**Project Title:**

Identify the Sentiments

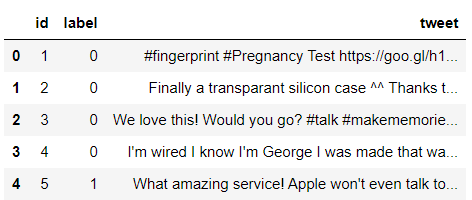
**Problem Statement:**

The problem in identify the sentiments is classifying the polarity of tweet dataset. whether the expressed opinion in a document, a sentence or an entity is positive or negative.

**Progress:**

**Load the Dataset**

Load the dataset which includes id, tweet, and the corresponding label



Initial dataset shape: (7920, 3)

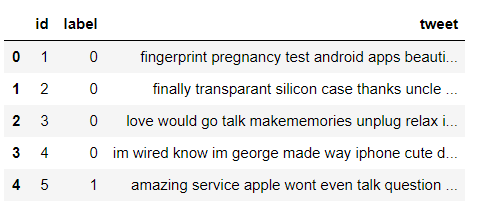
**Data Cleaning**

Dataset haven’t any null values and duplicate values.

Here we are working with text. So, we can follow different text data preprocessing techniques such as,

* Convert uppercase to lowercase
* Remove links
* Remove punctuation marks
* Remove numbers
* Remove stopwords
* Stemming and Lemmatization

After those preprocessing steps, dataset looks like,

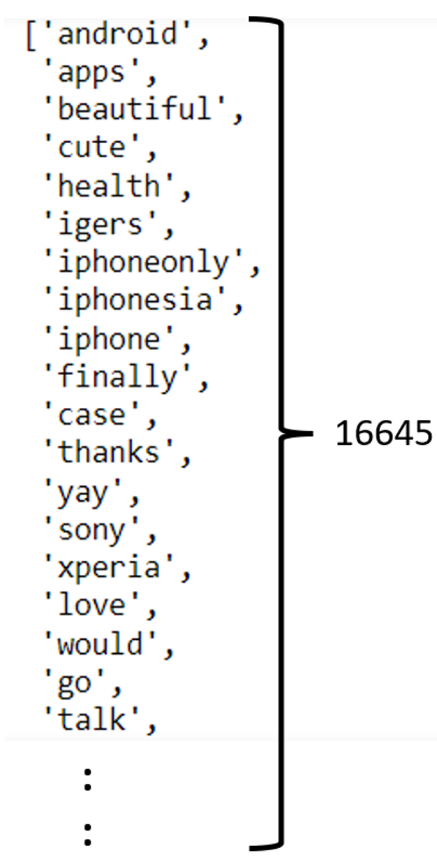


Note: We can see, all tweets are in lowercase. Also, there are no any punctuation marks, numbers in the resulted tweets.

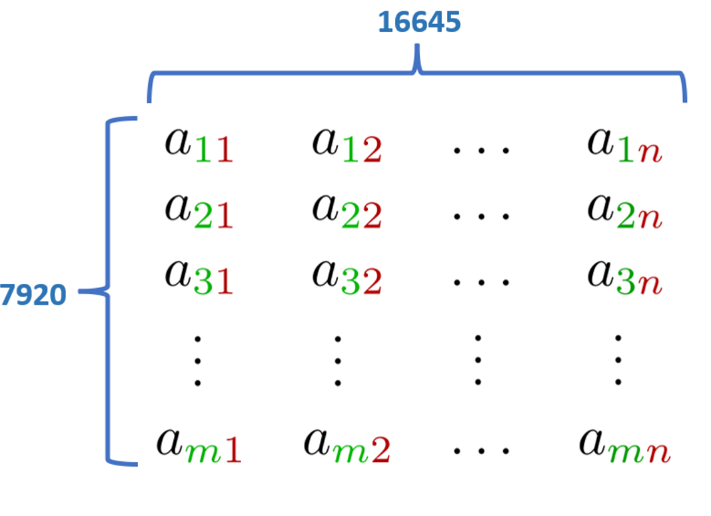
**Building a Vocabulary**

Before further next step, we need to build a vocabulary using all tweets that we have. To do that, we used **Counter** class which is in python collections. It gives us unique words along with frequencies.

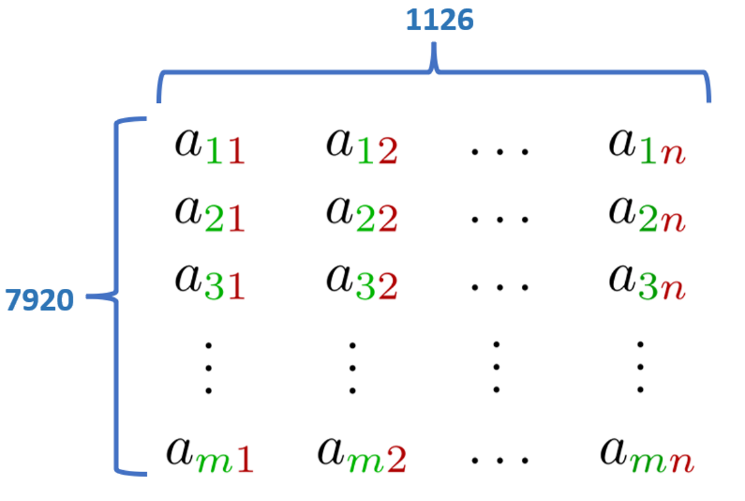
Initially our vocabulary has 16645 terms.



But we have only 7920 instances for training. Then most probably it has overfitting problem when we are training dataset like below.



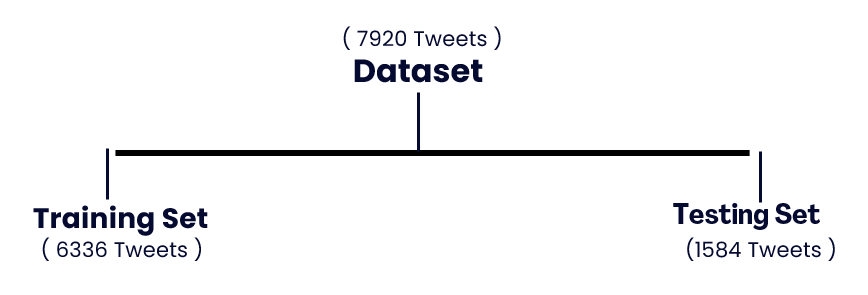
So, to reduce our vocabulary size, we consider terms that are more than 10 occurrences. That’s how we reduce the vocabulary size to 1126.



**Divide the dataset**

We need to divide our whole dataset into training and testing. For that, we used train\_test\_split method in sklearn library. Here we divide dataset with 5:1 ratio.

So, finally we got 6336 tweets for training purpose and 1584 tweets for testing purpose.



**Vectorized our Dataset**

Now we need to vectorized whole dataset using the vocabulary which we build earlier.

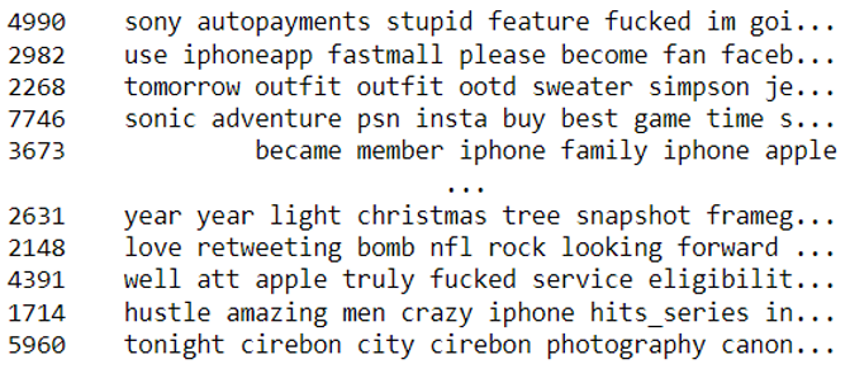
For example, it happened for each tweet like below.

Vocabulary: [ I, love, Python, programming, language, is, a, popular ]

Sentence: Python is a general-purpose programming language

Vectorized form: [ 0, 0 , 1 , 1 , 1 , 1 , 1 , 0 ]

Before vectorization



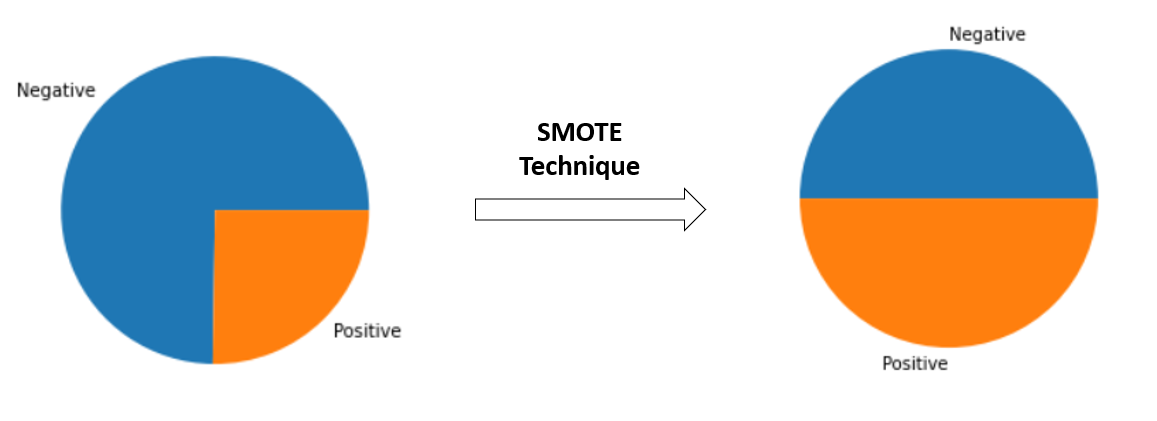
After vectorization

A picture containing scatter chart

Description automatically generated

**Handle imbalanced Dataset**

Here we are working with an imbalanced dataset. Because when we considering value counts in labels, we found there are 4715 negative tweets and 1621 positive tweets. Then we try to oversample the dataset using SMOTE technique.



Then we able to get same number of negative and positive data in the training set.

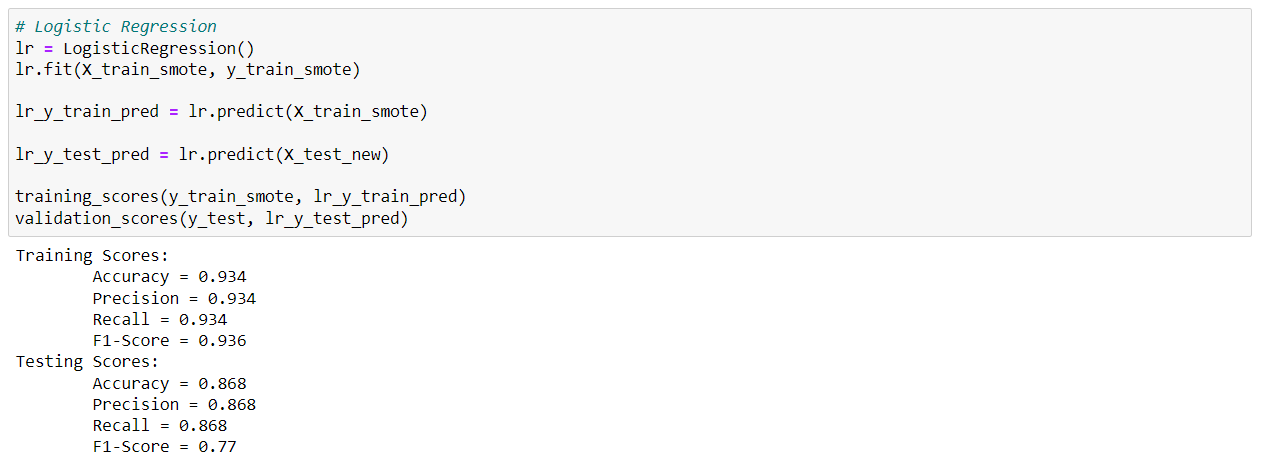
**Training and evaluation the models**

Here I used few different machine learning algorithms to implement sentiment analysis model.

* Logistic Regression
* Naïve Bayes Classifier
* Random Forest Classifier
* Gradient Boosting Classifier
* Support Vector Classifier

Also, we used different performance measures such as accuracy, precision, recall and f1-score for evaluate our models.

1. Logistic Regression



1. Naïve Bayes Classifier

Graphical user interface, text

Description automatically generated

1. Random Forest Classifier

Graphical user interface, text, application

Description automatically generated

1. Gradient Boosting Classifier

Graphical user interface, text

Description automatically generated

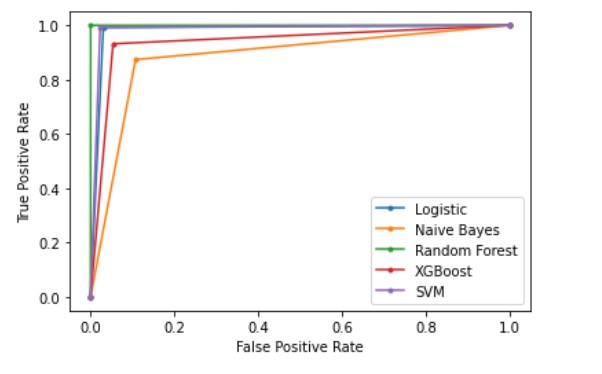
1. Support Vector Classifier

Text

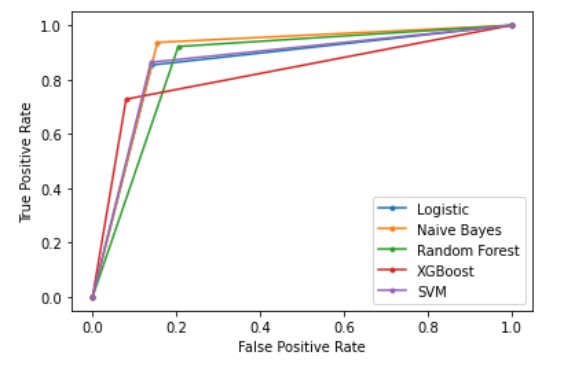
Description automatically generated

**Evaluate model using ROC Curve**

Training phase



Testing Phase



**Save the model**

* After checking the model performance, we save our model as a pickle file to further use.

**import pickle**

**with open('model\_pkl', 'wb') as files:**

**pickle.dump(rf, files)**

* Also, we save our vocabulary where we build using training dataset.

**def save\_list(lines, filename):**

**data = '\n'.join(lines)**

**file = open(filename, 'w', encoding="utf-8")**

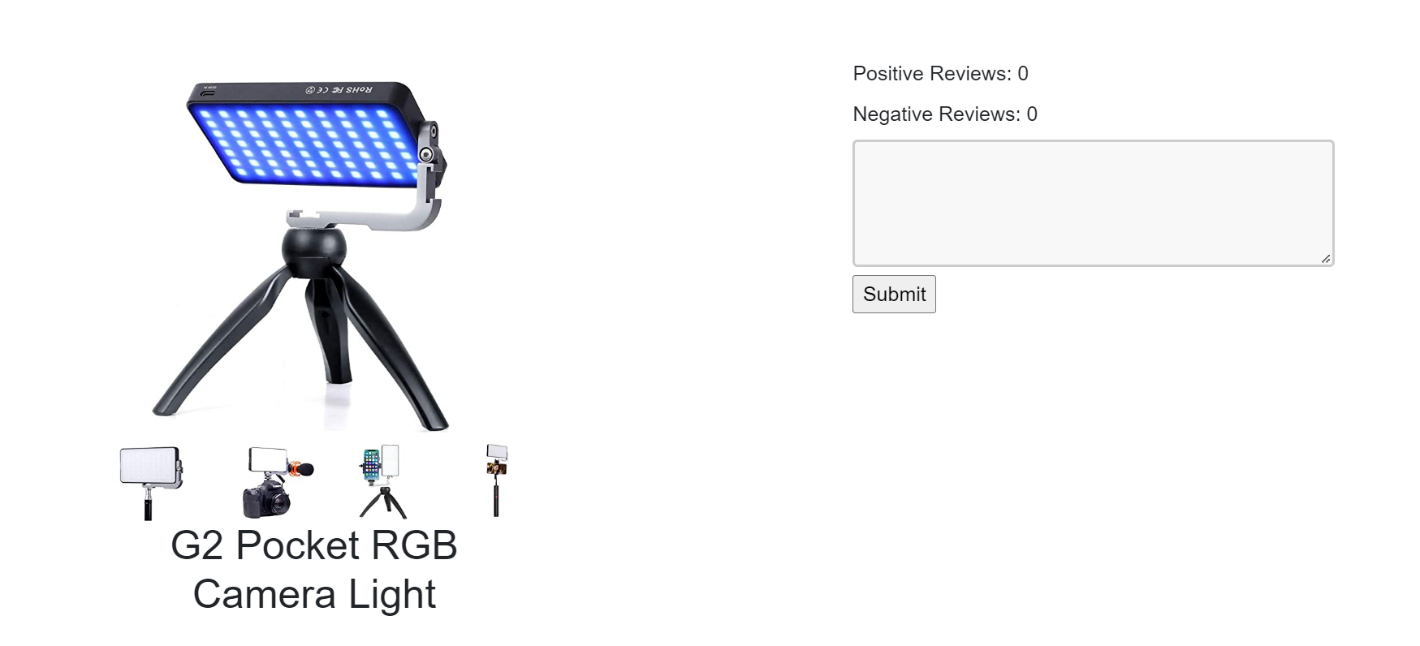
**file.write(data)**

**file.close()**

**save\_list(tokens, 'vocabulary.txt')**

**Build a simple web application**

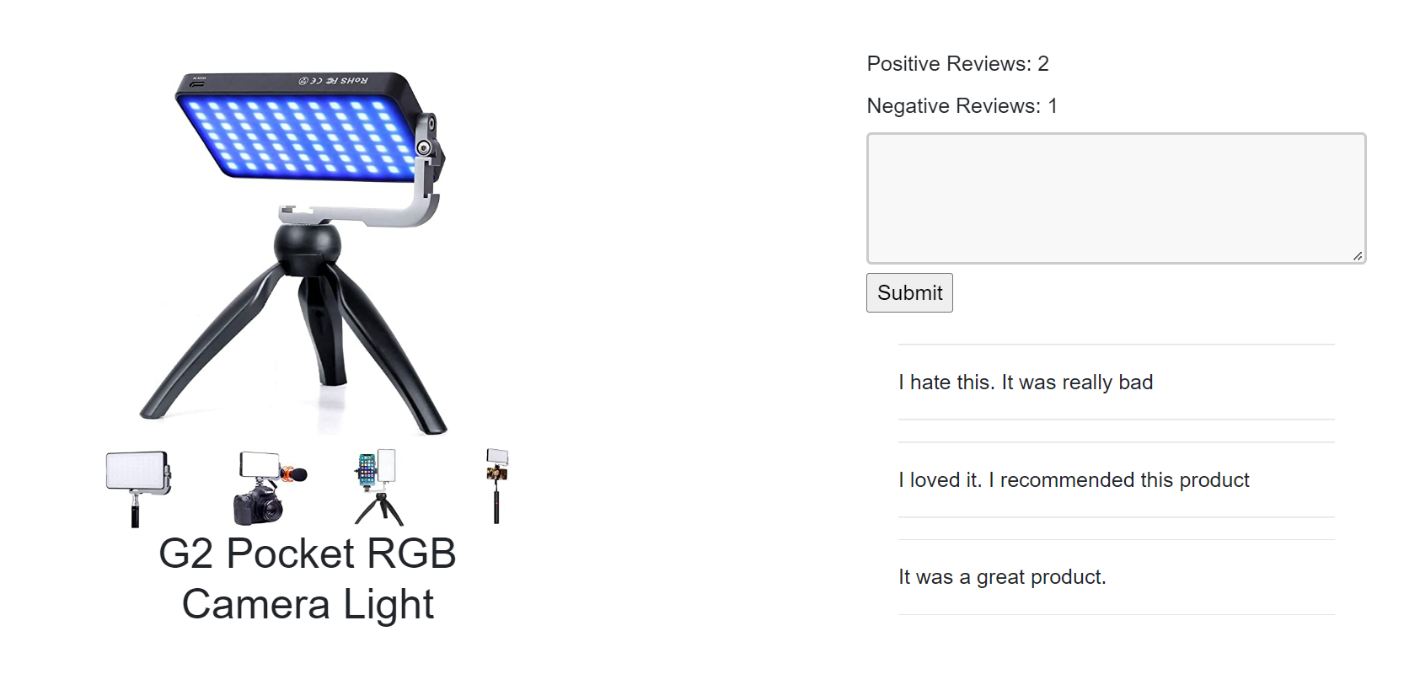
I created a web application using Flask, a Python web framework, that allows users to input text and receive a sentiment analysis of that text. The application uses natural language processing techniques to classify the sentiment of the input as positive or negative. The application is designed to be user-friendly and easy to use, with a clean and intuitive interface.



In the Flask server, I built a simple procedure to identify sentiments with,

* Load the model
* Get user input
* Text preprocessing
* Vectorization
* Get prediction

Then it will generate an output. So, we can demonstrate it like below,



Here we can see the number of positive and negative reviews according to our comments.

**Application link:** https://github.com/dineshpiyasamara/sentiment\_analysis\_application