The Assignment: Airline Tweet Sentiment Analysis

GROUP 2

Group Members

1. Isaac Mugasa
2. Keeya Tracy
3. Ogwang Samson
4. Maxensia Kasukukuru
5. Mboyo Botuli Pacic
6. Atukunda Pearl Gilian



Data Set Description

Dataset Details

Name: Kaggle - Data Science Salaries 2023 Kaggle

Source: Kaggle - Data Science **Salaries** 2023 Kaggle

Format: Csv

Problem Recap

Problem: We are applying machine learning to analyze the sentiment of tweets about U.S. airlines.

Goal: To classify whether a tweet expresses a positive or negative opinion.

Type of ML: Supervised Learning, specically a Binary Classication problem.

Use Case: This can help airlines monitor customer feedback and improve their services.

Target and Features

Target Variable (y)

airline_sentiment (converted to binary: 1 = positive, 0 = negative)

Input Features (X)

cleaned_text: The cleaned version of the original tweet text (after removing links, punctuation, stopwords, etc.)

Task Type

Classification

Justification

The airline_sentiment label directly reacts the outcome we want the model to predict — the opinion expressed in the tweet.

Dataset Features

Feature Name	Data Type	Description	Example Value
text	Text	The original tweet	"Southwest has the best crew ever!"
cleaned_text	Text	Preprocessed version of the tweet	"southwest best crew ever"
airline_sentiment	Categorical	Sentiment label (positive or negative)	"positive"
label	**Numeric**	Binary encoded sentiment (1 = positive, 0 = negative)	1
airline	Categorical	Airline being mentioned	"Southwest"
tweet_created	Datetime	When the tweet was created	2015-02-24 11:35:00

Data Import and Initial Setup

```
import pandas as pd  # For handling data
import nltk as nltk  # natural language tooltik
nltk.download('stopwords') # downloads updated english stop words
from nltk.corpus import stopwords
import re # For regular expressions (used to clean text)
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

```
[2] tweetsdf=pd.read_csv('https://datascience.timelessbranch.com/Tweets.csv')

[3] tweetsdf.head()
```

```
tweetsdf.head()
```

index	tweet_id	airline_sentiment	airline_sentiment_confidence	negativereason	negativereason_confidence	airline	airline_sentiment_gold	name	negativereason_gold	retweet_count	text
0	570306133677760513	neutral	1.0	NaN	NaN	Virgin America	NaN	cairdin	NaN	0	@VirginAmerica What @dhepburn said.
1	570301130888122368	positive	0.3486	NaN	0.0	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica plus you've added commercials to the experience... tacky.
2	570301083672813571	neutral	0.6837	NaN	NaN	Virgin America	NaN	yvonnalynn	NaN	0	@VirginAmerica I didn't today... Must mean I need to take another trip!
3	570301031407624196	negative	1.0	Bad Flight	0.7033	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica it's really aggressive to blast obnoxious "entertainment" in your guests' faces & they have little recourse
4	570300817074462722	negative	1.0	Can't Tell	1.0	Virgin America	NaN	jnardino	NaN	0	@VirginAmerica and it's a really big bad thing about it

Exploratory Data Analysis (EDA)

Summary Statistics

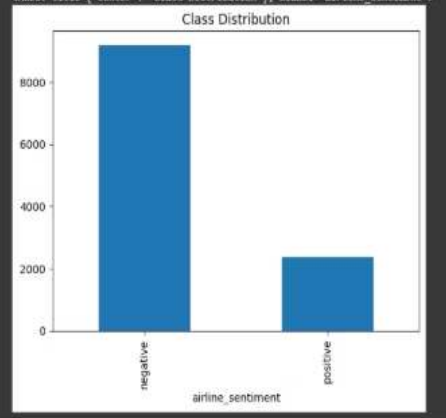
keep only the features (columns that we need for the model)

```
tweetsdf = tweetsdf[['text', 'airline_sentiment']] # we are only keeping what we need which is the actual tweet(text) and airline sentiment
tweetsdf = tweetsdf[tweetsdf['airline_sentiment'] != 'neutral'] #removes the neutral lable, we only need positive and negative
tweetsdf = tweetsdf.reset_index(drop=True) # resets the index to start from 0
```

Showing class distribution of sentiment (positive and negative)

```
# Show class distribution to check for balance
print(tweetsdf['airline_sentiment'].value_counts())
tweetsdf['airline_sentiment'].value_counts().plot(kind='bar', title='Class Distribution')
```

```
airline_sentiment
negative    9178
positive    2363
Name: count, dtype: int64
<Axes: title={'center': 'Class Distribution'}, xlabel='airline_sentiment'>
```



Observation: The dataset is imbalanced toward negative tweets.

The bar shows the data is imbalanced, we are to apply the stratify parameter to keep the sentiment ratios in the training and testing data so that the outcome is not affected by the imbalance in the data.

```
[ ]  # Check for missing values
     print(tweetsdf.isnull().sum())
```

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

Checking for any missing values in the data

Data Cleaning Process

Data cleaning, removing unwanted characters and symbols.

```
[ ]  stop_words = set(stopwords.words('english')) # This Loads a list of common English words like “the”, “and”, “is” that don’t add meaning – these are removed later
```

```
[ ]  # removing unwanted text patterns using regular expressions and storing the clean text in (clean_text)
     def clean_text(text):
         text = text.lower() # lowercase everything
         text = re.sub(r'http\S+', '', text) # remove links
         text = re.sub(r'@\w+', '', text) # remove mentions (@user)
         text = re.sub(r'#\w+', '', text) # remove hashtags
         text = re.sub(r'[^\w\s]', '', text) # remove punctuation
         text = re.sub(r'\d+', '', text) # remove numbers
         text = ' '.join(word for word in text.split() if word not in stop_words)
         return text  #  removes all stopwords from the tweet
```

Feature Selection

Selected Feature:

cleaned_text (text to be converted to numerical form using **CountVectorizer**)

Justication:

It holds the core information (words) from which sentiment is expressed.

Other features like airline or **tweet_created** were dropped because:

- They don't contain strong sentiment-bearing information.
- Could introduce bias or noise.

```
[ ] # this applies the cleaning function to every tweet in the dataset and stores the cleaned result in a new column called cleaned_text.
    tweetsdf['cleaned_text'] = tweetsdf['text'].apply(clean_text)
```

```
# confirm the changes
tweetsdf.head()
```

	text	airline_sentiment	cleaned_text
0	@VirginAmerica plus you've added commercials t...	positive	plus youve added commercials experience tacky
1	@VirginAmerica it's really aggressive to blast...	negative	really aggressive blast obnoxious entertainmen...
2	@VirginAmerica and it's a really big bad thing...	negative	really big bad thing
3	@VirginAmerica seriously would pay $30 a fligh...	negative	seriously would pay flight seats didnt playing...
4	@VirginAmerica yes, nearly every time I fly VX...	positive	yes nearly every time fly vx ear worm wont go ...

Data Preparation for Model Training

Binary encoding of the nominal categories (positive and negative)

```
# This creates a numerical label column: positive → 1 negative → 0 This is necessary because ML models don't understand words — they need numbers.
tweetsdf['label'] = tweetsdf['airline_sentiment'].map({'positive': 1, 'negative': 0})
```

```
# Displays the cleaned text and label column — now ready for training
tweetsdf[['cleaned_text', 'label']].head()
```

	cleaned_text	label
0	plus youve added commercials experience tacky	1
1	really aggressive blast obnoxious entertainmen...	0
2	really big bad thing	0
3	seriously would pay flight seats didnt playing...	0
4	yes nearly every time fly vx ear worm wont go ...	1

Encode **cleaned_text** feature(input) into a numerical matrix using a bow technique.

```
# Convert cleaned text into numerical features using bag-of-words technique(creates a numerical matrix)
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(tweetsdf['cleaned_text'])  # Feature matrix
y = tweetsdf['label']  # Target vector
```

```
# confirm the conversion of tweets to numeric
print("Shape of X:", X.shape)
```

```
Shape of X: (11541, 9649)
```

```
# Split data into 80% training and 20% testing sets
# This ensures both the training and test sets will have the same ratio of positive and negative tweets as the original dataset.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y #  ensures that both the training and testing sets maintain the same ratio of classes (positive vs. negative) as in the full dataset.
)

# Display sizes of the splits
print("Training set size:", X_train.shape[0])
print("Test set size:", X_test.shape[0])
```

```
Training set size: 9232
Test set size: 2309
```

What the above code does(split datset for training and testing)

X = all feature vectors (BoW matrix of tweets)

y = sentiment labels (0 = negative, 1 = positive)

test_size=0.2 = 20% of data goes to testing, 80% to training

random_state=42 = ensures reproducibility (the ability of independent researchers to obtain the same or similar results when performing the same experiment or analysis using the same methods and materials as the original study)