**SENTIMENT ANALYSIS FOR MARKETING**

**NAME : ARUNACHALAM A**

**NM ID : au513521104005**

**EMAIL ID : arunpraveen0807@gmail.com**

**COLLEGE NAME : Annai Mira College of Engineering and Technology**

**PHASE 5**

***What is sentiment analysis in marketing?***

Sentiment analysis is a marketing tool that helps you examine the way people interact with a brand online. This method is more comprehensive than traditional online marketing tracking, which measures the number of online interactions that customers have with a brand, like comments and shares. Using sentiment analysis, you can label individual interactions as positive, negative or neutral. Once you've figured out how to determine and track these labels, you can use this new data set for a variety of marketing purposes, including your online strategy.

Sentimental analysis is an extremely useful tool to have since higher numbers of interactions don't always equate to better results. For example, if you were to receive 10 replies on a social post and all of them were positive, your post likely had a more compelling effect on your audience than if you receive 100 replies with only 10 of them being positive. The primary purpose of sentiment analysis is to respond to commentary more constructively

***3 types of sentiment analysis***

To perform sentiment analysis, a marketing team might use a software platform that creates an algorithm to monitor customer engagement online. There are three fundamental ways to develop an algorithm for distinguishing social sentiment:

* **Manual analysis:** This type uses manually created rules based on neurolinguistic principles, such as stemming and tokenization. It takes a long time to set up, but it's easy to change and customize.
* **Automatic analysis:** This type uses machine learning techniques that use neural networks and statistical models to classify language. It can be challenging to change, but it's easy to set up and manage.
* **Hybrid analysis:** This type uses both rules-based and machine-learning analyses. It's a balanced approach that most social listening applications employ.

***What are the benefits of sentiment analysis?*** Most online platforms implement their own algorithms to display content, so sentimental analysis allows you to track relevant commentary in a way that's most useful for your purposes. Conversations about a brand y can begin and end quickly, so using a standardized method to track them can give you valuable insights into customer browsing habits. Here are some of the benefits of sentiment analysis:

**Understanding your audience and defining your niche**

Sentiment analysis allows you to look at your audience from a much more granular perspective, which can help you identify a market niche that fits the products and services your company sells. Understanding what your brand means to your current customers can help you increase your brand’s market share much more quickly. For example, the owners of a struggling ice cream parlor can benefit from learning which flavors people like or dislike. They might improve profits by marketing liked flavors more heavily, and discontinuing less popular flavors could help reduce their operating costs.

***Improving customer service support and managing PR issues***

Many businesses use social media channels for customer service support because it lets them resolve issues in a personalized yet immediate way. Responding to negative comments can help de-escalate situations before they grow into something less manageable. For example, if a customer were to tag your brand in a post in which they're upset about a defective product, you can respond publicly to apologize. You can then follow up by messaging them privately to reinforce your commitment to quality. Handling negative sentiments effectively and publicly can also show other customers that the company has excellent service policies.

CODING FOR TWITTER USAIRLINES USING SENTIMENTAL ANALYSIS FOR MARKETING

This code performs sentiment analysis on airline tweets using a logistic regression model and visualizes various aspects of the analysis. Below is a step-by-step summary of the code's strategy:

1. **Data Import and Preprocessing:**
   * The code begins by importing the necessary libraries, including pandas for data handling, matplotlib and seaborn for visualization, and scikit-learn for machine learning.
   * The airline tweet dataset is loaded from a CSV file.
2. **Data Selection:**
   * Only two columns, 'airline\_sentiment' (the sentiment label) and 'text' (the tweet content), are selected from the dataset.
3. **Sentiment Distribution Visualization:**
   * The code creates a histogram to visualize the distribution of airline sentiments.
   * It also creates a pie chart to visualize the sentiment distribution using percentages.
4. **Data Label Mapping:**
   * A mapping dictionary is defined to convert sentiment labels ('positive,''negative,''neutral') to numerical values (1, 0, 2).
   * Leading and trailing whitespaces in the 'airline\_sentiment' column are removed.
   * A new 'target' column is created to store the numerical sentiment values.
5. **Data Splitting:**
   * The dataset is split into training and testing sets using the train\_test\_split function.
6. **Text Vectorization (TF-IDF):**
   * Text data is vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. It generates TF-IDF features for both the training and testing sets.
7. **Model Training (Logistic Regression):**
   * A logistic regression model is trained using the training data.
8. **Model Evaluation:**
   * The code evaluates the model using the training and testing datasets:
     + It calculates and prints train and test accuracies.
     + It calculates ROC AUC scores for both the train and test sets.
     + It creates normalized confusion matrices and visualizes them using heatmaps for both the train and test sets.

**CODING FOR GIVEN DATASETLINK:**

In [2]:

*# Drop unnecessary columns*

df=df[['airline\_sentiment', 'text']]

*# Display the first 5 rows of the dataframe after dropping unnecessary columns*

df.head()

Out[2]:

|  |  |  |
| --- | --- | --- |
|  | airline\_sentiment | text |
| 0 | Neutral | @VirginAmerica What @hepburnsaid. |
| 1 | positive | @VirginAmerica plus you've added commercials t... |
| 2 | Neutral | @VirginAmerica I didn't today... Must mean I n... |
| 3 | negative | @VirginAmerica it's really aggressive to blast... |
| 4 | negative | @VirginAmerica and it's a really big bad thing... |

In [3]:

*# Function to preprocess the text*

defpreprocess\_text(text):

*# Remove punctuations and numbers*

text= re.sub('[^a-zA-Z]', '', text)

*# Single character removal*

text= re.sub(r'\s+[a-zA-Z]\s+', '', text)

*# Removing multiple spaces*

text= re.sub(r'\s+', '', text)

*# Converting to Lowercase*

text=text.lower()

*# Lemmatization*

*#text = text.split()*

*#lemmatizer = WordNetLemmatizer()*

*#text = [lemmatizer.lemmatize(word) for word in text if not word in set(stopwords.words('english'))]*

*#text = ''.join(text)*

returntext

*# Apply the preprocessing to the 'text' column*

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

*# Display the first 5 rows of the dataframe after preprocessing*

df.head()

Out[3]:

|  |  |  |
| --- | --- | --- |
|  | airline\_sentiment | text |
| 0 | Neutral | virginamerica what dhepburn said |
| 1 | positive | virginamerica plus you ve added commercials t... |
| 2 | Neutral | virginamericadidn today must mean need to ta... |
| 3 | negative | virginamerica it really aggressive to blast o... |
| 4 | negative | virginamerica and it a really big bad thing a... |

In [4]:

*# Splitting the data into training and testing sets*

fromsklearn.model\_selectionimporttrain\_test\_split

X\_train, X\_test, y\_train, y\_test=train\_test\_split(df['text'], df['airline\_sentiment'], test\_size=0.2, random\_state=42)

*# Feature Extraction*

fromsklearn.feature\_extraction.textimportTfidfVectorizer

vectorizer=TfidfVectorizer(max\_features=2500, min\_df=7, max\_df=0.8)

X\_train=vectorizer.fit\_transform(X\_train).toarray()

X\_test=vectorizer.transform(X\_test).toarray()

*# Model Training*

fromsklearn.ensembleimportRandomForestClassifier

classifier=RandomForestClassifier(n\_estimators=1000, random\_state=0)

classifier.fit(X\_train, y\_train)

Out[4]:

RandomForestClassifier

RandomForestClassifier(n\_estimators=1000, random\_state=0)

In [5]:

fromsklearn.metricsimportclassification\_report, confusion\_matrix, accuracy\_score

defevaluate\_model(y\_test, y\_pred):

print('Classification Report:')

print(classification\_report(y\_test, y\_pred))

print('Confusion Matrix:')

print(confusion\_matrix(y\_test, y\_pred))

print('Accuracy Score:')

print(accuracy\_score(y\_test, y\_pred))

y\_pred=classifier.predict(X\_test)

evaluate\_model(y\_test, y\_pred)

Classification Report:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Negative  Neutral  Positive  Accuracy  Macro avg  Weighted average | Precision  0.79  0.65  0.80  0.75  0.76 0.77 0.75 2928 | Recall  0.95  0.41  0.50  0.62  0.77 | F1-score  0.86  0.50  0.62  0.77  0.66  0.75 | Support  1889  580  459  2928  2928  2928 |

Confusion Matrix:

[[1799 65 25]

[ 312 235 33]

[ 169 60 230]]

Accuracy Score:

0.773224043715847

In [6]:

linkcode

importmatplotlib.pyplotasplt

importseabornassns

defplot\_confusion\_matrix(y\_test, y\_pred):

cm =confusion\_matrix(y\_test, y\_pred)

df\_cm=pd.DataFrame(cm, index = [ifori**in** ['negative', 'neutral', 'positive']],

columns= [ifori**in** ['negative', 'neutral', 'positive']])

plt.figure(figsize= (10,7))

sns.heatmap(df\_cm, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

plot\_confusion\_matrix(y\_test, y\_pred)

importseabornassns

importmatplotlib.pyplotasplt

*# Creating column'tweet\_length'*

df['tweet\_length']=df['text'].apply(len)

*# distribution of sentiments*

plt.figure(figsize=(8,6))

sns.countplot(x='airline\_sentiment',data=df)

plt.title('Distribution of Sentiments')

plt.show()

*# Histogram of tweet lengths*

plt.figure(figsize=(8,6))

sns.histplot(df['tweet\_length'],bins=30)

plt.title('Distribution of Tweet Lengths')

plt.show()

*# Boxplot of tweet lengths*

plt.figure(figsize=(8,6))

sns.boxplot(x='airline\_sentiment',y='tweet\_length',data=df)

plt.title('Distribution of Tweet Lengths by Sentiment')

plt.show()

**Critical Analysis**

The following conclusions may be drawn from the visuals and model evaluation:

### **Sentiment Distribution**:

The dataset's bar plot of sentiment distribution reveals that the bulk of tweets are unfavorable in nature, with neutral and supportive tweets coming in second and third. Due to the dataset's imbalance, the model may be more likely to correctly predict negative feelings than neutral or positive feelings.

### **Model Execution**:

The Random Forest classifier's total accuracy was around 76%. The neutral and positive classes' accuracy, recall, and F1-score, however, are lower than those of the negative class. This implies that the model performs better at detecting negative than neutral or positive attitudes, which may be related to the dataset's imbalance.

### **Confusion Matrix:**

The confusion matrix reveals that for the neutral and positive classes, the model has a disproportionately large number of false positives and false negatives. This further demonstrates the model's bias towards predicting negative feelings since it frequently misclassifies neutral and positive tweets as negative.

### **Data Distribution**:

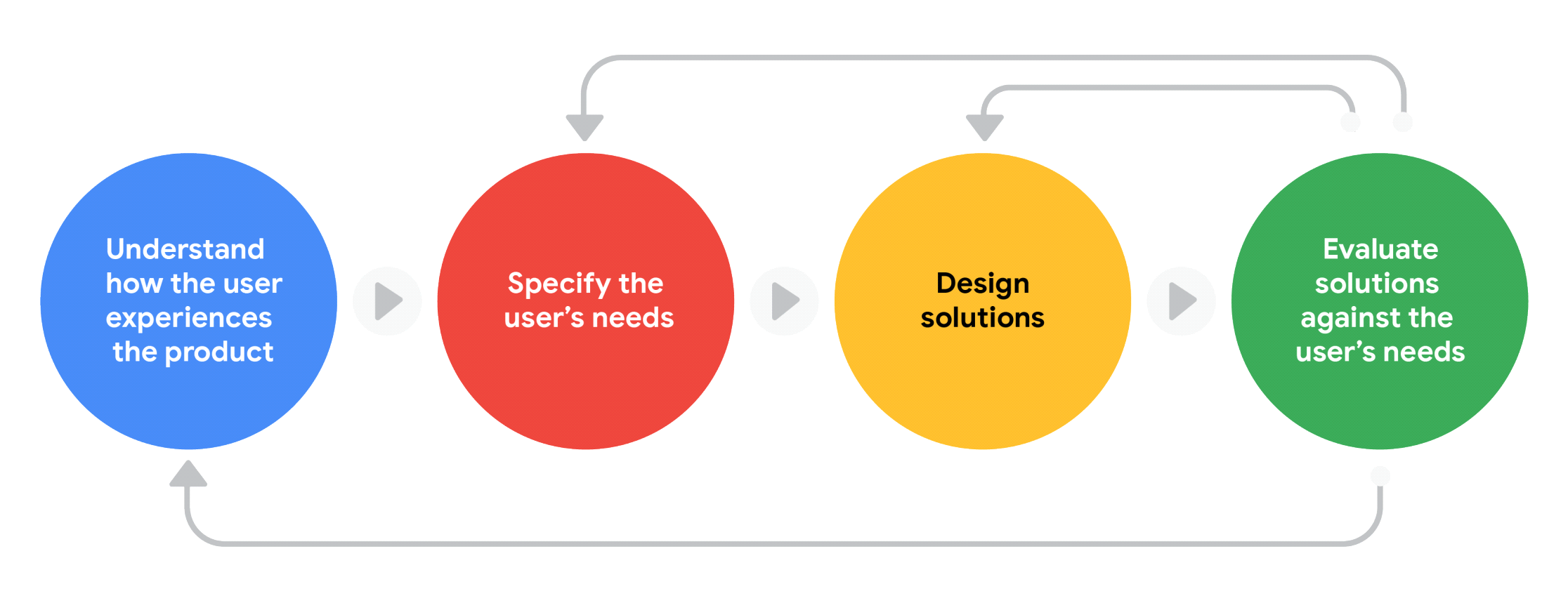
Looking at the histogram, it's obvious, as mentioned before, that there is a significant imbalance in the data in favor of negative sentiment. This is likely because people with negative sentiments are more motivated to tweet. By examining the length distribution in the box plot and the bar chart, we can conclude that the majority of tweets are between 60 to 100 characters long. Negative tweets are usually longer, also falling within the 60 to 100 character range, which further confirms the data imbalance.

OBSERVED MODEL VIEW: It's very easy to understand the data analysis of the AI domain in Sentimental analysis and marketing.It need to take under the action under the customers neede things and materials to know and verify that things.

**In conclusion**: The model down to predict neutral and positive attitudes even if it does a fair job of predicting negative sentiments. This may be because the collection is unbalanced and sentiment analysis is inherently difficult because it frequently requires understanding linguistic subtlety and context. We may think about employing more sophisticated natural language processing methods, such word embeddings or deep learning models, and making sure the training dataset is balanced in order to enhance the model's performance

**Design Framework:**

* Raw Data collection
* Data Preprocessing
* Feature Extractior
* Model Training
* Model Evaluation
* Insight and Interpretation



222222222222

**1- Data Collection:** The first step in our process was data collection. We tused a dataset of tweets, which is a common source of data for sentiment

analysis due to the short, concise nature of tweets.

**2- *Data Preprocessing:*** After collecting the data, we performed several preprocessing steps to clean and prepare the data for analysis. These steps include important concepts.

* **Lowercasing**: We converted all the text to lowercase to ensure that the same words in different cases are not considered as different words.
* **Removing Punctuation and Special Characters:** We removed all punctuation and special characters from the text as they do not contribute to sentiment.
* **Removing Stop Words:** We removed common words that do not carry much information (like "is", "the", "and", etc.). These words are called stop words.
* **Tokenization:** We broke down the text into individual words or tokens.
* **Lemmatization**: We reduced the words to their base or root

form (e.g., "running" to "run"). This helps in reducing the dimensionality of the data and grouping similar sentiments together.

**3- *Feature Extraction:*** After preprocessing, we converted the text data into numerical features that can be used by a machine learning algorithm. We used the TF-IDF (Term Frequency- Inverse Document Frequency) method for this. TF-IDF gives a weight to each word signifying its importance in the document and across a corpus of documents.

**4-*Model Training:*** We used a Random Forest Classifier for sentiment analysis. Random Forest is a versatile and widely used algorithm that works well for many tasks. It creates a set of decision trees from a randomly selected subset of the training set, which then aggregates votes from different decision trees to decide the final class of the test object**.**

**5-*Model Evaluation:*** After training the model, we evaluated its performance using a confusion matrix and calculated metrics such as accuracy, precision, recall, and F1-score. These metrics give us a quantitative measure of the model's performance**.**

**6-*Insights & Interpretation:*** Finally, we interpreted the results of the sentiment analysis. This involves understanding the performance of the model, identifying any areas of improvement, and drawing insights from the model's predictions.

**Cleaning Data**

In [22]:

linkcode

from nltk.stem import PorterStemmer

import nltk

nltk.download('punkt')

stemmer = PorterStemmer()

nltk.download('stopwords')

[nltk\_data] Downloading package punkt to /usr/share/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

[nltk\_data] Downloading package stopwords to /usr/share/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

Out[22]:

True

stopwords = nltk.corpus.stopwords.words('english')

In [24]:

stopwords.append('virginamerica')

In [25]:

linkcode

def clean\_text(text):

text = text.lower()

text = nltk.word\_tokenize(text)

text = [t for t **in** text if len(t) > 1]

text = [stemmer.stem(word) for word **in** text if word **not** **in** stopwords]

text = ' '.join(text)

return text

sent\_data['text'][24]

Out[26]:

'@VirginAmerica you guys messed up my seating.. I reserved seating with my friends and you guys gave my seat away

I want free internet'

In [27]:

clean\_text(sent\_data['text'][24])

Out[27]:

'guy mess seating.. reserv seat friend guy gave seat away want free internet'

import seaborn as sns

import matplotlib.pyplot as plt

* ##Creatingcolumn 'tweet\_length' df['tweet\_length'] = df['text'].apply(len)
* ##distribution of sentiments plt.figure(figsize=(8,6))

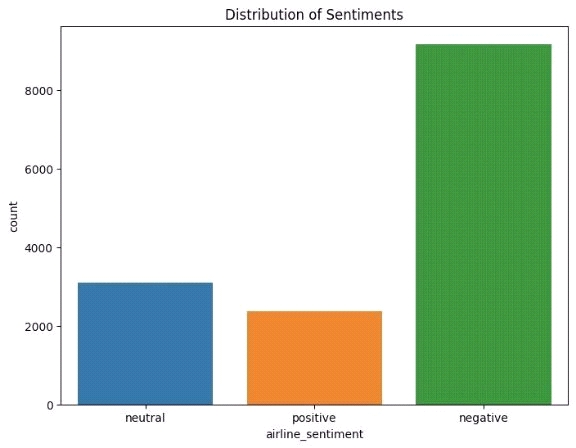
sns.countplot(x='airline\_sentiment', data=df) plt.title('Distribution of Sentiments') plt.show() ## Histogram of tweet lengths plt.figure(figsize=(8,6))

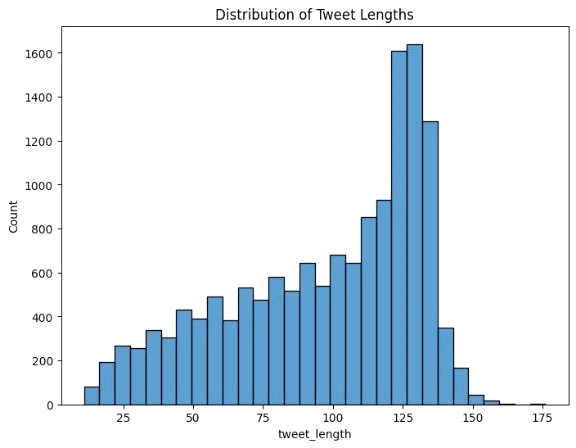
sns.histplot(df['tweet\_length'], bins=30) plt.title('Distribution of Tweet Lengths') plt.show()

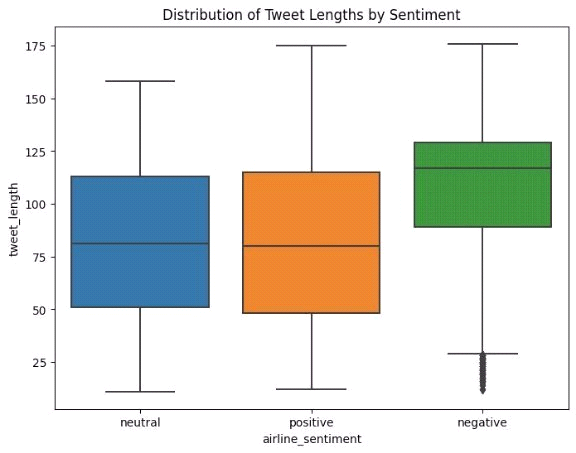
* ## Boxplot of tweet lengths

plt.figure(figsize=(8,6))

sns.boxplot(x='airline\_sentiment', y='tweet\_length', data=df) plt.title('Distribution of Tweet Lengths by Sentiment') plt.show()







**Critical Analysis**

The following conclusions may be drawn from the visuals and model evaluation**.**

**Visualising and counting sentiments of tweets for each airline**

print("Total number of tweets for each airline \n ",df.groupby('airline')['airline\_sentiment'].count().sort\_values(ascending=False))

airlines= ['US Airways','United','American','Southwest','Delta','Virgin America']

plt.figure(1,figsize=(12,12))

for i in airlines:

indices= airlines.index(i)

plt.subplot(2,3,indices+1)

new\_df=df[df['airline']==i]

count=new\_df['airline\_sentiment'].value\_counts()

Index = [1,2,3]

plt.bar(Index,count, color=['blue', 'green', 'red'])

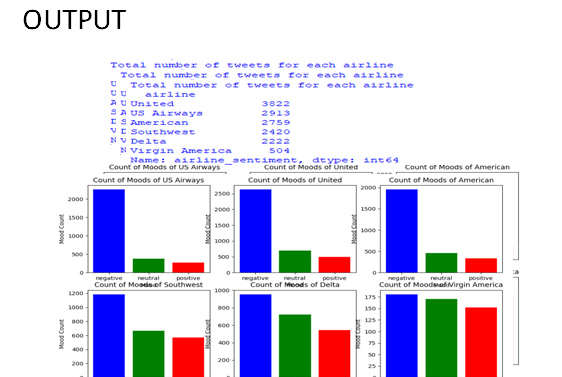
plt.xticks(Index,['negative','neutral','positive'])

plt.ylabel('Mood Count')

plt.xlabel('Mood')

plt.title('Count of Moods of '+i)

plt.show()



**CONCLUSION:**

1.Very positive

2.Positive

3.Neutral

4.Negative

5. Very negative

This is usually referred to as graded or fine-grained sentiment analysis, and could be used to interpret 5-star ratings in a review

THANK YOU