SENTIMENTAL ANALYSIS FOR MARKETING USING AI

41012110426:MEENAKSHI.K

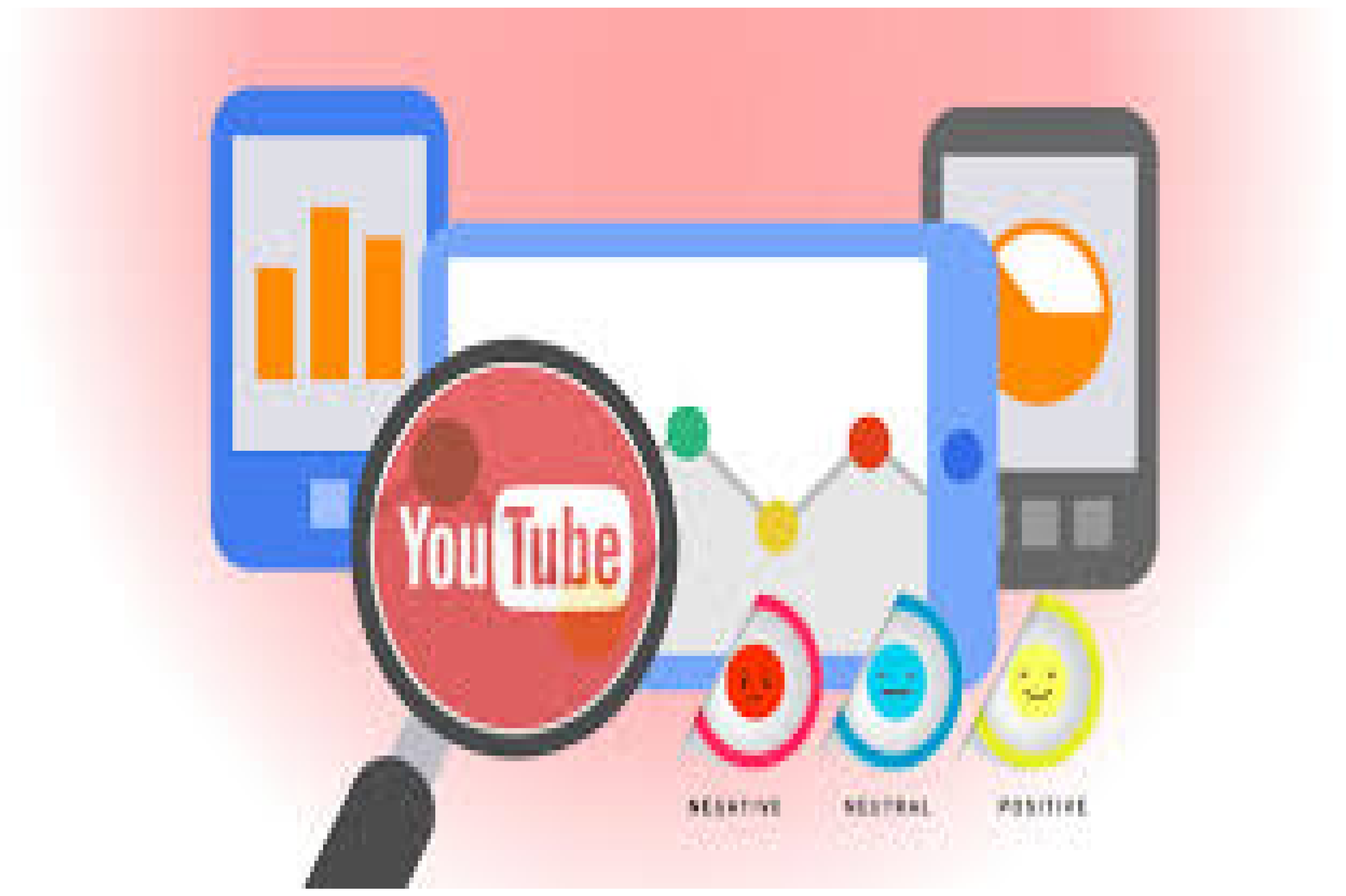
Phase 5 submission document

Project Title: Sentimental analysis for marketing

Topic:

In this part we will document the complete project and prepare it for submission.

Sentimental analysis for marketing



Introduction:

Sentiment analysis, also known as opinion mining, is a process used to determine and understand the sentiment expressed in a piece of text. In the context of marketing, sentiment analysis involves analyzing consumer opinions, reviews, and social media interactions to gain insights into customer attitudes, preferences, and emotions related to a product, service, brand, or marketing campaign.

Marketers use sentiment analysis to:

1. Customer Feedback Analysis: Analyze customer reviews and feedback to understand their satisfaction levels and identify areas for improvement.

2. Brand Reputation Management: Monitor social media mentions and online conversations to gauge public perception of the brand and take necessary actions to maintain a positive image.

3. Product Development: Utilize sentiment analysis to identify features that customers like or dislike about existing products, aiding in the development of new products aligned with customer preferences.

4. Competitor Analysis: Compare sentiments around your brand with those of competitors to identify competitive advantages or areas where your brand can excel.

5. Campaign Evaluation: Evaluate the effectiveness of marketing campaigns by analyzing the sentiment of social media posts and customer responses, helping in refining future marketing strategies.

6. Customer Service Improvement: Analyze customer support interactions to assess customer satisfaction levels

and identify areas where customer service can be enhanced.

Sentiment analysis in marketing often employs natural language processing (NLP) techniques and machine learning algorithms to classify text as positive, negative, or neutral. By leveraging sentiment analysis, marketers can make data-driven decisions, improve customer experiences, and enhance overall brand perception in the market.

Dataset Link: https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment

Here’s a list of tools and software commonly used in the process:

Sentiment analysis in marketing relies on various tools and techniques to analyze and interpret customer sentiments. Some common tools and methods include:

1. Natural Language Processing (NLP) Libraries: Tools like NLTK, spaCy, and TextBlob are used to preprocess and analyze text data, including tokenization, lemmatization, and part-of-speech tagging.

2. Machine Learning Algorithms: Supervised machine learning algorithms, such as Support Vector Machines (SVM), Naive Bayes, and deep learning models (e.g., LSTM and BERT), are employed to classify text as positive, negative, or neutral.

3. Social Media Monitoring Tools: Platforms like Brandwatch, Hootsuite, and Sprout Social help marketers monitor and analyze social media mentions and comments to gauge sentiment.

4. Sentiment Analysis APIs: Services like the Google Cloud Natural Language API, IBM Watson Natural Language Understanding, and Microsoft Azure Text Analytics provide pre-built sentiment analysis capabilities.

5. Custom Datasets: Building a custom dataset of labeled text data (positive, negative, neutral) specific to your brand or industry can enhance the accuracy of sentiment analysis models.

6. Data Visualization Tools: Tools like Tableau and Power BI help in creating visual representations of sentiment analysis results for easier interpretation.

7. Social Listening Tools: Tools like Talkwalker and Mention enable tracking of brand mentions and sentiment on various online platforms.

8. Word Cloud Generators: Word clouds can visually represent the most common words in positive and negative sentiment to identify key themes and trends.

9. Sentiment Analysis Dashboards: Creating custom dashboards in tools like Google Data Studio or Tableau to track sentiment trends over time.

10. Survey and Feedback Analysis Software: Tools like SurveyMonkey and Qualtrics can be used to collect and analyze structured feedback and survey responses.

11. Competitor Analysis Tools: Tools like SEMrush and Ahrefs can be used to analyze the sentiment surrounding competitors in your industry.

12.Social Media Listening Tools: Tools like Awario and Brand24 help monitor social media conversations and customer opinions.

13. Customer Relationship Management (CRM) Software: CRM systems like Salesforce or HubSpot often include sentiment analysis features to track customer interactions and sentiment.

14. Web Scraping Tools: Tools like BeautifulSoup and Scrapy can be used to gather unstructured data from websites, forums, and review platforms for sentiment analysis.

15. Language Translation Services: When dealing with multilingual data, translation services like Google Translate can be used to convert text to a common language before sentiment analysis.

DESGIN THINKING AND PRESENT IN FORM OF DOCUMENT:

1. Empathize:

- Understand the target audience by collecting data from social media, surveys, and customer feedback.

- Use sentiment analysis tools to identify emotional responses and opinions related to the product or service.

2. Define:

- Define the marketing problem or opportunity based on the sentiment analysis data.

- Identify the specific emotional triggers that influence consumer behavior, such as positive reviews or negative comments.

3. Ideate:

- Brainstorm creative marketing strategies to address the defined problem or capitalize on the opportunity.

- Encourage a diverse group of team members to generate innovative ideas that resonate with the identified sentiments.

4. Prototype:

- Develop prototypes of marketing materials, campaigns, or advertisements that align with the chosen strategies.

   - Ensure the prototypes convey the desired emotional tone and message based on the sentiment analysis.

5. Test:

   - Launch small-scale marketing campaigns or A/B tests to gauge the emotional impact on the target audience.

   - Continuously monitor sentiment through sentiment analysis tools to measure the effectiveness of the strategies.

6. Implement:

   - Roll out the most effective marketing strategies based on the test results.

   - Ensure that the emotional content and messaging align with the sentiments of the audience.

7. Evaluate:

   - Continuously assess the impact of marketing efforts by tracking sentiment and key performance indicators (KPIs).

   - Make adjustments as needed based on evolving sentiment and consumer feedback.

8. Iterate:

- Use the feedback and insights gathered to refine and improve marketing strategies.
- Apply design thinking principles to adapt to changing emotional trends and consumer preferences.

Phase 1-Design and Thinking:

1. Text Preprocessing:

Tokenization: Break down text into words or phrases for analysis.

Lowercasing: Convert all text to lowercase to ensure consistency.

Removing Stopwords: Eliminate common words (e.g., "and," "the") that don't carry significant meaning.

Stemming and Lemmalization: Reduce words to their base or root form to capture core meaning.

2. Feature Extraction:

Bag of Words (BoW):Represent text as a collection of words, ignoring grammar and word order.

Term Frequency-Inverse Document Frequency (TF-IDF):Weigh words based on their importance in a document relative to a collection of documents.

Word Embeddings (Word2Vec, GloVe):Represent words as vectors in multi-dimensional space, capturing semantic relationships.

3. Sentiment Analysis Algorithms:

Rule-Based Approaches: Use predefined rules and dictionaries to determine sentiment polarity.Machine Learning Models (Naive Bayes, Support Vector Machines, Neural Networks): Train models on labeled data to predict sentiment.

4. Aspect-Based Sentiment Analysis:

Dependency Parsing: Identify relationships between words to extract specific aspects mentioned in the text.

Target-Dependent Sentiment Classification: Classify sentiment concerning specific aspects or entities mentioned in the text.

5. Emotion Recognition:

Emotion Lexicons: Utilize dictionaries mapping words to emotions to identify emotional tones in text.

Emotion Classifiers: Train models to recognize primary emotions (e.g., happiness, anger, sadness) in customer expressions.

6. Social Media-Specific Techniques:

Hashtag and Mention Analysis: Extract sentiments associated with specific hashtags or mentions.

Emoji Analysis: Interpret emojis to understand emotional context in social media posts.

7. Domain Adaptation and Customization:

Custom Dictionaries: Create domain-specific dictionaries and vocabularies tailored to the marketing context.

Fine-Tuning Pre-trained Models: Adapt existing models on marketing-specific data to enhance accuracy.

8. Visualization and Interpretation:

Word Clouds: Visualize frequently occurring words to identify common sentiments and topics.

Sentiment Heatmaps: Present sentiment scores visually, providing an overview of positive and negative sentiments over different aspects.

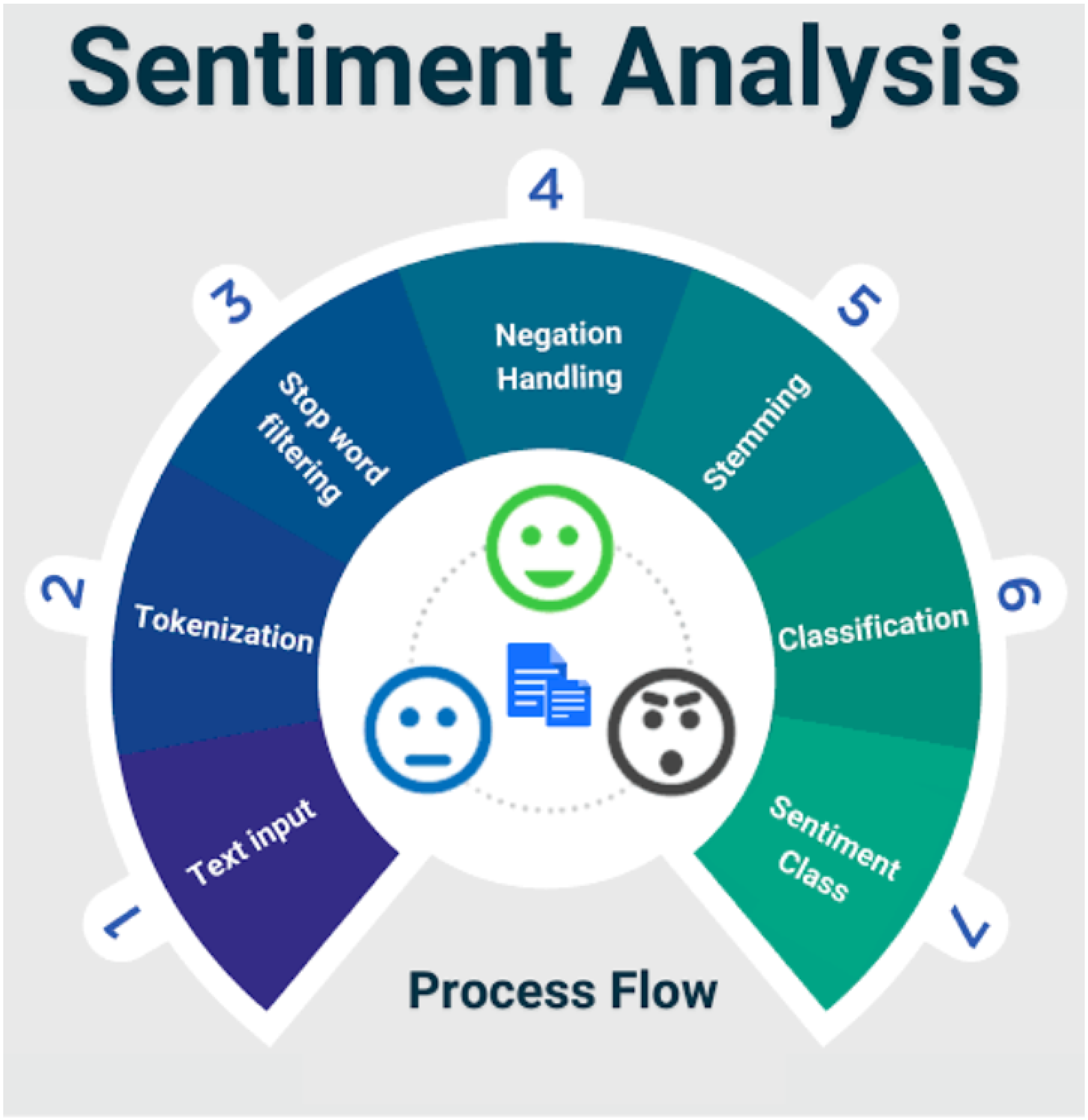
Interactive Dashboards: Build interactive dashboards to explore sentiment trends and customer feedback visually.

9. Continuous Monitoring and Feedback:

Feedback Loop: Establish mechanisms for users to provide feedback on the accuracy of sentiment predictions, allowing for continuous improvement.

Regular Model Updating: Periodically update models to adapt to changing language trends and customer expressions.

Project model:



Phase 2-Innovation:

BERT:

BERT, or Bidirectional Encoder Representations from Transformers, is a natural language processing (NLP) model developed by Google. It's designed to understand the context of words in a sentence by considering the surrounding words on both sides. BERT has significantly improved the accuracy of tasks such as language understanding, sentiment analysis, and question answering in NLP applications.

RoBERTa:

RoBERTa, short for Robustly optimized BERT approach, is a natural language processing (NLP) model developed by Facebook AI. It is based on the BERT (Bidirectional Encoder Representations from Transformers) architecture but comes with improvements. RoBERTa removes the Next Sentence Prediction objective from BERT, uses dynamic masking during training, and trains on a larger dataset. These modifications make RoBERTa a powerful model for various NLP tasks, such as text classification, named entity recognition, and language understanding.

Program:

```
from mpl_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt # plotting

import numpy as np # linear algebra

import os # accessing directory structure

import pandas as pd # data processing, CSV file I/O (e.g.
pd.read_csv)

There is 1 csv file in the current version of the dataset:
```

```
print(os.listdir('../input'))
```

```
['database.sqlite', 'Tweets.csv']
```

The next hidden code cells define functions for plotting data. Click on the "Code" button in the published kernel to reveal the hidden code.

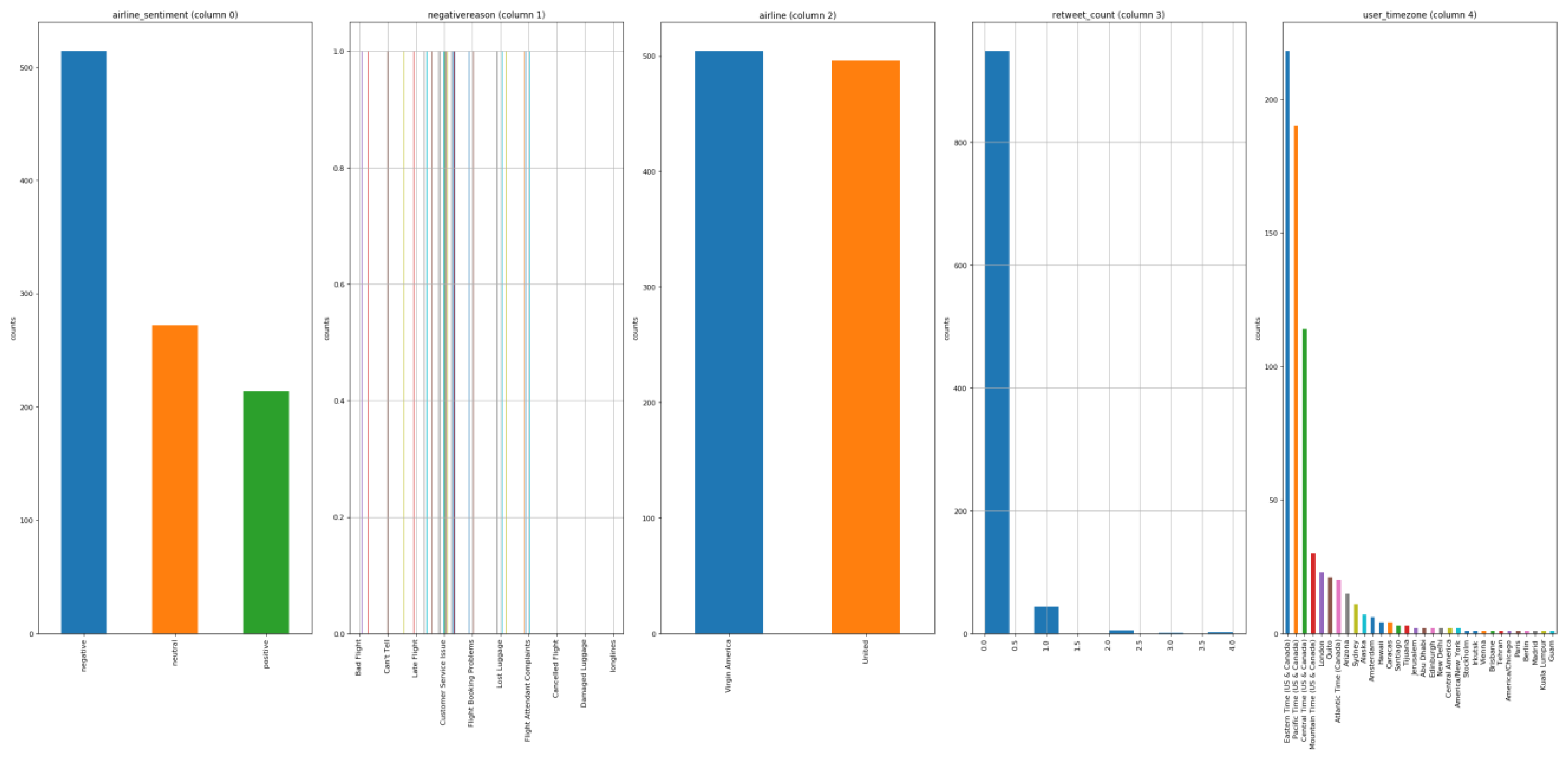
```
# Distribution graphs (histogram/bar graph) of column data
def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):
    nunique = df.nunique()
    df = df[[col for col in df if nunique[col] > 1 and nunique[col] < 50]] # For displaying purposes, pick columns that have between 1 and 50 unique values
    nRow, nCol = df.shape
    columnNames = list(df)
    nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow
    plt.figure(num = None, figsize = (6 * nGraphPerRow, 8 * nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')
    for i in range(min(nCol, nGraphShown)):
        plt.subplot(nGraphRow, nGraphPerRow, i + 1)
        columnDf = df.iloc[:, i]
        if (not np.issubdtype(type(columnDf.iloc[0]), np.number)):
            valueCounts = columnDf.value_counts()
            valueCounts.plot.bar()
        else:
            columnDf.hist()
        plt.ylabel('counts')
        plt.xticks(rotation = 90)
        plt.title(f'{columnNames[i]} (column {i})')
    plt.tight_layout(pad = 1.0, w_pad = 1.0, h_pad = 1.0)
    plt.show()

# Correlation matrix
def plotCorrelationMatrix(df, graphWidth):
    filename = df.dataframeName
    df = df.dropna('columns') # drop columns with NaN
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns
where there are more than 1 unique values
    if df.shape[1] < 2:
        print(f'No correlation plots shown: The number of non-NaN
or constant columns ({df.shape[1]}) is less than 2')
        return
    corr = df.corr()
    plt.figure(num=None, figsize=(graphWidth, graphWidth),
dpi=80, facecolor='w', edgecolor='k')
    corrMat = plt.matshow(corr, fignum = 1)
    plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)
    plt.yticks(range(len(corr.columns)), corr.columns)
    plt.gca().xaxis.tick_bottom()
    plt.colorbar(corrMat)
    plt.title(f'Correlation Matrix for {filename}', fontsize=15)
    plt.show()
# Scatter and density plots
def plotScatterMatrix(df, plotSize, textSize):
    df = df.select_dtypes(include =[np.number]) # keep only
numerical columns
    # Remove rows and columns that would lead to df being
singular
    df = df.dropna('columns')
    df = df[[col for col in df if df[col].nunique() > 1]] # keep columns
where there are more than 1 unique values
    columnNames = list(df)
    if len(columnNames) > 10: # reduce the number of columns
for matrix inversion of kernel density plots
        columnNames = columnNames[:10]
    df = df[columnNames]
    ax = pd.plotting.scatter_matrix(df, alpha=0.75, figsize=[plotSize,
 plotSize], diagonal='kde')
    corrs = df.corr().values
    for i, j in zip(*plt.np.triu_indices_from(ax, k = 1)):
        ax[i, j].annotate('Corr. coef = %.3f' % corrs[i, j], (0.8, 0.2),
xycoords='axes fraction', ha='center', va='center', size=textSize)
    plt.suptitle('Scatter and Density Plot')
    plt.show()nRowsRead = 1000 # specify 'None' if want to read
whole file
# Tweets.csv has 14640 rows in reality, but we are only
loading/previewing the first 1000 rows
df1 = pd.read_csv('../input/Tweets.csv', delimiter=',', nrows =
nRowsRead)
df1.dataframeName = 'Tweets.csv'
nRow, nCol = df1.shape
print(f'There are {nRow} rows and {nCol} columns')



df1.head(5)
```

Output:



Phase 3-loading and preprocessing dataset:

Necessary step to follow:

Import Necessary Libraries:

Start by importing the required libraries, such as Pandas, NumPy, and Natural Language Processing (NLP) libraries like NLTK or spaCy.

Load and Explore the Dataset:

Load your sentiment dataset (e.g., CSV file) using Pandas. You should have a column with text data and another with labels (positive/negative).

Text Preprocessing:

Preprocess the text data. This includes:

- Lowercasing the text.

- Removing punctuation and special characters.

- Tokenization (splitting text into words or tokens).

- Removing stop words (common words like "the," "and," "is" that do not contribute much to sentiment).

Text Vectorization:

You need to convert text data into numerical form. Common techniques include:

- Bag of Words (BoW): Count the frequency of each word in the text.

- TF-IDF (Term Frequency-Inverse Document Frequency): Weigh words based on their importance in the document and across the corpus.

- Word Embeddings (e.g., Word2Vec or GloVe): Represent words in a dense vector space.

Split the Dataset:

Divide your dataset into training and testing sets to evaluate the model's performance.

Select a Machine Learning Model:

Choose a machine learning algorithm for sentiment analysis, like Logistic Regression, Naive Bayes, or a neural network.

Train and Evaluate the Model:

Train the model using the training data and evaluate its performance

using the testing data. Common metrics include accuracy, precision, recall, and F1-score.

Make Predictions:

Once the model is trained, you can use it to make sentiment predictions for new text data.

Program:

```
    # import contractions library.
!pip install contractions missingno wordcloud
In [13]:
linkcode
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import warnings
warnings.filterwarnings(action='ignore')

# Import NLTK and download required resources
```

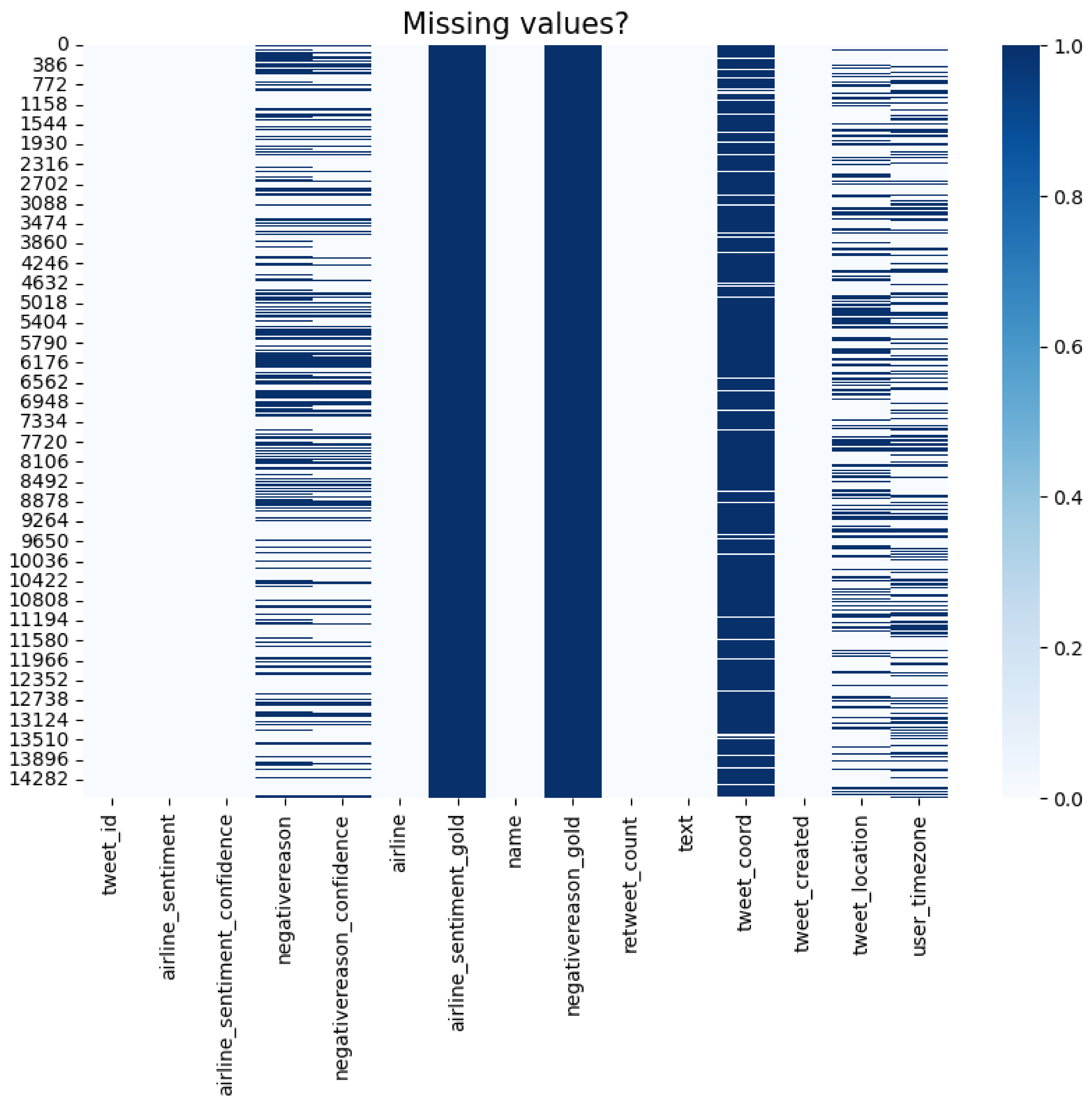
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import LancasterStemmer, WordNetLemmatizer

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')

# Import other libraries
import re
import string
import unicodedata
import contractions
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
import wordcloud
import train_test_split, StratifiedKFold
from sklearn.svm import LinearSVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import (
    recall_score,
    accuracy_score,
    confusion_matrix,
    classification_report,
    f1_score,
    precision_score,
    precision_recall_fscore_support
)

# Set options for displaying data
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", 200)

df = pd.read_csv('Tweets.csv')
df.head()
```



Phase 4-Building the project:

EMPLOYING NLP TECHNIQUES:

As the range of methods and purposes for sentiment analysis is so broad, for the purposes of this introduction we will be focusing on the most common form of

sentiment analysis system: assigning a positive, neutral, or negative sentiment to text data drawn from any source - this could be in the form of social media comments, emails, online conversations, or even the automatically generated transcription from phone conversations.

There are two main approaches to perform this kind of sentiment analysis: classical and deep learning. Both of these methods are available with python.

Using a dictionary of manually defined keywords :

It is based on the assumption that we know what words are typically associated with positive and negative emotions. For example, if we are going to classify movie reviews, we expect to find words such as “great”, “super”, and “love” in positive comments and words like “hate”, “bad”, and “awful” in negative comments. We can count the number of occurrences of every selected word to define feature vectors. Then, we can train a sentiment analysis classifier on each comment.

BAG OF WORDS:

Instead of calculating only words selected by domain experts, we can calculate the occurrences of every word that we have in our language (or every word that occurs at least once in all of our data). This will cause our vectors to be much longer, but we can be sure that we will not miss any word that is important for prediction of sentiment.

TF-IDF STRATERGY:

We can think about TF-IDF as a modified version of the bag of words.

Instead of treating every word equally, we normalize the number of occurrences of specific words by the number of its occurrences in our whole data set and the number of words in our document (comments, reviews, etc.). This means that our model will be less sensitive to occurrences of common words like “and”, “or”, “the”, “opinion” etc., and focus on the words that are valuable for analysis.

PROGRAM:

```
import nltk

from nltk.corpus import stopwords

from nltk.stem import WordNetLemmatizer



def expression_check(prediction_input):

    if prediction_input == 0:

        print("Input statement has Negative Sentiment.")

    elif prediction_input == 1:

        print("Input statement has Positive Sentiment.")

    else:

        print("Invalid Statement.")

# function to take the input statement and perform the same
transformations we did earlier

def sentiment_predictor(input):
    input = text_transformation(input)
    transformed_input = cv.transform(input)
    prediction = rfc.predict(transformed_input)
    expression_check(prediction)
input1 = ["Sometimes I just want to punch someone in the face."]
input2 = ["I bought a new phone and it's so good."]
sentiment_predictor(input1)
sentiment_predictor(input2)
```

Output:

GENERATING INSIGHTS :

· Topics: Sentiment analysis can identify the main themes that are coming up in your data – like product quality, shipping, customer service, pricing, and so on. You can get pretty granular by analyzing sentiment for each topic to prioritize where you should focus your efforts. Often times, a single customer comment will cover many topics at once.

· Intent: Going beyond simple polarity (positive or negative), sentiment analysis can uncover where customers are asking questions or even giving suggestions. Looking at intent can provide more qualitative context behind customer comments.

· Emotion: Of course, this is the bread and butter of sentiment analysis. What are your customers feeling in their interactions with your brand? Delight, frustration, excitement, anger? NLP does the heavy lifting here to identify and categorize customer sentiment.

· Root Cause: When customers are having issues, sentiment analysis can get to the root cause, a powerful outcome for any company. Details like at which stage of the customer journey a problem occurs enable you to take action to resolve it quickly and efficiently.

· Sentiment Score: This is a pretty common metric designed to benchmark the general sentiment for analyzed text data. Drawing correlations between trends in the sentiment score and business decisions, new product offerings, or other milestones can indicate how customers felt about a specific product, service, or experience.

PROGRAM:

```
    import pandas as pd
import matplotlib.pyplot as plt


# Sample sentiment data
data = [
  {"text": "I love this product!", "sentiment": "positive"},
  {"text": "It's an okay movie.", "sentiment": "neutral"},
  {"text": "This is awful.", "sentiment": "negative"},
  # Add more data as needed
]
```

```
# Create a DataFrame from the sample data
df = pd.DataFrame(data)

# Overall sentiment distribution
sentiment_counts = df["sentiment"].value_counts()

# Plot the sentiment distribution
sentiment_counts.plot(kind="bar", title="Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.show()

# Calculate and display the percentage of positive, negative, and
neutral sentiments
total_count = len(df)
positive_percentage = (sentiment_counts.get("positive", 0) /
total_count) * 100
negative_percentage = (sentiment_counts.get("negative", 0) /
total_count) * 100
neutral_percentage = (sentiment_counts.get("neutral", 0) /
total_count) * 100
```

```
print(f"Positive Sentiment: {positive_percentage:.2f}%")
print(f"Negative Sentiment: {negative_percentage:.2f}%")
print(f"Neutral Sentiment: {neutral_percentage:.2f}%")
```

OUTPUT:

```
Positive Sentiment: 33.33%
Negative Sentiment: 33.33%
Neutral Sentiment: 33.33%
```

Conclusion:

In conclusion, sentiment analysis is a valuable tool for marketing that allows businesses to gain insights into customer opinions and emotions. By analyzing sentiment in social media, reviews, and other online content, marketers can better understand their audience, identify trends, and make data-driven decisions to improve products, services, and campaigns. Leveraging sentiment analysis can lead to more effective marketing strategies, enhanced customer engagement, and ultimately, improved brand perception and profitability.