aimed for an accuracy of 70%.

In addition, we expanded the model to a multiclass classifier to include the neutral sentiment category.

Testing Model

Evaluation

To evaluate the performance of the final model, we employed accuracy. This metric provided insight into the model's ability to correctly classify instances and predict positive and negative sentiment.

Accuracy is a commonly used metric for evaluating sentiment analysis models because it provides a straightforward measure of the overall performance. In sentiment analysis, the primary goal is to classify a given text or document into positive, negative, or neutral sentiment categories accurately. Accuracy, in this context, represents the proportion of correctly classified instances compared to the total number of instances. It gives a clear indication of how well a sentiment analysis model is able to predict sentiment labels correctly across the entire dataset.

In conclusion, the sentiment analysis model developed using logistic regression demonstrated satisfactory performance in classifying tweet sentiments. The pre-processing steps and modeling choices contributed to accurate predictions and meaningful insights. However, limitations such as missing values and data quality issues should be considered when interpreting the results. Recommendations include further refining the model and expanding the dataset for improved sentiment analysis accuracy.

Deployment

We deployed the model using the framework STREAMLIT. The application takes a sentence and returns the sentiment. Link to web app https://moringaphase4project.streamlit.app/ (https://moringaphase4project.streamlit.app/)

Data understanding

Load Libraries

We will start by importing all libraries used throughout the notebook.

In [1]:

```
#loading libraries

import re
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import string
import warnings
warnings.filterwarnings(action='ignore', category=FutureWarning)

import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from sklearn.preprocessing import LabelEncoder
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectori
from nltk import FreqDist

# modelling libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confus
from sklearn import metrics
from sklearn.naive_bayes import MultinomialNB
import xgboost as xgb

#imbalance libraries
from imblearn.over_sampling import RandomOverSampler
from imblearn.under_sampling import RandomUnderSampler
import xgboost as xgb
from sklearn.metrics import accuracy_score
```

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

Loading the data

In [2]:
```
# loading and previewing data
data = pd.read_csv('data.csv', encoding='ISO-8859-1')
data.head()
```

Out[2]:

	tweet_text	emotion_in_tweet_is_directed_at	is_there_an_emotion_directed_at_a_brand_or_pro
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative em
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive em
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive em
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative em
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive em

Statistic Describing the dataset

In [3]:
```
# class to describe dataset

class Describer:

    # initialize object
    def __init__(self, df):
        self.df = df

    # method to check shape of data
    def shape(self):
        out = print(f"The DataFrame has:\n\t* {self.df.shape[0]} rows\n\t
        return out

    # method to check info on dataset
    def data_info(self):
        out = print(self.df.info(), '\n')
        return out

    # method to describe numerical columns
    def data_describe(self):
        out = self.df.describe()
        return out
```

In [4]:
```
# creating an instance of the class describer
describe_df = Describer(data)

# lets view the shape of the data
describe_df.shape()
```

```
The DataFrame has:
        * 9093 rows
        * 3 columns
```

In [5]:
```
# lets print summary infomation on the dataset
print('Summary infomation on dataset')
print('---------------------------------------------')
describe_df.data_info()
```

```
Summary infomation on dataset
---------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9093 entries, 0 to 9092
Data columns (total 3 columns):
 #   Column                                              Non-Null Count  Dty
pe
---  ------                                              --------------  ---
--
 0   tweet_text                                          9092 non-null   obj
ect
 1   emotion_in_tweet_is_directed_at                     3291 non-null   obj
ect
 2   is_there_an_emotion_directed_at_a_brand_or_product  9093 non-null   obj
ect
dtypes: object(3)
memory usage: 213.2+ KB
None
```

Summary of Data Understanding

- The dataset has 3 columns namely:tweet_text, emotion_in_tweet_is_directed_at ,is_there_an_emotion_directed_at_a_brand_or_product
- The dataset has 9093 rows
- All the dataset's columns datatypes are objects

3. Data Preparation

Introduction

We will be preparing our data for analysis by checking for attributes such as;

> Missing values

> Duplicates

> Other inconsistencies as computed below

Type *Markdown* and LaTeX: α^2

Missing values

In [6]:

```
# function to identify missing values

# identify missing
def identify_missing_values(data):
    """Identify is the data has missing values"""
    # identify if data has missing values(data.isnull().any())
    # empty dict to store missing values
    missing = []
    for i in data.isnull().any():
        # add the bool values to empty list
        missing.append(i)
    # covert list to set (if data has missing value, the list should have
    missing_set = set(missing)
    if (len(missing_set) == 1):
        out = print("The Data has no missing values")
    else:
        out = print("The Data has missing values.")

    return out

identify_missing_values(data)
```

```
The Data has missing values.
```

In [7]:
```
# function to display missing values

def missing_values(data):
    """A simple function to identify data has missing values"""
    # identify the total missing values per column
    # sort in order
    miss = data.isnull().sum().sort_values(ascending = False)

    # calculate percentage of the missing values
    percentage_miss = (data.isnull().sum() / len(data)).sort_values(ascen

    # store in a dataframe
    missing = pd.DataFrame({"Missing Values": miss, "Percentage(%)": perc

    # remove values that are missing
    missing.drop(missing[missing["Percentage(%)"] == 0].index, inplace =

    return missing


missing_values(data)

```

Out[7]:

	Missing Values	Percentage(%)
emotion_in_tweet_is_directed_at	5802	0.638073
tweet_text	1	0.000110

In [8]:
```
# dropping missing row in tweet_text

data.dropna(axis=0,subset=['tweet_text'],inplace=True)
```

In [9]:
```
# confirmation
data.isna().any()
```

Out[9]:
```
tweet_text                                            False
emotion_in_tweet_is_directed_at                        True
is_there_an_emotion_directed_at_a_brand_or_product    False
dtype: bool
```

In [10]:
```
# replacing null values in recipient with none
data['emotion_in_tweet_is_directed_at'].fillna('none', inplace =True)
```

In [11]:
```
#confirming we replaced the 'nan' with 'none'
data['emotion_in_tweet_is_directed_at'].value_counts()
```

Out[11]:
```
none                                5801
iPad                                 946
Apple                                661
iPad or iPhone App                   470
Google                               430
iPhone                               297
Other Google product or service      293
Android App                           81
Android                               78
Other Apple product or service        35
Name: emotion_in_tweet_is_directed_at, dtype: int64
```

In [12]:
```
data.isna().any()
```

Out[12]:
```
tweet_text                                          False
emotion_in_tweet_is_directed_at                     False
is_there_an_emotion_directed_at_a_brand_or_product  False
dtype: bool
```

Justification

- We discovered that there is `0.01%` missing values in **tweet_text** column and `63.81%` missing values in **emotion_in_tweet_is_directed_at** column.
- We replaced the missing values in the `emotion tweet is directed at column` with none for computational purposes and dropped the missing valuesin the `tweet text` column

Duplicates

In [13]:
```
# checking for duplicates

# Duplicated entries
def identify_duplicates(data):
    """Simple function to identify any duplicates"""
    # identify the duplicates (dataframename.duplicated() , can add .sum(
    # empty list to store Bool results from duplicated
    duplicates = []
    for i in data.duplicated():
        duplicates.append(i)
    # identify if there is any duplicates. (If there is any we expect a T
    duplicates_set = set(duplicates)
    if (len(duplicates_set) == 1):
        print("The Data has no duplicates")
    else:
        no_true = 0
        for val in duplicates:
            if (val == True):
                no_true += 1
        # percentage of the data represented by duplicates
        duplicates_percentage = np.round(((no_true / len(data)) * 100), 3
        print(f"The Data has {no_true} duplicated rows.\nThis constitutes



identify_duplicates(data)
```

```
The Data has 22 duplicated rows.
This constitutes 0.242% of the data set.
```

In [14]:
```
data.duplicated().sum()
```

Out[14]:
```
22
```

In [15]:
```
# handling the duplicates
def remove_duplicated_rows(data):
    """Simple Function to remove duplicated rows"""
    data.drop_duplicates(subset=None, keep="first", inplace=True)
    # confirm if the duplicated rows have been removed
    confirm = identify_duplicates(data)

    return confirm


remove_duplicated_rows(data)
```

```
The Data has no duplicates
```

Summary

> We found out that the **tweet text** column had 22 duplicates (`0.242%`) and went ahead and dropped them.

Renaming long column names

In [16]:
```
# renaming the columns
data = data.rename(columns={'emotion_in_tweet_is_directed_at':'recipient'
                    'is_there_an_emotion_directed_at_a_brand_or_product':
data.head()
```

Out[16]:

	tweet_text	recipient	emotion
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion
3	@sxsw I hope this year's festival isn't as cra...	iPad or iPhone App	Negative emotion
4	@sxtxstate great stuff on Fri #SXSW: Marissa M...	Google	Positive emotion

Justification

We renamed the column `emotion_in_tweet_is_directed_at` to `recipient` and the column `is_there_an_emotion_directed_at_a_brand_or_product` to `emotion` to enhance clarity and facilitate their usage in our subsequent computation

4. Exploratory Data Analysis (EDA)

Introduction

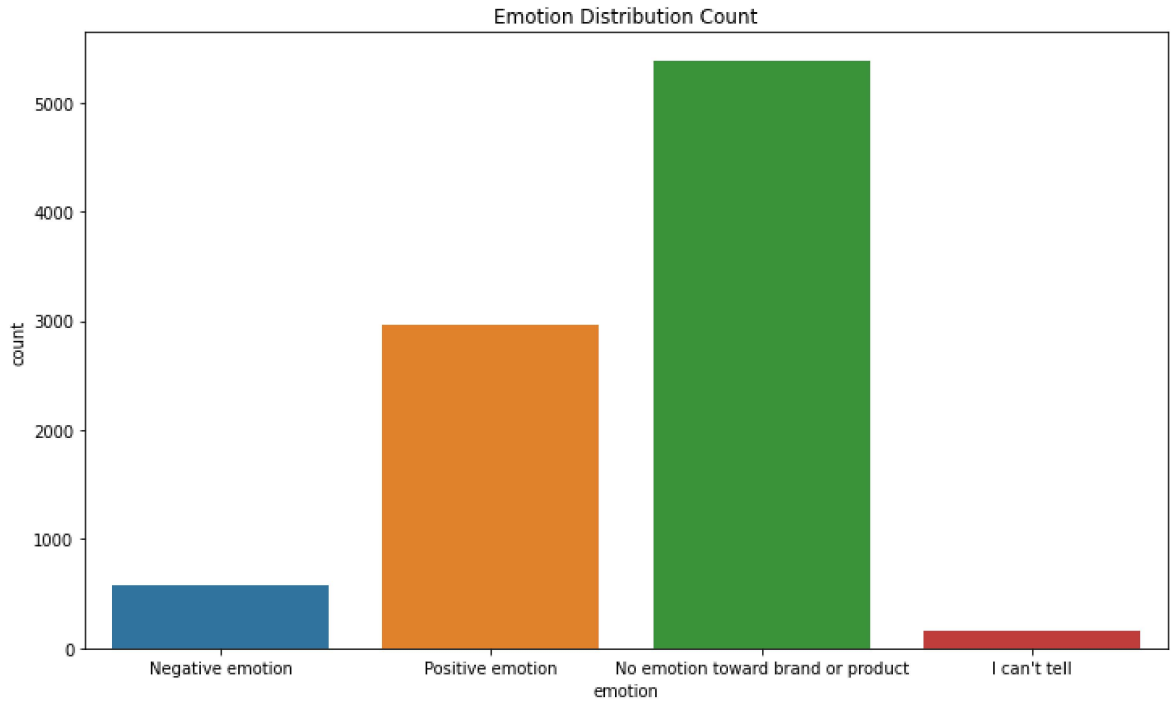
We will conduct Univariate and Bivariate analysis of the sentiments and create visualizations to see how they relate with each other and individually.

Univariate analysis

We will plot a count plot to show the distribution of the column 'emotion'

In [17]:
```
# checking the distribution of emotions
percentage = data['emotion'].value_counts(normalize=True)
print(percentage * 100)

# count plot
plt.figure(figsize=(12, 7))
sns.countplot(data['emotion'])
plt.title('Emotion Distribution Count')
plt.show();
```

```
No emotion toward brand or product    59.261301
Positive emotion                      32.745314
Negative emotion                       6.273429
I can't tell                           1.719956
Name: emotion, dtype: float64
```



Observation

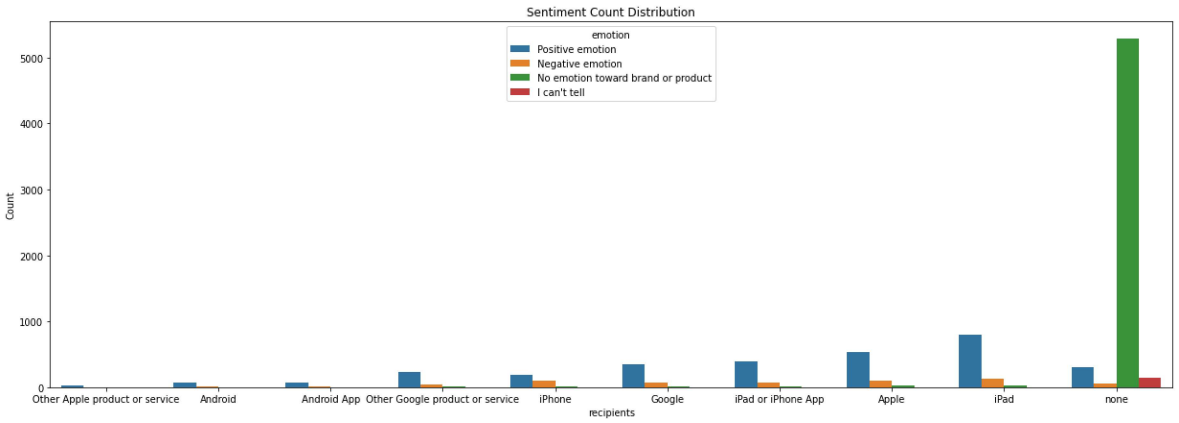
We see that majority of people had `No emotion toward brand or product` at `59.26%` followed by people with `Positive emotions` at `32.75%`

`I can't tell` makes up less than `2%` of our dataset, and doesn't offer much more information in the way of word significance than the tweets labeled No emotion toward brand or product.

Bivariate analysis

We plot a bar plot to visualize the distribution of sentiments'emotion' relative to the recipient

In [18]:
```
# let's see sentiment distribution product-wise
recipient_group = data.groupby('recipient')['emotion'].value_counts().res
# Calculate the ascending order of states based on churn count
ascending_order = recipient_group.groupby('recipient')['count'].sum().sor

plt.figure(figsize=(21,7))
sns.barplot(x='recipient', y='count', hue='emotion', data=recipient_group
plt.xlabel('recipients')
plt.ylabel('Count')
plt.title('Sentiment Count Distribution')
plt.show()
```



Interpretation

From the graph above `Ipad` had the `highest positive emotions` followed by `Apple`

From the observation above since we similar product brand names we will map products to their brand

Mapping products to Brands

In [19]:
```
#What are value counts of recipient column
data['recipient'].value_counts()
```

Out[19]:
```
none                                5788
iPad                                 945
Apple                                659
iPad or iPhone App                   469
Google                               428
iPhone                               296
Other Google product or service      293
Android App                           80
Android                               77
Other Apple product or service        35
Name: recipient, dtype: int64
```

In [20]:
```
# overall brand distribution
# feature engineering
data['brand'] = data['recipient'].map({'Other Apple product or service':
                                       'Android': 'Google',
                                       'Android App': 'Google',
                                       'Other Google product or servi
                                       'iPhone': 'Apple',
                                       'Google':'Google',
                                       'iPad or iPhone App': 'Apple',
                                       'Apple':'Apple',
                                       'iPad':'Apple',
                                       'none': 'none'})
```

In [21]:
```
# confirm brand column
data.head(3)
```

Out[21]:

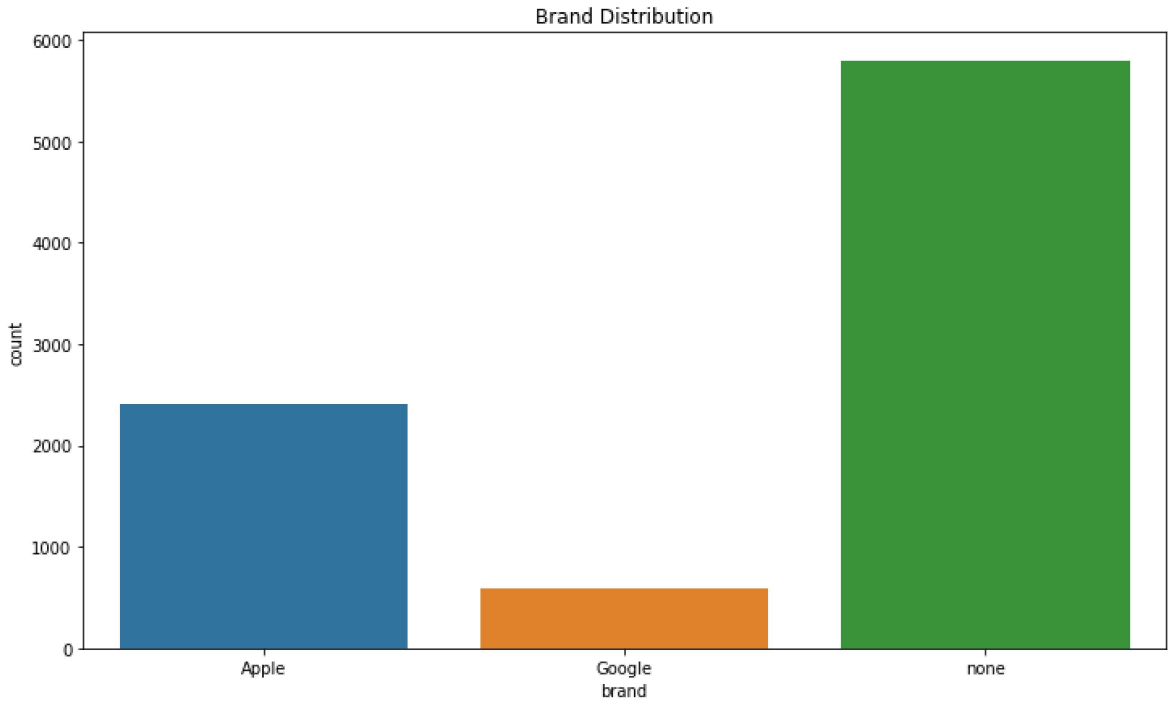
	tweet_text	recipient	emotion	brand
0	.@wesley83 I have a 3G iPhone. After 3 hrs twe...	iPhone	Negative emotion	Apple
1	@jessedee Know about @fludapp ? Awesome iPad/i...	iPad or iPhone App	Positive emotion	Apple
2	@swonderlin Can not wait for #iPad 2 also. The...	iPad	Positive emotion	Apple

Plot brand Distributions

Univariate Analysis

In [22]:
```
# checking the distribution of brand sentiment
percentage = data['brand'].value_counts(normalize=True)
print(percentage * 100)

# count plot
plt.figure(figsize=(12, 7))
sns.countplot(data['brand'])
plt.title('Brand Distribution')
plt.show();
```

```
none      65.945084
Apple     27.389769
Google     6.665148
Name: brand, dtype: float64
```



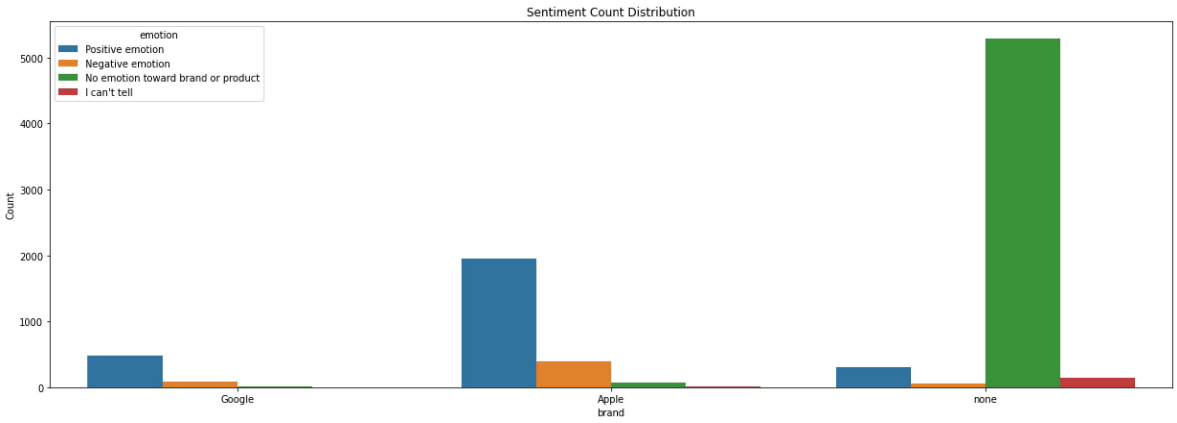
Insight

From the graph above Majority of people had not specified the brand, and for those who did, `Apple` had the highest brand feedback as compared to `Google`

Bivariate analysis

We plot a bar plot to visualize the distribution of sentiments ***'emotion'*** relative to the ***Brand***

In [23]:
```
# lets see see sentiment distribution brand-wise
brand_group = data.groupby('brand')['emotion'].value_counts().reset_index
# Calculate the ascending order of states based on churn count
ascending_order = brand_group.groupby('brand')['count'].sum().sort_values

plt.figure(figsize=(21,7))
sns.barplot(x='brand', y='count', hue='emotion', data=brand_group, order=
plt.xlabel('brand')
plt.ylabel('Count')
plt.title('Sentiment Count Distribution')
plt.show()
```



Interpretation

From the graph above `Apple` had the highest `positive emotions` compared to `Google`

5. Preprocessing

Introduction

In this section we will;

- Convert the tweet text to lower case
- Remove html tags
- Remove the Url
- Expand the contractions
- Remove the punctuations
- Tokenize
- Remove stopwords
- Lemmatize the tweet

So as to prepare the data for modeling.

In [24]:
```
# what are the columns?
data.columns
```

Out[24]:
```
Index(['tweet_text', 'recipient', 'emotion', 'brand'], dtype='object')
```

Changing text to lowercase

In [25]:
```
# normalizing the data to lowecase
data['tweet_text'] = data['tweet_text'].str.lower()
data.head(2)
```

Out[25]:

	tweet_text	recipient	emotion	brand
0	.@wesley83 i have a 3g iphone. after 3 hrs twe...	iPhone	Negative emotion	Apple
1	@jessedee know about @fludapp ? awesome ipad/i...	iPad or iPhone App	Positive emotion	Apple

Removing Html tags and URL

In [26]:
```
#Identifying Html tag
data['tweet_text'][14]
```

Out[26]:
```
'great #sxsw ipad app from @madebymany: http://tinyurl.com/4nqv92l' (http://tinyurl.com/4nqv92l')
```

In [27]:
```
# removing the html tags
def remove_html(review):
    pattern = re.compile('<.*?>')
    return pattern.sub(r'', review)

data['tweet_text'] = data['tweet_text'].apply(remove_html)
```

In [28]:
```
# removing URL and @ sign
def preprocess_text_removingq_URLand_atsign(text):
    # Remove URLs
    clean_text = re.sub(r"http\S+|www\S+|https\S+", "", text)
    text = re.sub(r'@[^\s]+', 'user', clean_text)
    # Other preprocessing steps like removing punctuation, converting to
    # ...
    return text

data['tweet_text'] = data['tweet_text'].apply(preprocess_text_removingq_U
```

In [29]:
```
#lets confirm the url has been replaced with an empty
data['tweet_text'][14]
```

Out[29]:
```
'great #sxsw ipad app from user '
```

In [30]:
```
1 data.head()
```

Out[30]:

	tweet_text	recipient	emotion	brand
0	.user i have a 3g iphone. after 3 hrs tweeting...	iPhone	Negative emotion	Apple
1	user know about user ? awesome ipad/iphone app...	iPad or iPhone App	Positive emotion	Apple
2	user can not wait for #ipad 2 also. they shoul...	iPad	Positive emotion	Apple
3	user i hope this year's festival isn't as cras...	iPad or iPhone App	Negative emotion	Apple
4	user great stuff on fri #sxsw: marissa mayer (...	Google	Positive emotion	Google

Expanding Words

We convert abbreviated words to their full form

In [31]:
```
1  # expanding the contractions (is-nots)
2  import contractions
3  #from nltk.stem import WordNetLemmatizer
4
5  def expand(text):
6      # Expand contractions
7      expanded_text = contractions.fix(text)
8
9      return expanded_text
10
11 data['tweet_text'] = data['tweet_text'].apply(expand)
```

In [32]:
```
1 data.tweet_text[3]
```

Out[32]:
```
"user i hope this year's festival is not as crashy as this year's iphone ap
p. #sxsw"
```

Removing Punctuations

In [33]:
```
1 # punctuation
2 exclude = string.punctuation
3 print(exclude)
```

```
!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~
```

In [34]:
```
1 # remove punctuations
2 def remove_punctuations(tweet):
3     return tweet.translate(str.maketrans('','',exclude))
```

In [35]:
```
data['tweet_text']=data['tweet_text'].apply(remove_punctuations)
data.head()
```

Out[35]:

	tweet_text	recipient	emotion	brand
0	user i have a 3g iphone after 3 hrs tweeting a...	iPhone	Negative emotion	Apple
1	user know about user awesome ipadiphone app t...	iPad or iPhone App	Positive emotion	Apple
2	user can not wait for ipad 2 also they should ...	iPad	Positive emotion	Apple
3	user i hope this years festival is not as cras...	iPad or iPhone App	Negative emotion	Apple
4	user great stuff on fri sxsw marissa mayer goo...	Google	Positive emotion	Google

In [36]:
```
data.tweet_text[20]
```

Out[36]:
```
'need to buy an ipad2 while i am in austin at sxsw not sure if i will need t
o q up at an austin apple store'
```

Tokenization and Removing Stopwords

In [37]:
```
# tokenize the tweets
def tokenize_text(tweet):
    return word_tokenize(tweet)

data['tweet_text'] = data['tweet_text'].apply(tokenize_text)
data.head()
```

Out[37]:

	tweet_text	recipient	emotion	brand
0	[user, i, have, a, 3g, iphone, after, 3, hrs, ...	iPhone	Negative emotion	Apple
1	[user, know, about, user, awesome, ipadiphone,...	iPad or iPhone App	Positive emotion	Apple
2	[user, can, not, wait, for, ipad, 2, also, the...	iPad	Positive emotion	Apple
3	[user, i, hope, this, years, festival, is, not...	iPad or iPhone App	Negative emotion	Apple
4	[user, great, stuff, on, fri, sxsw, marissa, m...	Google	Positive emotion	Google

In [38]:
```
# removing stop words
stop_words = set(stopwords.words('english'))

def remove_stopwords(tweet):
    # Use list comprehension for efficient list creation
    new_tweet = [word for word in tweet if word not in stop_words]
    return " ".join(new_tweet)

# Apply the function to the 'review' column
data['tweet_text'] = data['tweet_text'].apply(remove_stopwords)
data.head()
```

Out[38]:

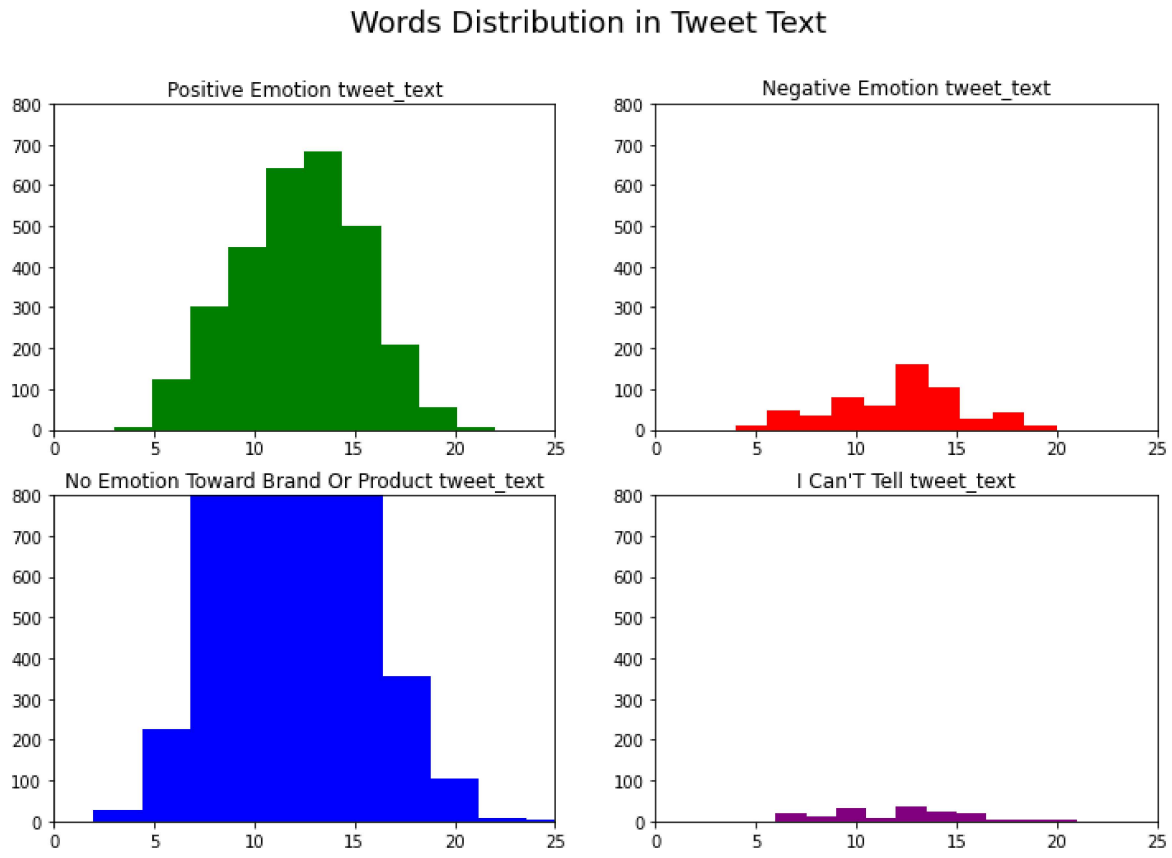
	tweet_text	recipient	emotion	brand
0	user 3g iphone 3 hrs tweeting riseaustin dead ...	iPhone	Negative emotion	Apple
1	user know user awesome ipadiphone app likely a...	iPad or iPhone App	Positive emotion	Apple
2	user wait ipad 2 also sale sxsw	iPad	Positive emotion	Apple
3	user hope years festival crashy years iphone a...	iPad or iPhone App	Negative emotion	Apple
4	user great stuff fri sxsw marissa mayer google...	Google	Positive emotion	Google

In [39]:
```
data.emotion.value_counts()
```

Out[39]:
```
No emotion toward brand or product    5375
Positive emotion                      2970
Negative emotion                       569
I can't tell                           156
Name: emotion, dtype: int64
```

Words Distribution in Tweet Text

In [40]:
```
def generate_review_hist_plot(data, label, color, ax):
    tweet_len = data[data['emotion']==label].tweet_text.str.split().map(l
    ax.set_xlim(0, 25)
    ax.set_ylim(0, 800)
    ax.hist(tweet_len, color=color)
    ax.set_title(f'{label.title()} tweet_text')

fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(12,8))
generate_review_hist_plot(data,'Positive emotion','green',ax[0,0])
generate_review_hist_plot(data,'Negative emotion','red',ax[0,1])
generate_review_hist_plot(data,'No emotion toward brand or product','blue
generate_review_hist_plot(data,"I can't tell",'purple',ax[1,1])

fig.suptitle("Words Distribution in Tweet Text", fontsize=18)

plt.show()
```



Interpretation

Majority of the tweets in all the target emotions classes were 10-15 words long.

Re-tokenizing

In [41]:
```
# re-tokenizing the data
data['tweet_text'] = data['tweet_text'].apply(tokenize_text)
data.head()
```

Out[41]:

	tweet_text	recipient	emotion	brand
0	[user, 3g, iphone, 3, hrs, tweeting, riseausti...	iPhone	Negative emotion	Apple
1	[user, know, user, awesome, ipadiphone, app, l...	iPad or iPhone App	Positive emotion	Apple
2	[user, wait, ipad, 2, also, sale, sxsw]	iPad	Positive emotion	Apple
3	[user, hope, years, festival, crashy, years, i...	iPad or iPhone App	Negative emotion	Apple
4	[user, great, stuff, fri, sxsw, marissa, mayer...	Google	Positive emotion	Google

In [42]:
```
data.tweet_text[0]
```

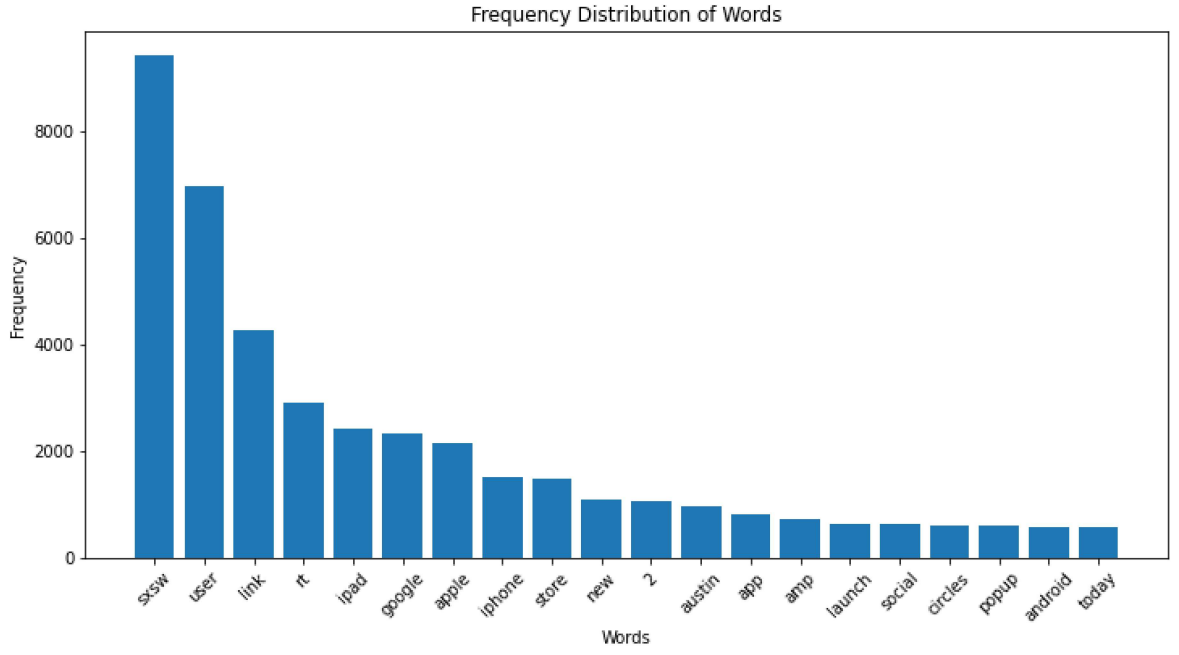
Out[42]:
```
['user',
 '3g',
 'iphone',
 '3',
 'hrs',
 'tweeting',
 'riseaustin',
 'dead',
 'need',
 'upgrade',
 'plugin',
 'stations',
 'sxsw']
```

Justification for Retokenizing

We retokinized since after removing the stopwords the tweets were no longer split as strings.

Frequency Distribution after Removing Stopwords

In [43]:
```
tweet_texts = data.tweet_text
# Flatten the list of tokens into a single list
all_tokens = [tweet for sublist in tweet_texts for tweet in sublist]

# Create the frequency distribution
freq_dist = FreqDist(all_tokens)

# Get the most common words
most_common = freq_dist.most_common(20)  # Example: Top 20 most common wo

# Extract the words and frequencies
words, frequencies = zip(*most_common)

# Create a bar plot
plt.figure(figsize=(12, 6))
plt.bar(words, frequencies)
plt.xlabel('Words')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of Words')
plt.xticks(rotation=45)
plt.show()
```



Observation

Just by looking at this, we can see there is a high occurence of

- Venue **(south by south west)** specific words such as `sxsw`
- Twitter specific words(**re-tweet** such as `rt`
- Brand or product specific words such as `iphone`

These words probably occur across our various sentiments and may not necessarily add any

Lemmatization

In [44]:
```
# Lemmatization - example cats - cat
# Create Lemmatizer
word_lem = WordNetLemmatizer()

# function
def lem_words(tweet):
    return [word_lem.lemmatize(word) for word in tweet]
```

In [45]:
```
data['tweet_text'] = data['tweet_text'].apply(lem_words)
data['tweet_text'].head()
```

Out[45]:
```
0    [user, 3g, iphone, 3, hr, tweeting, riseaustin...
1    [user, know, user, awesome, ipadiphone, app, l...
2              [user, wait, ipad, 2, also, sale, sxsw]
3    [user, hope, year, festival, crashy, year, iph...
4    [user, great, stuff, fri, sxsw, marissa, mayer...
Name: tweet_text, dtype: object
```

In [46]:
```
data.head()
```

Out[46]:

	tweet_text	recipient	emotion	brand
0	[user, 3g, iphone, 3, hr, tweeting, riseaustin...	iPhone	Negative emotion	Apple
1	[user, know, user, awesome, ipadiphone, app, l...	iPad or iPhone App	Positive emotion	Apple
2	[user, wait, ipad, 2, also, sale, sxsw]	iPad	Positive emotion	Apple
3	[user, hope, year, festival, crashy, year, iph...	iPad or iPhone App	Negative emotion	Apple
4	[user, great, stuff, fri, sxsw, marissa, mayer...	Google	Positive emotion	Google

Dealing with Emojis

We create a function that replaces emojis in tweet text with their corresponding meanings.

In [47]:
```
# Defining dictionary containing all emojis with their meanings.
emojis = {':)': 'smile', ':-)': 'smile', ';d': 'wink', ':-E': 'vampire',
          ':-(': 'sad', ':-<': 'sad', ':P': 'raspberry', ':O': 'surprised
          ':-@': 'shocked', ':@': 'shocked',':-$': 'confused', ':\\': 'an
          ':#': 'mute', ':X': 'mute', ':^)': 'smile', ':-&': 'confused',
          '@@': 'eyeroll', ':-!': 'confused', ':-D': 'smile', ':-0': 'yel
          '<(-_-)>': 'robot', 'd[-_-]b': 'dj', ":'-)": 'sadsmile', ';)':
          ';-)': 'wink', 'O:-)': 'angel','O*-)': 'angel','(:-D': 'gossip'
```

In [48]:
```
#function to remove emojis and sequence of letters like heeeey)
def process(tweets):

    processed_tweet = []


    # Defining regex patterns.

    sequencePattern   = r"(.)\1\1+" #matches three or more consecutive oc
    seqReplacePattern = r"\1\1"

    for tweet in tweets:

        # Replace all emojis.
        for emoji in emojis.keys():
            tweet = tweet.replace(emoji, "EMOJI" + emojis[emoji])
            # Replace 3 or more consecutive letters by 2 letter.
            tweet = re.sub(sequencePattern, seqReplacePattern, tweet)

        processed_tweet.append(tweet)

    return processed_tweet
```

In [49]:
```
data['tweet_text'] = data['tweet_text'].apply(process)
data.head()
```

Out[49]:

	tweet_text	recipient	emotion	brand
0	[user, 3g, iphone, 3, hr, tweeting, riseaustin...	iPhone	Negative emotion	Apple
1	[user, know, user, awesome, ipadiphone, app, l...	iPad or iPhone App	Positive emotion	Apple
2	[user, wait, ipad, 2, also, sale, sxsw]	iPad	Positive emotion	Apple
3	[user, hope, year, festival, crashy, year, iph...	iPad or iPhone App	Negative emotion	Apple
4	[user, great, stuff, fri, sxsw, marissa, mayer...	Google	Positive emotion	Google

In [50]:
```
data.tweet_text[0]
```

Out[50]:
```
['user',
 '3g',
 'iphone',
 '3',
 'hr',
 'tweeting',
 'riseaustin',
 'dead',
 'need',
 'upgrade',
 'plugin',
 'station',
 'sxsw']
```

vectorization

CountVectorization

Techniques is used for converting text documents into numerical representations

In [51]:
```
# create instance 1 vectorizer
bow = CountVectorizer()
#create lemmatized_review column
data['lemmatized_review'] = data['tweet_text'].str.join(" ")
x = bow.fit_transform(data['lemmatized_review'])
```

In [52]:
```
#checking feature names
feature_name = bow.get_feature_names_out()
feature_name
```

Out[52]:
```
array(['02', '03', '0310', ..., 'ûómy', 'ûóthe', 'ûóuser'], dtype=object)
```

Feature engineering

Label Encoding the Target

Here we label encode the target feature to transform the values to numerical values.

In [53]:
```
#instantiate labelencoder
le = LabelEncoder() # initializing the lib
#fit transform
data['emotion_code'] = le.fit_transform(data.emotion)
le.classes_ # viewing the classes
```

Out[53]:
```
array(["I can't tell", 'Negative emotion',
       'No emotion toward brand or product', 'Positive emotion'],
      dtype=object)
```

Previewing to ensure encoding is present

In [54]:
```
# preview
data.head()
```

Out[54]:

	tweet_text	recipient	emotion	brand	lemmatized_review	emotion_code
0	[user, 3g, iphone, 3, hr, tweeting, riseaustin...	iPhone	Negative emotion	Apple	user 3g iphone 3 hr tweeting riseaustin dead n...	1
1	[user, know, user, awesome, ipadiphone, app, l...	iPad or iPhone App	Positive emotion	Apple	user know user awesome ipadiphone app likely a...	3
2	[user, wait, ipad, 2, also, sale, sxsw]	iPad	Positive emotion	Apple	user wait ipad 2 also sale sxsw	3
3	[user, hope, year, festival, crashy, year, iph...	iPad or iPhone App	Negative emotion	Apple	user hope year festival crashy year iphone app...	1
4	[user, great, stuff, fri, sxsw, marissa, mayer...	Google	Positive emotion	Google	user great stuff fri sxsw marissa mayer google...	3

In [55]:
```
# displaying the encoding scheme
data[['emotion', 'emotion_code']].head(10)
```

Out[55]:

	emotion	emotion_code
0	Negative emotion	1
1	Positive emotion	3
2	Positive emotion	3
3	Negative emotion	1
4	Positive emotion	3
5	No emotion toward brand or product	2
7	Positive emotion	3
8	Positive emotion	3
9	Positive emotion	3
10	Positive emotion	3

6. Modeling

- The problem at hand is a classification problem.
- We will explore 2 models: a binary logistic regression model, a multi-class XGBoost model and MultinomialNB.
- Model accuracy will be the metric for evaluation.
- Justification: Accuracy to get a verdict if a tweet is positive or negative.

- Accuracy of 70% will be the threshold to deem the model as successful.

6.1 Binary classification

- In this section we create a base model to identify if a tweet is 'Positive' or 'Negative'.
- LogisticRegression will be used for the classification.
- The normal preprocessing of vectorization and train test split will be implemented.

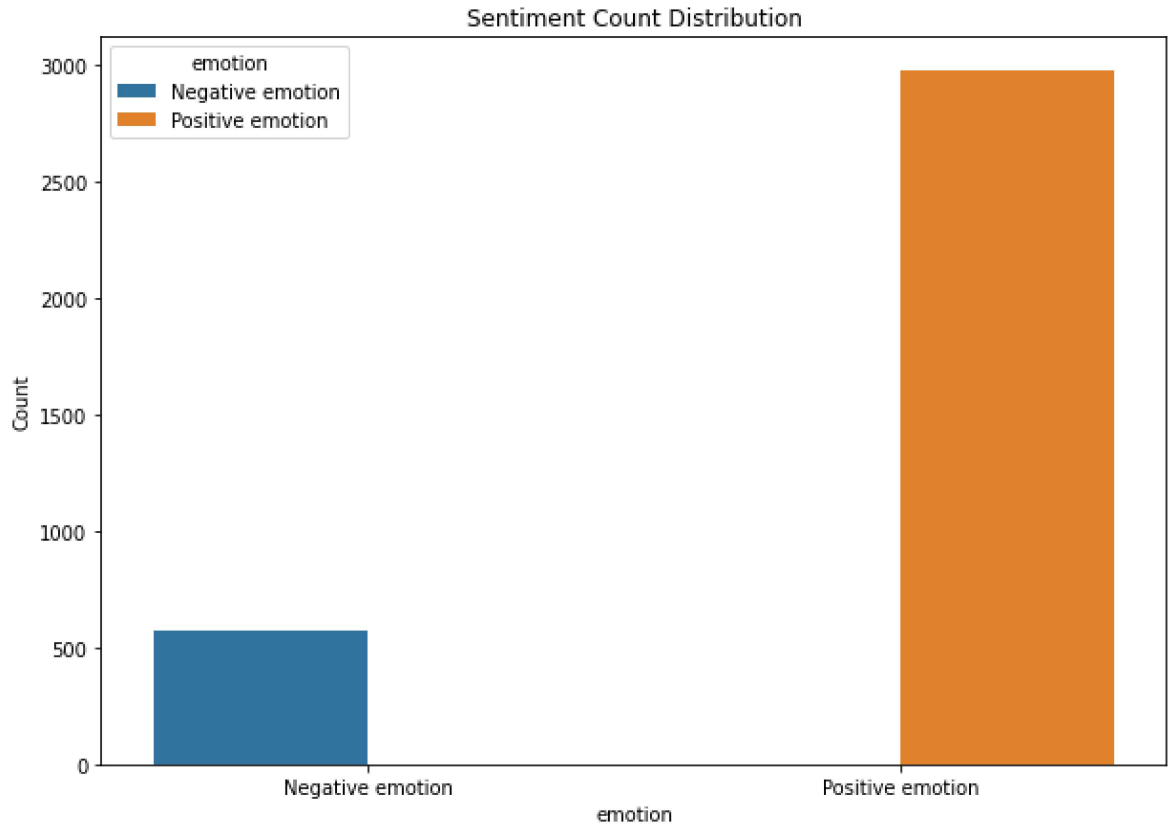
In [56]:
```
# creating a copy of the original data so as to remove unwanted rows i.e.
data_copy = data.copy()
```

In [57]:
```
# Define the values to drop
values_to_drop = [0, 2]

# Drop rows that have the values in column 'B'
data_copy = data_copy[~data_copy['emotion_code'].isin(values_to_drop)]
```

In [58]:
```
# value counts
data_copy['emotion_code'].value_counts()
```

Out[58]:
```
3    2970
1     569
Name: emotion_code, dtype: int64
```

In [59]:
```
# lets see see binary sentiment distribution emotion-wise
emotion_group = data_copy.groupby('emotion')['emotion_code'].value_counts
# Calculate the ascending order of states based on churn count
ascending_order = emotion_group.groupby('emotion')['count'].sum().sort_va

plt.figure(figsize=(10,7))
sns.barplot(x='emotion', y='count', hue='emotion', data=emotion_group, or
plt.xlabel('emotion')
plt.ylabel('Count')
plt.title('Sentiment Count Distribution')
plt.show()
```



Intepretation

- From the graph above it is clear that most of the sentiments were positive in nature

Vectorizing Data Copy

CountVectorizer

In [60]:
```
# create instance of vectorizer
bow2 = CountVectorizer()
data_copy['lemmatized_review'] = data_copy['tweet_text'].str.join(" ")
# our X
x2 = bow2.fit_transform(data_copy['lemmatized_review'])
```

In [61]:
```
# preview
data_copy.head()
```

Out[61]:

	tweet_text	recipient	emotion	brand	lemmatized_review	emotion_code
0	[user, 3g, iphone, 3, hr, tweeting, riseaustin...	iPhone	Negative emotion	Apple	user 3g iphone 3 hr tweeting riseaustin dead n...	1
1	[user, know, user, awesome, ipadiphone, app, l...	iPad or iPhone App	Positive emotion	Apple	user know user awesome ipadiphone app likely a...	3
2	[user, wait, ipad, 2, also, sale, sxsw]	iPad	Positive emotion	Apple	user wait ipad 2 also sale sxsw	3
3	[user, hope, year, festival, crashy, year, iph...	iPad or iPhone App	Negative emotion	Apple	user hope year festival crashy year iphone app...	1
4	[user, great, stuff, fri, sxsw, marissa, mayer...	Google	Positive emotion	Google	user great stuff fri sxsw marissa mayer google...	3

Train-test split

In [62]:
```
#x = bow.fit_transform(data['lemmatized_review'])
# our y
y = data_copy['emotion_code']
```

In [63]:
```
# train-test split
X_train, X_test, y_train, y_test = train_test_split(x2, y, test_size=0.2,
```

In [64]:
```
# confirming shape

X_train.shape, y_train.shape
```

Out[64]: ((2831, 5871), (2831,))

Fitting logistic regression model on split data

In [65]:
```
# instantiating model 1
log_reg = LogisticRegression(random_state=42)
```

In [66]:
```
# fitting
log_reg.fit(X_train, y_train)
```

Out[66]:
```
LogisticRegression
LogisticRegression(random_state=42)
```

In [67]:
```
# predicting
y_pred = log_reg.predict(X_test)
y_train_pred = log_reg.predict(X_train)
```

Evaluation

In [68]:
```
#Accuracy score
print('train',accuracy_score(y_train, y_train_pred))
print('test',accuracy_score(y_test, y_pred))
```

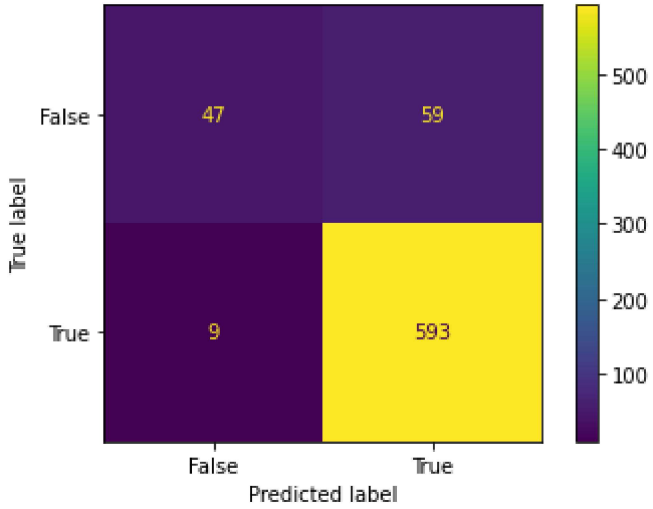
```
train 0.9699752737548569
test 0.903954802259887
```

In [69]:
```
# Classification metric Report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           1       0.84      0.44      0.58       106
           3       0.91      0.99      0.95       602

    accuracy                           0.90       708
   macro avg       0.87      0.71      0.76       708
weighted avg       0.90      0.90      0.89       708
```

In [70]:
```
#Confusion Matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
```

In [71]:
```
#Display Confusion Matrix
cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_
cm_display.plot()
plt.show()
```



Machine Learning Communication

Rationale why modeling was implemented.

- While simpler forms of data analysis, such as descriptive statistics or basic data visualization, can provide initial insights, they may not be sufficient for complex problems or large datasets. Machine learning leverages advanced algorithms to uncover hidden patterns.

Results.

> - Training accuracy: 96%
> - Testing accuracy: 90%

- The accuracy means that the model can predict with an accuracy of 90% whether a tweet is positive or negative.
- The current model is fit for prediction since it is generalizing well to new data even with high accuracy.

Limitations of binary model.

- Not fit for multiclass datasets.

Testing Logistic Regression Model here

- We take a sample sentence, preprocess it then pass it to the model to see the output.

In [72]:
```
# tweet
review = 'Awwww!! the google was sooo sweeet!! I loved it'
#review = 'i hate apple'
```

In [73]:
```
# lowercase
review = review.lower()
review
```

Out[73]: 'awwww!! the google was sooo sweeet!! i loved it'

In [74]:
```
# removing punctuations
review = remove_punctuations(review)
review
```

Out[74]: 'awwww the google was sooo sweeet i loved it'

In [75]:
```
# tokenizing the data
review = tokenize_text(review)
review
```

Out[75]: ['awwww', 'the', 'google', 'was', 'sooo', 'sweeet', 'i', 'loved', 'it']

In [76]:
```
# removing stopwords
review = remove_stopwords(review)
review
```

Out[76]: 'awwww google sooo sweeet loved'

In [77]:
```
review = tokenize_text(review)
review
```

Out[77]: ['awwww', 'google', 'sooo', 'sweeet', 'loved']

In [78]:
```
# lemmatization
review = lem_words(review)
review
```

Out[78]: ['awwww', 'google', 'sooo', 'sweeet', 'loved']

In [79]:
```
# joining to one sentence
review = ' '.join(review)

review
```

Out[79]:
```
'awwww google sooo sweeet loved'
```

In [80]:
```
# putting to list
review = [review]
review
```

Out[80]:
```
['awwww google sooo sweeet loved']
```

In [81]:
```
# vectorizing
review_x = bow2.transform(review)
review_x
```

Out[81]:
```
<1x5871 sparse matrix of type '<class 'numpy.int64'>'
        with 2 stored elements in Compressed Sparse Row format>
```

In [82]:
```
test_predict = log_reg.predict(review_x)
test_predict
```

Out[82]:
```
array([3])
```

6.2 Multiclass Classifier

- Here we work with the original dataset.
- We build a multi-class classifier.
- MultinomialNB and XGBoost model will be tested.

MultinomialNB model

In [83]:
```
# target variable
y = data['emotion_code']
```

In [84]:
```
# splitting the data
X_train_multi, X_test_multi, y_train_multi, y_test_multi = train_test_spl
```

In [85]:
```
X_train_multi.shape, y_train_multi.shape
```

Out[85]:
```
((7256, 9867), (7256,))
```

In [86]:
```
# instatiating the multi class model
multi_class = MultinomialNB()
# fitting the model
multi_class.fit(X_train_multi, y_train_multi)
```

Out[86]:
```
MultinomialNB()
```

In [87]:
```
# predicting multi-class
y_pred_multi = multi_class.predict(X_test_multi)
y_train_pred_multi = multi_class.predict(X_train_multi)
```

In [88]:
```
# checking the accuracy
print('train',accuracy_score(y_train_multi, y_train_pred_multi))
print('test',accuracy_score(y_test_multi, y_pred_multi))
```

```
train 0.8036108048511577
test 0.6543550165380375
```

Machine Learning Communication - MultiNomialmodel.

Rationale why MultiNomial modeling was implemented.

- While simpler forms of modeling can do , they may not be able to work with multi-class datasets.

Results.

- Accuracy on the training set: 80%
- Accuracy on the testing set: 66%
- The model is overfitting

Limitations of multinomial model.

- The current model is not fit for prediction since it is not generalizing well to new data even with high accuracy. The model is overfitting.

XGBoost model

In [89]:
```
# Create an XGBoost classifier instance
xgb_classifier = xgb.XGBClassifier()

# Fit the model on the training data
xgb_classifier.fit(X_train_multi, y_train_multi)

```

Out[89]:
```
                                XGBClassifier

XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=
1,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwi
se',
              importance_type=None, interaction_constraints='',
              learning_rate=0.30000001, max_bin=256, max_cat_to_onehot=
4,
              max_delta_step=0, max_depth=6, max_leaves=0, min_child_wei
```

In [90]:
```
# Make predictions on the testing data
#predictions = xgb_classifier.predict(test_x)
y_pred_xgb =  xgb_classifier.predict(X_test_multi)
y_train_pred_xgb =  xgb_classifier.predict(X_train_multi)
```

In [91]:
```
# checking the accuracy
print('train',accuracy_score(y_train_multi, y_train_pred_xgb))
print('test',accuracy_score(y_test_multi, y_pred_xgb))
```

```
train 0.7848676957001103
test 0.685226019845645
```

Machine Learning Communication XGBoost.

Rationale why ensemble modeling was implemented.

- While simpler forms of data analysis, such as descriptive statistics or basic data visualization, can provide initial insights, they are not sufficient for complex problems or large datasets such as this one. Ensemble models leverages advanced algorithms to uncover hidden patterns, make accurate predictions.

Results.

- Accuracy on the training set: 78%
- Accuracy on the testing set: 68%
- The accuracy means that the model can predict with an accuracy of 78% whether a tweet falls within the specified labels.

Limitations.

- The current model is not fit for prediction since it is overfitting.
- This we see from the difference between train and test accuracy.

Resampling to handle the class imbalance issue.

We try to battle overfitting by handling imbalance issues in the classes, then modelling again.

Trying Resampling with xgboost

In [92]:
```
# Creating an instance of the resampler
resampler = RandomOverSampler()  # or RandomUnderSampler()

# Perform resampling on the training data
resampled_features, resampled_labels = resampler.fit_resample(X_train_mul

# Create an XGBoost classifier instance
xgb_classifier = xgb.XGBClassifier()

# Fit the model on the resampled training data
xgb_classifier.fit(resampled_features, resampled_labels)

# Make predictions on the testing data
predictions = xgb_classifier.predict(X_test_multi)

# Evaluate the model
accuracy = accuracy_score(y_test_multi, predictions)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6278941565600882
```

ML Communication

Result

The resampled model has an accuracy of 61% which is not an improvement over prior models.

Limitations

The resampled model is not fit for modelling since its modelling accuracy is low.

Final note on modeling

- Despite implementing resampling techniques to address class imbalance, the model's accuracy has not improved significantly. This suggests that other factors within the dataset may be limiting predictive performance.

- The binary logistic regression model performs best.
- For Multiclass XGBoost was better

Limitation and Challenges

- Class Imbalance Issue: The dataset suffers from class imbalance, where one sentiment class is dominant while others are underrepresented. This can result in biased models that are more accurate for the majority class but perform poorly on the minority classes. Addressing this issue is important to ensure fair and balanced sentiment analysis.
- Limited Dataset Size: The dataset used for sentiment analysis is relatively small, which can limit the model's ability to capture the full complexity of sentiments expressed in text. A larger and more diverse dataset would provide a broader representation of sentiments and improve the model's performance and generalization.
- Language Ambiguity and Sarcasm Detection: Language can be inherently ambiguous, and detecting sarcasm in text adds an extra layer of complexity. Sarcasm detection is challenging due to the subtleties and nuances involved. Developing robust strategies to handle language ambiguity and detect sarcasm is crucial for accurate sentiment analysis

7.Findings

1. Most tweets were directed to None brand category. This may indicate that customers were not engaging with the brand.
2. Positive sentiments had the highest count compared to Negative sentiments, indicating that most people in general liked respective brands(Google and Apple)
3. Most of the positive tweets were directed to Apple brands
4. In the field of sentiment analysis, one of the significant challenges is dealing with language ambiguity and sarcasm detection. Natural language is complex and often subjective, making it difficult to accurately interpret sentiments from text.
5. On average most of the tweets were 10-15 words long - more words increase ambiguity.
6. NLP is a difficult task to model accurately.

8.Recommendations

We recommend that there be more customer engagement.

Probably check on this areas;

- Churn ratio - rate at which customers discontinue their relationship with a product company within a given time period
- Social media influencers through brand or product endorsement
- Customer feedback - The brands can introduce a rating system to accurately capture the sentiments of their customers

9. Next steps

- In our future work, we plan to explore advanced techniques such as incorporating attention mechanisms, using ensemble methods to further enhance the model's performance by incorporating domain-specific and fine-tuning the model on industry-specific datasets could improve its accuracy and adaptability.
- By considering these evaluation metrics, addressing limitations, and planning for future improvements, we aim to develop a robust NLP sentiment analysis solution that effectively captures sentiment

In [93]:
```
import joblib

# Save the trained model as a pickle file
joblib.dump(log_reg, 'logistic_regression_model.pkl')
joblib.dump(xgb_classifier, 'xgb.pkl')
```

Out[93]:
```
['xgb.pkl']
```