SENTIMENTALANALYSIS

Dhasaradh A M : 720921104028

Final Development

Project: **Sentimental Analysis**

Sentiment Analysis

In this notebook, we will implement a sentiment analysis model. We will use the Twitter US Airline Sentiment dataset for this task.

Corpus Introduction:

The corpus used in this assignment is the "Twitter US Airline Sentiment" dataset. This dataset is a collection of tweets about various US airlines, scraped from Twitter in February 2015. The tweets have been pre-classified as positive, negative, or neutral, and negative tweets have been further categorized by the reason for the negative sentiment, such as "late flight" or "rude service". The dataset contains around 15,000 entries.

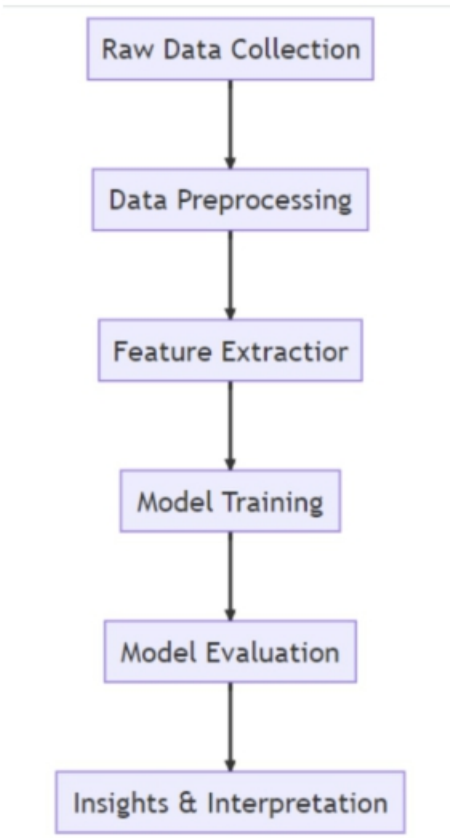
Objective:

The objective of this sentiment analysis task is to determine the polarity of public opinion about different US airlines. By analyzing the sentiment expressed in these tweets, we aim to understand how the public perceives these airlines and what specific issues or aspects contribute to these perceptions.

Scope:

The scope of this sentiment analysis task is to classify the sentiment expressed in each tweet as either positive, negative, or neutral. This will involve processing and analyzing the text of the tweets to identify sentiment-bearing phrases and determine their polarity. The analysis will also consider the specific reasons given for negative sentiments, providing deeper insight into the issues that lead to negative public opinion. The results of this analysis could be used to inform decision-making and strategy for airlines, helping them to address public concerns and improve their service

Design Framework:



1. Data Collection:

The first step in our process was data collection. We used 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 includ:

· **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.

In [1]:

```
import pandas as pd

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix,
classification_report
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
import nltk
nltk.download('stopwords')
nltk.download('wordnet')

# Load the dataset
df = pd.read_csv('/kaggle/input/twitter-airline-
sentiment/Tweets.csv')

# Display the first 5 rows of the dataframe
```

```
df.head()
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"

[nltk_data] Downloading package stopwords to /usr/share/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /usr/share/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out [1]:

	Tweetid	Airline_settlement	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	negativereason_gold	text	tweet_coord	tweet_created	tweet_location	user_timezone
0	570306133677760513	neutral	1,0000	NaN	NaN	Virgin America	NaN	cairdin	NaN	0	@VirginAmerica What @dhepburn said.	NaN	2015-02-24 11:35:52 -0800	NaN	Eastern Time (US & Canada)
1	570301130888122368	positive	0,3486	NaN	0,0000	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica plus you've added commercials t...	NaN	2015-02-24 11:15:59 -0800	NaN	Pacific Time (US & Canada)
2	57030108367281357	neutral	0,6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN	0	@VirginAmerica I didn't today... Must mean I n...	NaN	2015-02-24 11:15:48 -0800	Lets Play	Central Time (US & Canada)
3	570301031407624196	negative	1,0000	Bad Flight	0,7033	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica it's really aggressive to blast...	NaN	2015-02-24 11:15:48 -0800	NaN	Pacific Time (US & Canada)
4	570300817074462722	negative	1.0000	Can't Tell	1,0000	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica and it's a really big bad thing...	NaN	2015-02-24 11:14:45 -0800	NaN	Pacific Time (US & Canada)

In [3]:

```
df.isnull().sum()
```

Out [3]:

```
tweet_id                                 0
airline_sentiment                        0
airline_sentiment_confidence
0negativereason
negativereason_confidence             4118
airline                                  0
airline_sentiment_gold               14600
name                                     0
negativereason_gold                  14608
retweet_count                            0
text                                     0
tweet_coord                          13621
tweet_created                            0
tweet_location                        4733
user_timezone                         4820
dtype: int64
```

In [4]:

```
df.shape
```

Out[4]:

```
(14640, 15)
```

In [5]:

```
df.columns
```

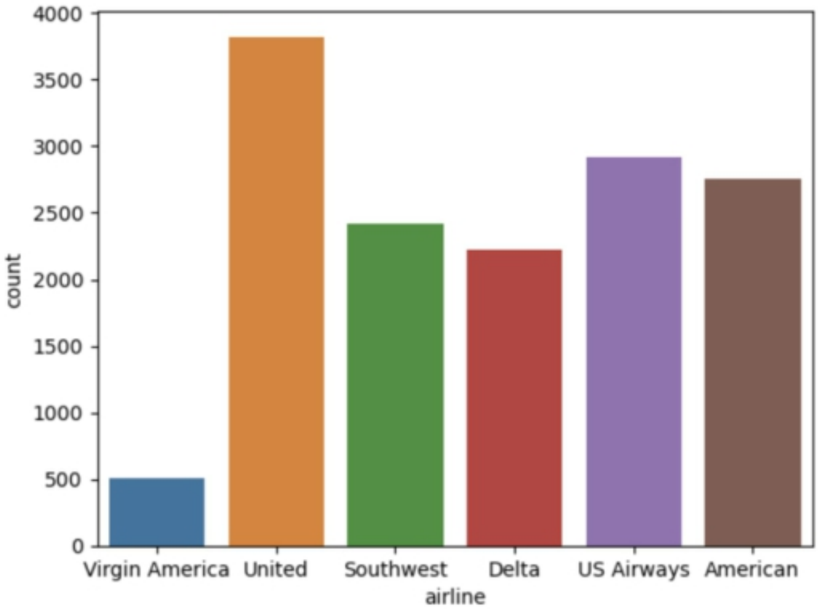
Out[5]:

```
Index(['tweet_id', 'airline_sentiment',
'airline_sentiment_confidence',
```

```
        'negativereason', 'negativereason_confidence',
'airline',
        'airline_sentiment_gold', 'name',
'negativereason_gold',
        'retweet_count', 'text', 'tweet_coord',
'tweet_created',
        'tweet_location', 'user_timezone'],
       dtype='object')
```

In [6]:

```
sns.countplot(data=df,x="airline")
plt.show()
```



In [7]:

```
       df.drop(['tweet_id', 'airline_sentiment_confidence',
        'negativereason', 'negativereason_confidence',
'airline',
        'airline_sentiment_gold', 'name',
'negativereason_gold',
```

```
        'retweet_count', 'tweet_coord', 'tweet_created',
        'tweet_location', 'user_timezone'],
axis=1,inplace=True)
```

In [8]:

```
        df.isnull().sum()
```

out[8]:

```
        airline_sentiment     0
        text                  0
        dtype: int64
```

In [9]:

```
        df.drop_duplicates(inplace=True)
```

In [10]:

```
        df.dropna(inplace=True
```

In [11]:

```
         df.shape
```

Out[11]:

```
         (14452, 2)
```

lets starts

In [12]:

In [12]:

```
         df
```

Out[12]:

	airline_sentiment	text
0	neutral	@VirginAmerica What @dhepburn said.
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...
...	...	...
14635	positive	@AmericanAir thank you we got on a different f...
14636	negative	@AmericanAir leaving over 20 minutes Late Flig...
14637	neutral	@AmericanAir Please bring American Airlines to...
14638	negative	@AmericanAir you have my money, you change my ...
14639	neutral	@AmericanAir we have 8 ppl so we need 2 know h...

14452 rows × 2 columns

In [13]:

```
#percentage
df[“airline_sentiment"].value_counts()len(df["airline_sentiment"])
```

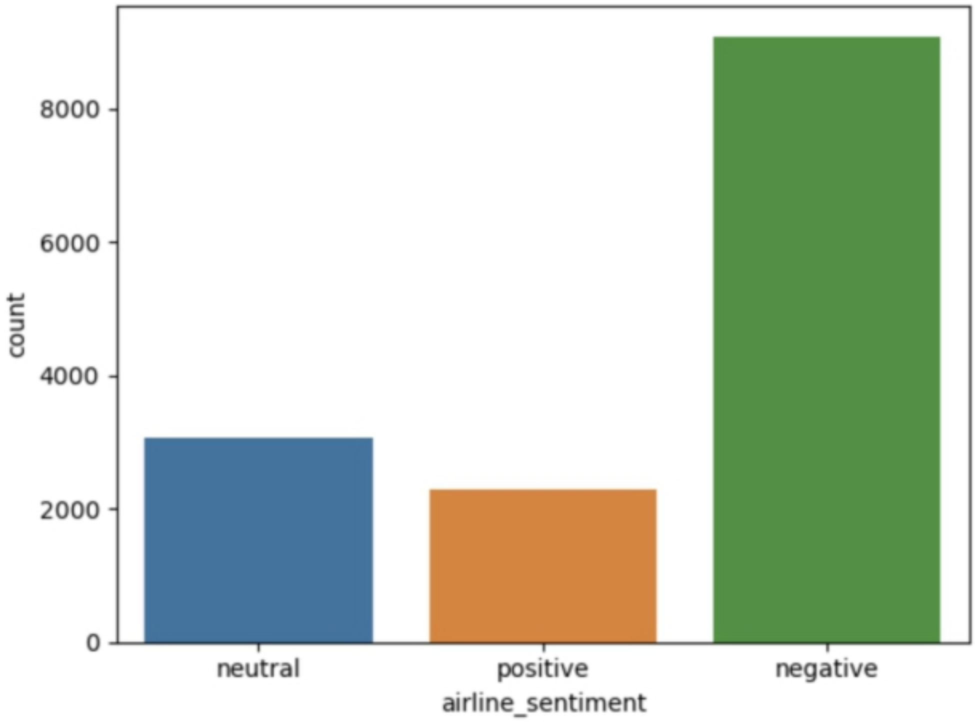
Out[13]:

```
negative    0.628771
neutral     0.212220
positive    0.159009
Name: airline_sentiment, dtype: float64
```

In [14]:

```
sns.countplot(data=df, x=“airline_sentiment")
plt.show()
```

In [15]:




```
#maxi length of a tweet in a text
maxi_length=df.text.apply(len)
maxi_length.max()
```

Out[15]:

186

In [16]:

```
mini_length=df.text.str.len()
mini_length.min()
```

Out[16]:

12

In [17]:

```
pd.DataFrame(df.text.apply(len).describe())
```

Out[17]:

	text
count	14452.000000
mean	104.118738
std	35.991567
min	12.000000
25%	77.000000
50%	115.000000
75%	136.000000
max	186.000000

Preprocessing

In [18]:

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import pandas as pd

# Create a stemmer instance
stemmer = PorterStemmer()

# Modify the preprocess_text function to include stemming
def preprocess_text(text):
    # Remove special characters, URLs, and user mentions
    text = ' '.join(word for word in word_tokenize(text)
if not word.startswith('@') and word.isalnum())
```

```python
    # Convert to lowercase
    text = text.lower()
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    words = word_tokenize(text)
    text = ' '.join(word for word in words if word not in
stop_words)
    # Apply stemming to each word
    words = word_tokenize(text)
    text = ' '.join(stemmer.stem(word) for word in words)
    return text


# Assuming 'df' is your DataFrame with a 'text' column
df['text'] = df['text'].apply(preprocess_text)
```

In [19]:

```python
df
```

Out[19]:

	airline_sentiment	text
0	neutral	virginamerica dhepburn said
1	positive	virginamerica plu ad commerci experi tacki
2	neutral	virginamerica today must mean need take anoth ...
3	negative	virginamerica realli aggress blast obnoxi ente...
4	negative	virginamerica realli big bad thing
...	...	...
14635	positive	americanair thank got differ flight chicago
14636	negative	americanair leav 20 minut late flight warn com...
14637	neutral	americanair pleas bring american airlin blackb...
14638	negative	americanair money chang flight answer phone su...
14639	neutral	americanair 8 ppl need 2 know mani seat next f...

14452 rows × 2 columns

Sentiment	Numerical Value
negative	0
neutral	1
positive	2

In [20]:

```
df["airline_sentiment"].unique()
```

Out[20]:

```
array(['neutral', 'positive', ‘negative'],
                                  dtype=object).
```

In [21]:

```
mapping = {"negative": 0, 'neutral': 1,
"positive": 2}
df["airline_sentiment"]df[“airline_sentiment"].map(mapping)
```

In [22]:

```
X=df["text"]
y=df["airline_sentiment"]
```

In [23]:

```
from sklearn.feature_extraction.text import
CountVectorizer
from sklearn.model_selection import
train_test_split
from sklearn.naive_bayes import MultinomialNB
```

In [24]:

```
count_vectorizer = CountVectorizer()
X_counts = count_vectorizer.fit_transform(X)
feature_names = count_vectorizer.get_feature_names_out()
```

In [25]:

```
X=pd.DataFrame(X_counts.toarray())
```

In [26]:

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.30,
random_state=50)
```

In [27]:

```
X=pd.DataFrame(X_counts.toarray())
```

In [28]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=50)
```

In [29]:

```
print(f'shape of X_train {X_train.shape} & shape of y_train {y_train.shape}')
print(f'shape of X_test {X_test.shape} & shape of y_test {y_test.shape}')
```

```
shape of X_train (10116, 10627) & shape of y_train (10116,)
shape of X_test (4336, 10627) & shape of y_test (4336,)
```

In [30]:

```
bayes = MultinomialNB(alpha=0.45833).fit(X_train, y_train)
```

In [31]:

```
bayes_pred=bayes.predict(X_test)
```

In [32]:

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

In [33]:

```
accuracy_score(y_test,bayes_pred)
```

Out[33]:

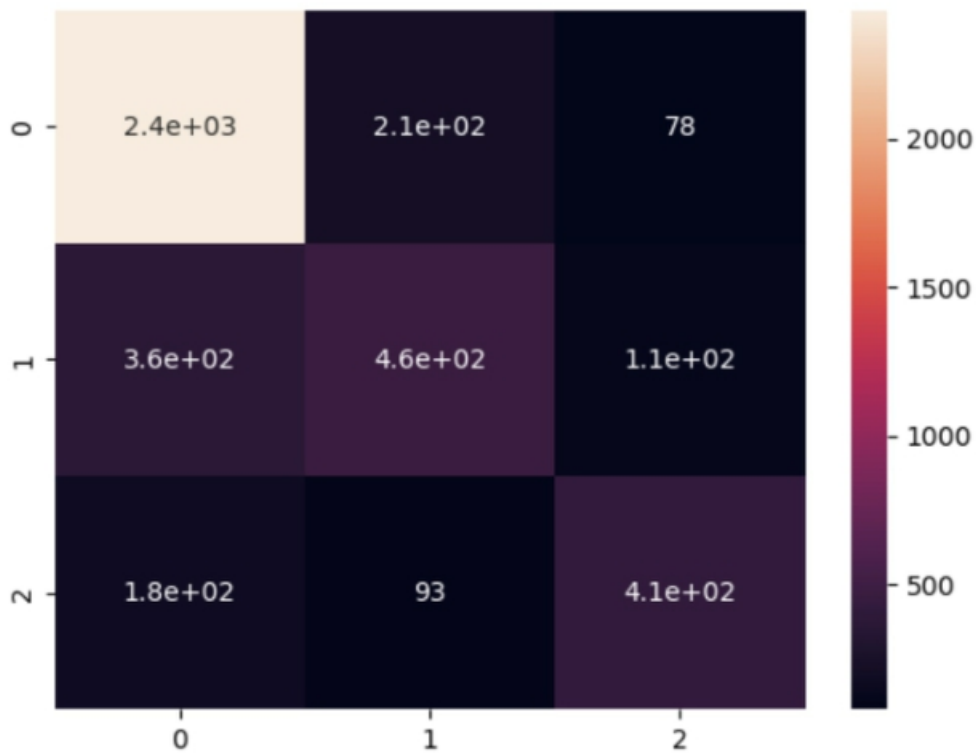
0.761070110701107

In [34]:

```
print(classification_report(y_test, bayes_pred))
```

In [35]:

```
sns.heatmap(confusion_matrix(y_test,
bayes_pred), annot=True)
plt.show()
```



In [36]:

```
import lightgbm as lgb
```

In [37]:

```
light_model =
lgb.LGBMClassifier(n_estimators=200,
learning_rate=.1,reg_lambda=0,
  n_jobs=-1).fit(X_train, y_train)
```

In [38]:

```
light_model_predict = light_model.predict(X_test)
```

In [39]:

```
accuracy_score(y_test, light_model_predict)
```

Out[39]:0.7822878228782287

In [40]:

```
print(classification_report(y_test,
light_model_predict))
```

```
      precision    recall  f1-score   support
```

	precision	recall	f1-score	support
0	0.83	0.90	0.87	2723
1	0.64	0.54	0.58	934
2	0.72	0.64	0.68	679
accuracy			0.78	4336
macro avg	0.73	0.69	0.71	4336
weighted avg	0.77	0.78	0.78	4336

In [41]:

```
sns.heatmap(confusion_matrix(y_test,
light_model_predict), annot=True)
plt.show()
```