TWITTER SENTIMENT ANALYSIS USING NAIVE BAYES CLASSIFIER IN PYTHON

Aim:

- In this case study, we will build, train, test and deploy an Artificial Intelligence (AI) model to predict sentiment from thousands of tweets.
- Sentiment prediction involves understanding of people feelings about a product or service.
- The dataset is taken from the following source:

  https://www.kaggle.com/sid321axn/amazon-alexa-reviews/kernels

Tools used:

- Anaconda, Python, Scikit-learn, Matplotlib, Seaborn

Data:

- Inputs:
  - Twitter tweets (text data)
- Output:
  - Sentiment (0 or 1)

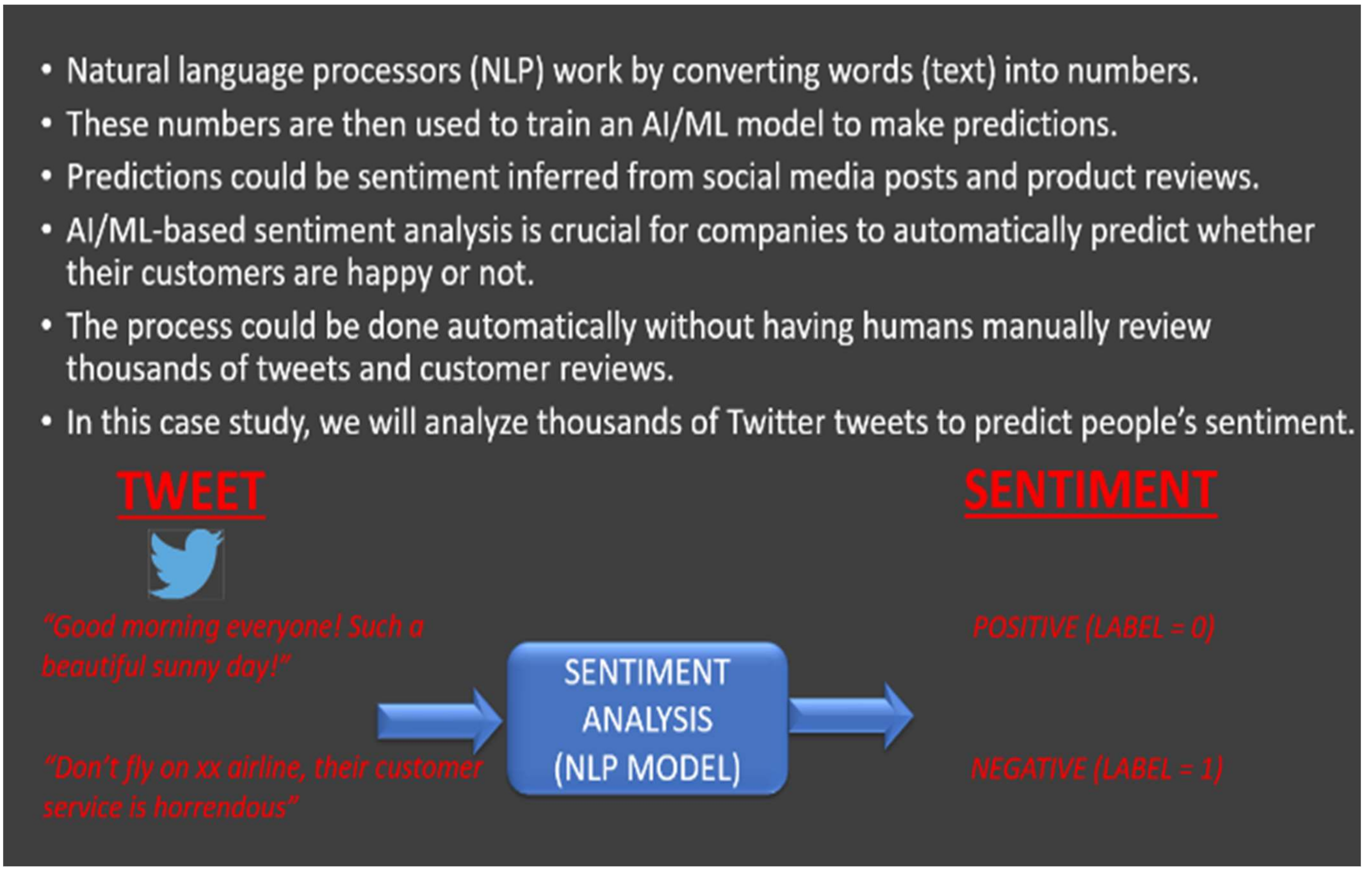
Objectives:

1. Apply python libraries such as Pandas, Matplotlib and Seaborn to analyse and visualize text dataset
2. Perform exploratory data analysis and plot word cloud
3. Perform text data cleaning such as removing punctuation and stop words
4. Understand the concept of count vectorisation and perform tokenisation to tweet text data using Scikit Learn library

5. Understand the theory and intuition behind Naïve Bayes classifiers and learn the difference between prior probability, posterior probability and likelihood
6. Train Naïve Bayes classifier models using Scikit-Learn to perform classification and evaluate its performance using various KPIs

Procedure:

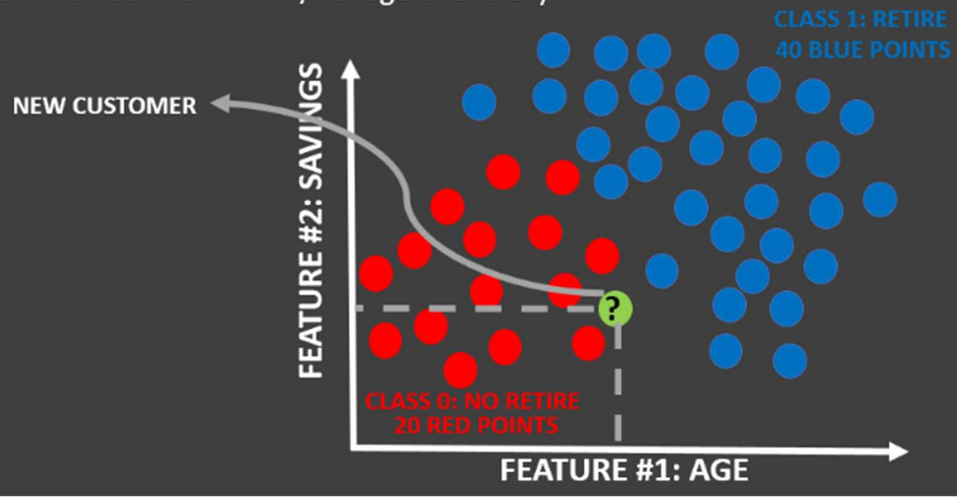
What is NLP:



Naïve Bayes Intuition:

NAÏVE BAYES: INTUITION

- Naïve Bayes is a classification technique based on Bayes' Theorem.
- Let's assume that you are data scientist working major bank in NYC and you want to classify a new client as eligible to retire or not.
- Customer features are his/her age and salary.



NAÏVE BAYES: 1. PRIOR PROBABILITY

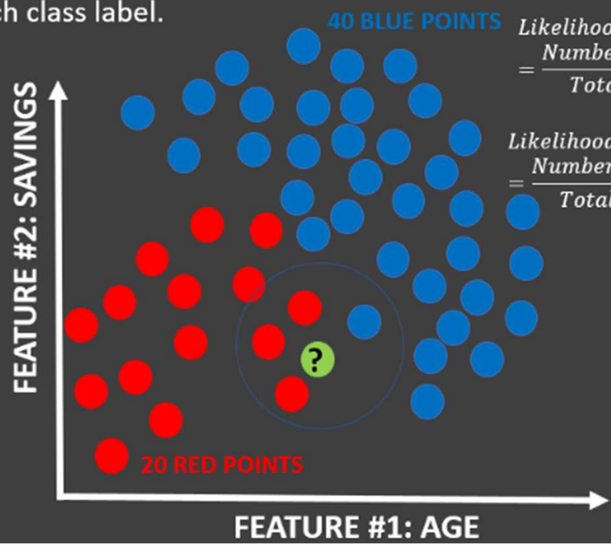
- Points can be classified as RED or BLUE and our task is to classify a new point to RED or BLUE.
- Prior Probability: Since we have more BLUE compared to RED, we can assume that our new point is twice as likely to be BLUE than RED.

$$\text{Prior Probability for RED} = \frac{\text{Number of RED Points}}{\text{Total Number of Points}} = \frac{20}{60}$$

$$\text{Prior Probability for BLUE} = \frac{\text{Number of BLUE Points}}{\text{Total Number of Points}} = \frac{40}{60}$$

NAÏVE BAYES: 2. LIKELIHOOD

- For the new point, if there are more BLUE points in its vicinity, it is more likely that the new point will be classified as BLUE.
- So we draw a circle around the point, then we calculate the number of points in the circle belonging to each class label.

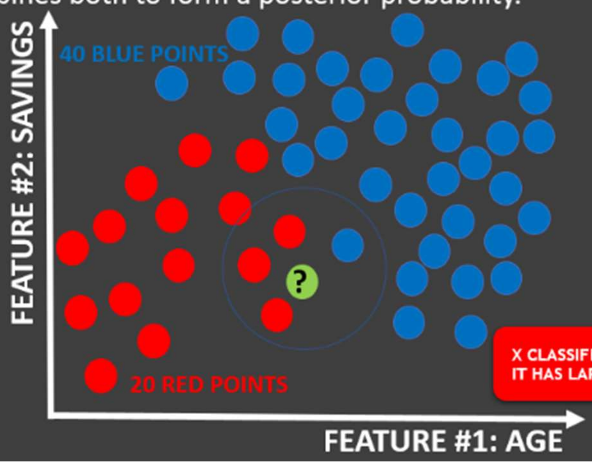


$$\textit{Likelihood of X being RED} = \frac{\textit{Number of RED Points in vicinity}}{\textit{Total Number of RED Points}} = \frac{3}{20}$$

$$\textit{Likelihood of X being BLUE} = \frac{\textit{Number of BLUE Points in vicinity}}{\textit{Total Number of BLUE Points}} = \frac{1}{40}$$

NAÏVE BAYES: 3. POSTERIOR PROBABILITY

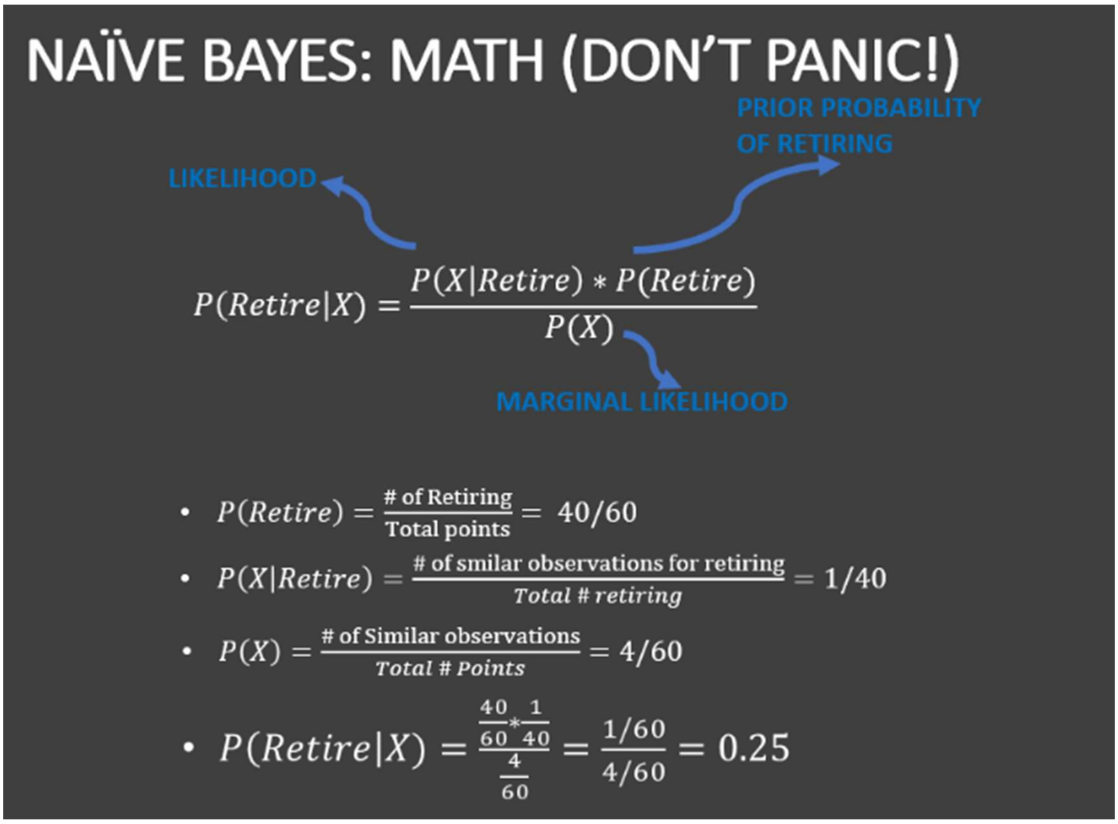
- Let's combine prior probability and likelihood to create a posterior probability.
- Prior probabilities: suggests that X may be classified as BLUE Because there are 2x as much blue points.
- Likelihood: suggests that X is RED because there are more RED points in the vicinity of X.
- Bayes' Rule combines both to form a posterior probability.



$$\textit{Posterior Probability of X being RED} = \textit{Prior Probability of RED} * \textit{Likelihood of X being RED} = \frac{20}{60} * \frac{3}{20} = \frac{1}{20}$$

$$\textit{Posterior Probability of X being BLUE} = \textit{Prior Probability of BLUE} * \textit{Likelihood of X being BLUE} = \frac{40}{60} * \frac{1}{40} = \frac{1}{60}$$

X CLASSIFIED AS RED (NON RETIRING) SINCE IT HAS LARGER POSTERIOR PROBABILITY



Importing the libraries and dataset:

In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
# setting the style of the notebook to be monokai theme
# this line of code is important to ensure that we are able to see the x and y axes clearly
# If you don't run this code line, you will notice that the xlabel and ylabel on any plot is black on black and it will be h
```

In [5]:
```
# Load the data
tweets_df = pd.read_csv(r'C:\Users\jai\Desktop\Twitter Sentiment Analysis\twitter.csv')
```

In [6]:
```
tweets_df
```

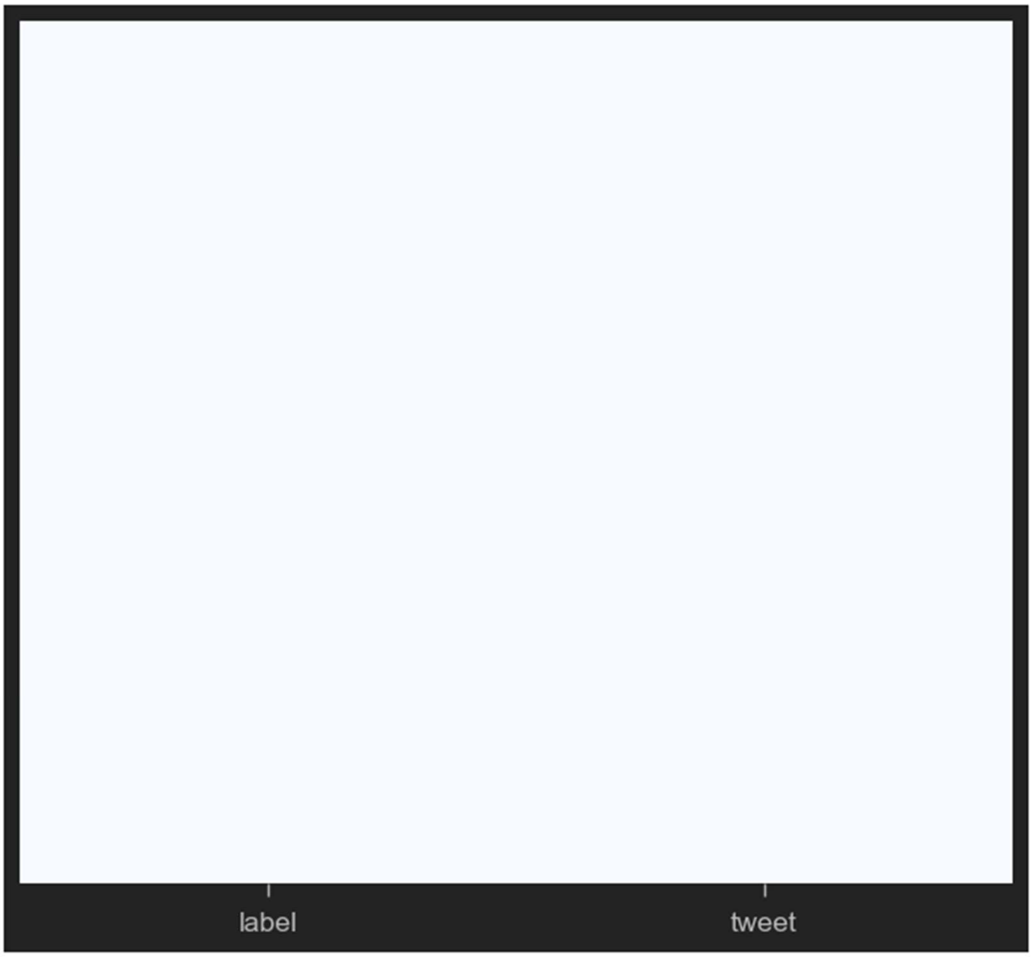
Out[6]:

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation
...	...	...	...
31957	31958	0	ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð□□□ð...
31958	31959	0	to see nina turner on the airwaves trying to...
31959	31960	0	listening to sad songs on a monday morning otw...
31960	31961	1	@user #sikh #temple vandalised in in #calgary,...
31961	31962	0	thank you @user for you follow

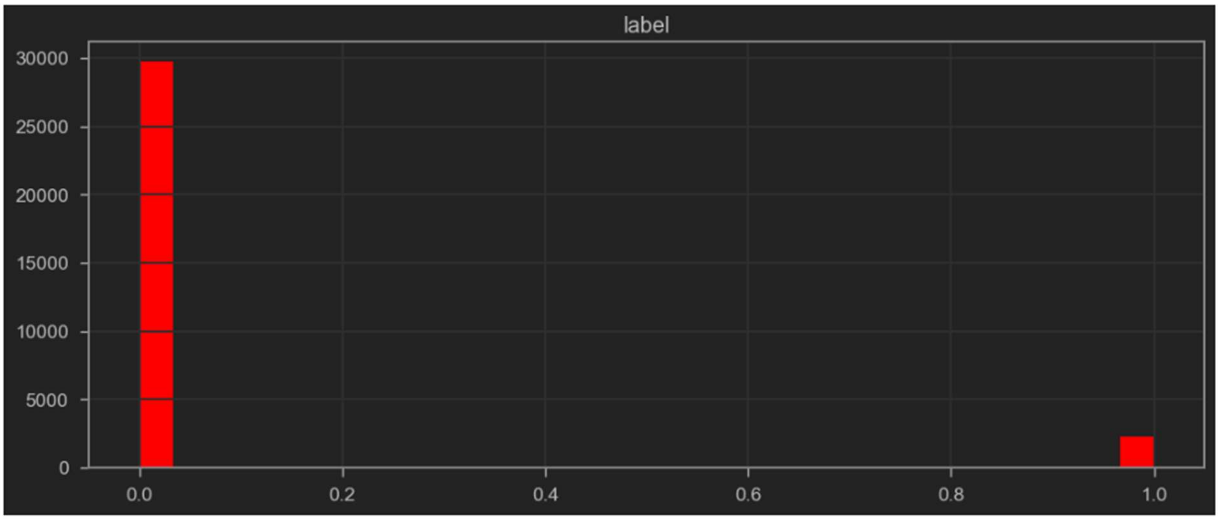
31962 rows × 3 columns

Exploring the dataset:

In [11]:
```
sns.heatmap(tweets_df.isnull(), yticklabels = False, cbar = False, cmap="Blues")
```
Out[11]: <AxesSubplot:>



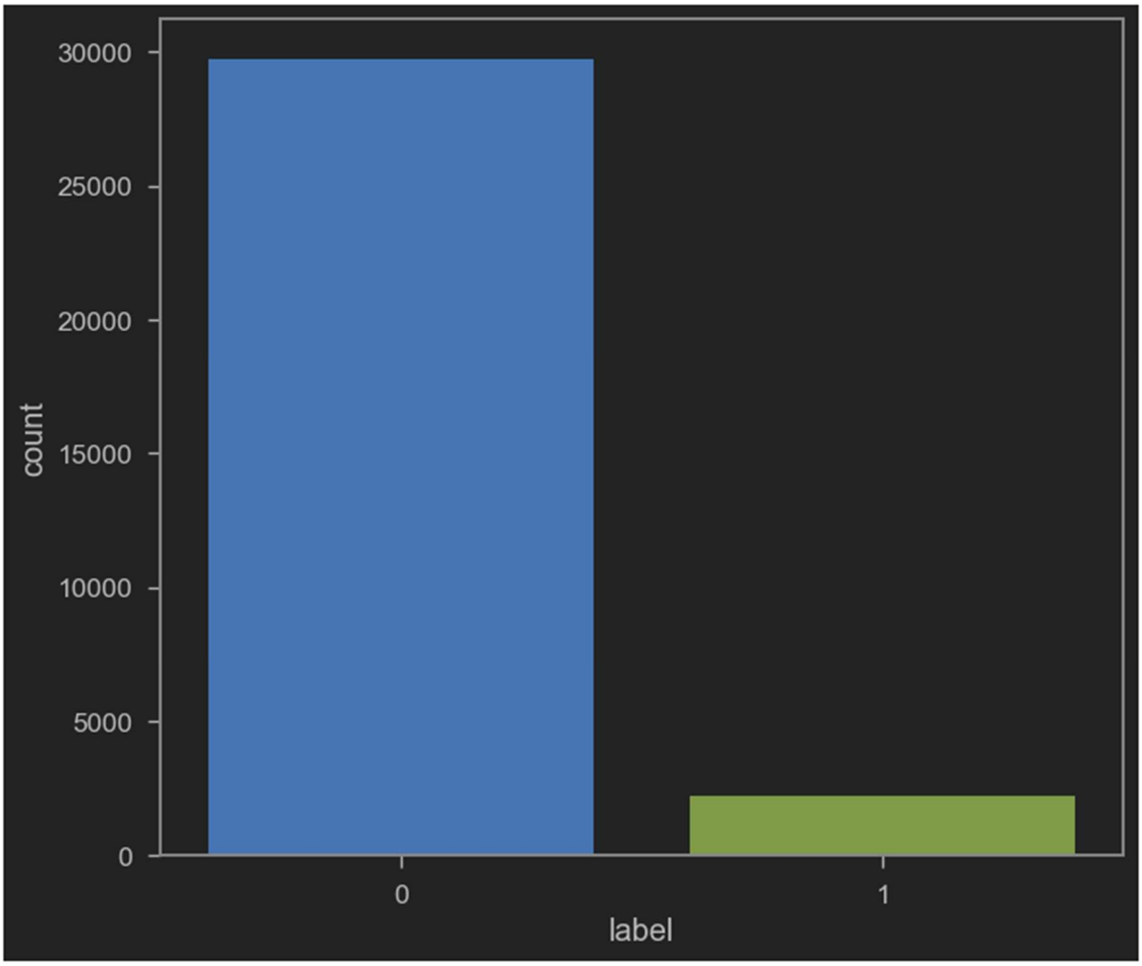
In [12]:
```
# Plot the histogram
tweets_df.hist(bins = 30, figsize = (13,5), color='red')
```
Out[12]: array([[<AxesSubplot:title={'center':'label'}>]], dtype=object)



```
In [13]: # Plot countplot
         sns.countplot(tweets_df['label'], label = 'Count')

C:\Users\jai\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the fol
d arg: x. From version 0.12, the only valid positional argument will be `data`, and passing oth
licit keyword will result in an error or misinterpretation.
  warnings.warn(

Out[13]: <AxesSubplot:xlabel='label', ylabel='count'>
```

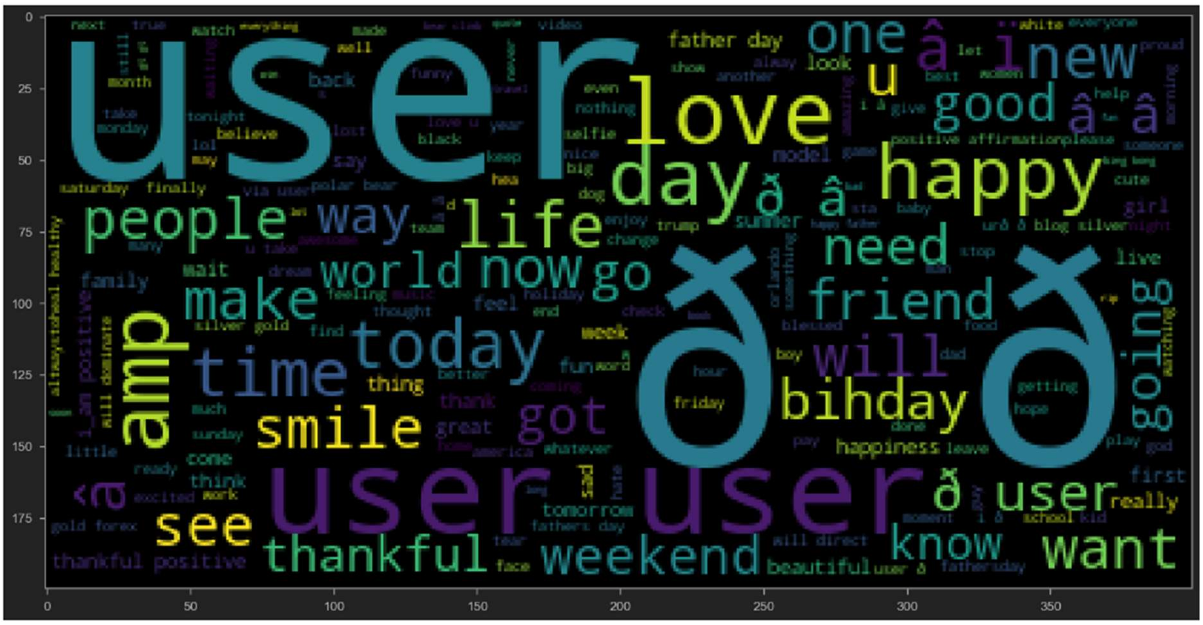


Plot the wordcloud:

```
In [29]: from wordcloud import WordCloud

         plt.figure(figsize=(20,20))
         plt.imshow(WordCloud().generate(sentences_as_one_string))
```

```
Out[29]: <matplotlib.image.AxesImage at 0x13a39cd7c70>
```



Perform Data Cleaning - Remove Punctuation From Text:

```
In [34]: Test_punc_removed = [char for char in Test if char not in string.punctuation]
         Test_punc_removed
```

```
Out[34]: ['G',
          'o',
          'o',
          'd',
          ' ',
          'm',
          'o',
          'r',
          'n',
          'i',
          'n',
          'g',
          ' ',
          'b',
          'e',
          'a',
          'u',
          't',
          'i',
          'f',
          'u',
          'l',
          ' ',
          'p',
          'e',
          'o',
          'p',
          'l',
          'e',
          ' ',
          ' ',
          'I',
          ' ',
          'a',
          'm',
          ' ',
          'h',
          'a',
          'v',
          'i',
          'n',
          'g',
          ' ',
```

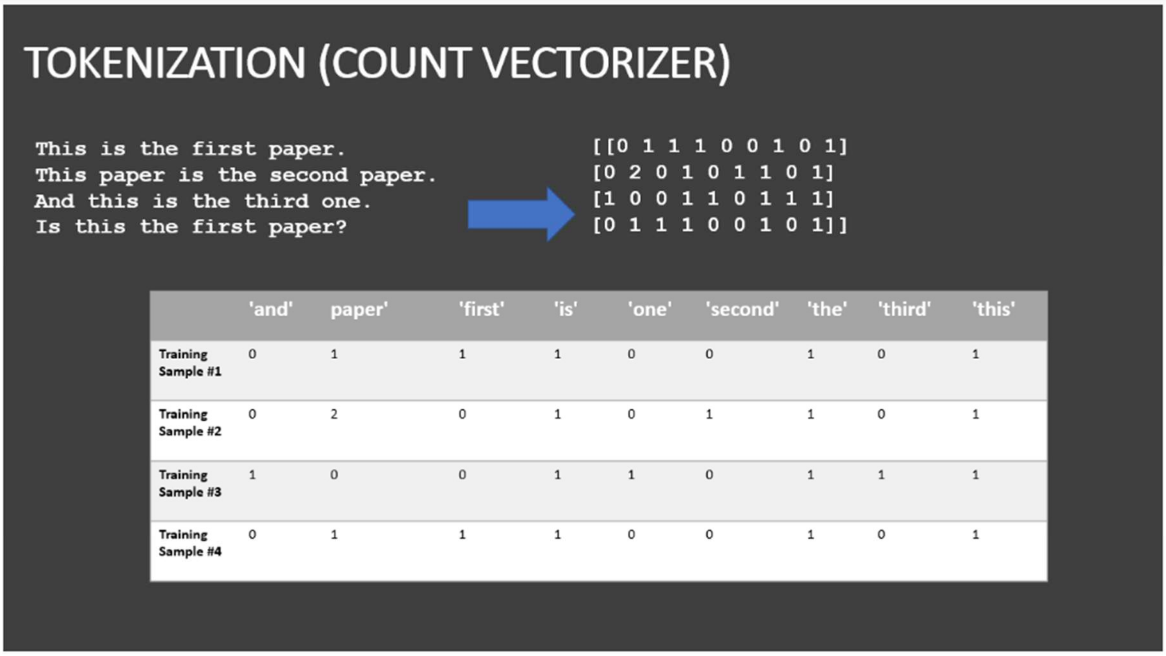
Perform Data Cleaning - Remove Stopwords:

```
In [38]: import nltk # Natural Language tool kit
         nltk.download('stopwords')

         # You have to download stopwords Package to execute this command
         from nltk.corpus import stopwords
         stopwords.words('english')
```

```
Out[38]: ['i',
          'me',
          'my',
          'myself',
          'we',
          'our',
          'ours',
          'ourselves',
          'you',
          "you're",
          "you've",
          "you'll",
          "you'd",
          'your',
          'yours',
          'yourself',
          'yourselves',
          'he',
          'him',
```

Perform Count Vectorization(Tokenization):



	'and'	paper'	'first'	'is'	'one'	'second'	'the'	'third'	'this'
Training Sample #1	0	1	1	1	0	0	1	0	1
Training Sample #2	0	2	0	1	0	1	1	0	1
Training Sample #3	1	0	0	1	1	0	1	1	1
Training Sample #4	0	1	1	1	0	0	1	0	1

```
In [43]: from sklearn.feature_extraction.text import CountVectorizer
         sample_data = ['This is the first paper.','This document is the second paper.','And this is the third one.','Is this the fi
         vectorizer = CountVectorizer()
         X = vectorizer.fit_transform(sample_data)
```

In [44]:
```
print(vectorizer.get_feature_names())
```

```
['and', 'document', 'first', 'is', 'one', 'paper', 'second', 'the', 'third', 'this']
```

```
C:\Users\jai\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning:
recated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use g
  warnings.warn(msg, category=FutureWarning)
```

In [45]:
```
print(X.toarray())
```

```
[[0 0 1 1 0 1 0 1 0 1]
 [0 1 0 1 0 1 1 1 0 1]
 [1 0 0 1 1 0 0 1 1 1]
 [0 0 1 1 0 1 0 1 0 1]]
```

Create A Pipeline To Remove Punctuations, Stopwords And Perform Count Vectorization:

In [46]:
```
# Let's define a pipeline to clean up all the messages
# The pipeline performs the following: (1) remove punctuation, (2) remove stopwords

def message_cleaning(message):
    Test_punc_removed = [char for char in message if char not in string.punctuation]
    Test_punc_removed_join = ''.join(Test_punc_removed)
    Test_punc_removed_join_clean = [word for word in Test_punc_removed_join.split() if word.lower() not in stopwords.words(
    return Test_punc_removed_join_clean
```

In [47]:
```
# Let's test the newly added function
tweets_df_clean = tweets_df['tweet'].apply(message_cleaning)
```

In [48]:
```
print(tweets_df_clean[5]) # show the cleaned up version
```

```
['22', 'huge', 'fan', 'fare', 'big', 'talking', 'leave', 'chaos', 'pay', 'disputes', 'get', 'allshowandnogo']
```

In [49]:
```
print(tweets_df['tweet'][5]) # show the original version
```

```
[2/2] huge fan fare and big talking before they leave. chaos and pay disputes when they get there. #allshowandnogo
```

In [50]:
```
from sklearn.feature_extraction.text import CountVectorizer
# Define the cleaning pipeline we defined earlier
vectorizer = CountVectorizer(analyzer = message_cleaning, dtype = np.uint8)
tweets_countvectorizer = vectorizer.fit_transform(tweets_df['tweet'])
```

In [51]:
```
print(vectorizer.get_feature_names())
```

```
['0', '0000001', '00027', '001', '0035', '00h30', '01', '0115', '0161', '019', '01926889917', '02', '0265', '026680809
9', '02900', '03', '030916', '03111880779', '032', '033', '0345', '039', '04', '045', '04k', '05', '0506823156', '06',
'06052016', '0606', '060616', '0608', '0608wed', '0609', '0610', '061116', '0612', '0613', '0616', '0617', '0618', '0618
saturday7monthscouple', '0618â\x99¡', '0620', '06202016', '0622', '0624', '06Â', '07', '07000', '07040', '07044', '0715
0', '07190', '07400', '07468', '07500', '076', '07788427999', '07800', '07840', '07850', '07870', '07900', '07930', '079
50', '08', '0806', '080616', '088b', '08â\x80¦', '09', '09062016', '0933m', '09600', '0k', '0shares', '0tolerancemovie',
'0ð\x9f\x98¥â\x98¹ï¸\x8f', '1', '10', '100', '1000', '100000', '10003', '10007', '1000gifts', '1000th', '1000x', '1000y
r', '1000â\x82¬', '1001', '1001000s', '10014', '10021', '10025', '10040', '100616', '10064', '100d', '100daysofcode', '1
00daysofpigpaintings', '100daysoftea', '100faces', '100happydays', '100happydaysâ\x80¦', '100happysongs', '100juiceð\x9f
\x8d\x8dð\x9f\x8d\x93ð\x9f\x8d\x87ð\x9f\x8d\x92ð\x9f\x8d\x91ð\x9f\x8d\x8b', '100k', '100ml', '100pm', '100yr', '100â\x80
\x99s', '101', '10125', '1014', '10143hr', '1015', '1017', '1019', '101dalmatians', '101daysofsmiles', '101ð\x9f\x98\x89
ð\x9f\x98\x89ð\x9f\x8e\x89ð\x9f\x8e\x89ð\x9f\x92¥ð\x9f\x92¥', '1027', '102816', '102pm', '1030', '10353', '104', '1044',
'10450', '10480', '10550', '1059am', '105kg', '106', '10650', '10670', '1070', '10700', '1080', '10830', '1096', '10a',
'10alltypespos', '10am', '10days', '10hrs', '10k', '10kday', '10kms', '10m', '10meses', '10miler', '10millionmiler', '10
```

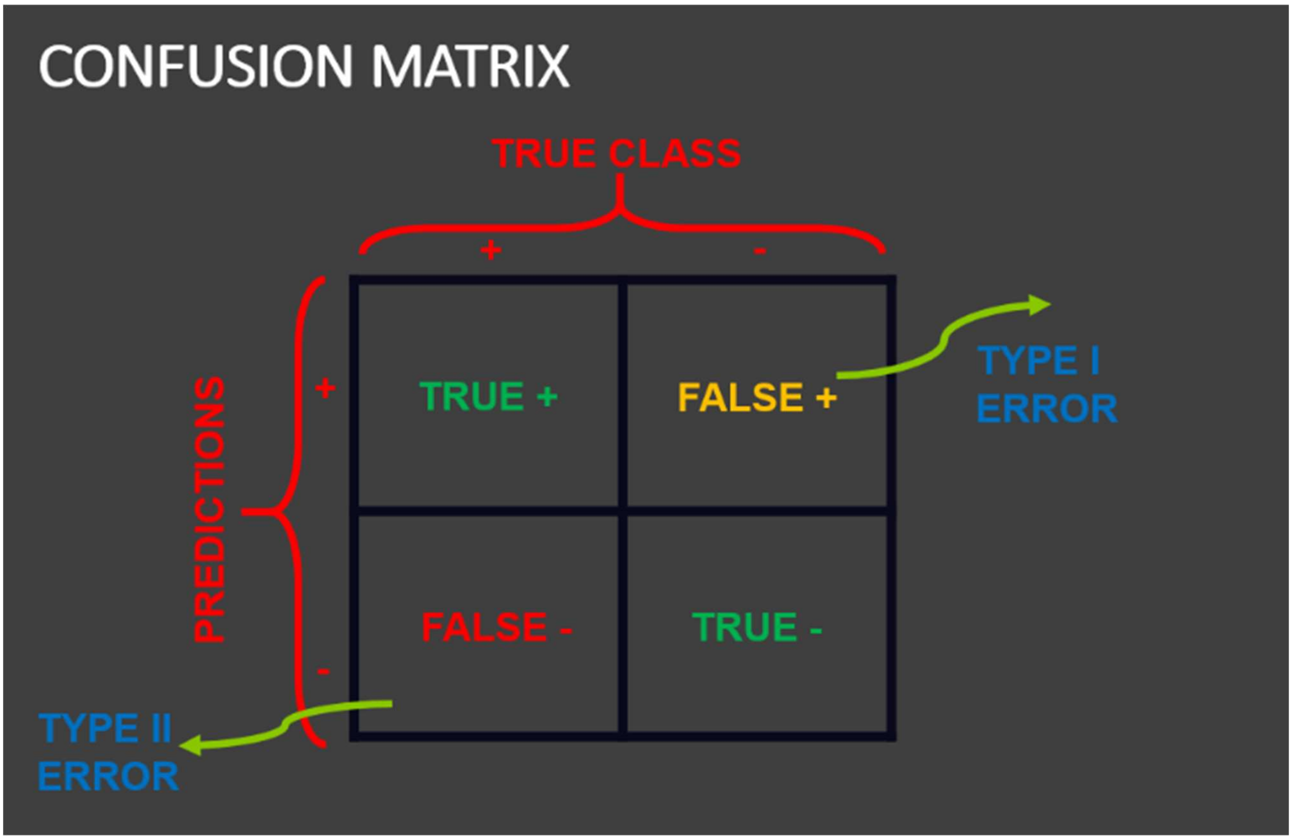
Train And Evaluate A Naive Bayes Classifier Model:

```
In [57]: X.shape
Out[57]: (31962, 47386)

In [58]: y.shape
Out[58]: (31962,)

In [59]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)

In [60]: from sklearn.naive_bayes import MultinomialNB
         NB_classifier = MultinomialNB()
         NB_classifier.fit(X_train, y_train)
```



Conclusion:

So, this is how we build, train and visualize the “Twitter Sentiment Analysis Using Naive Bayes Classifier” in Python.