FEATURE EXTRACTION BASED SENTIMENT ANALYSIS

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

VIPUL SHARMA 191266

Under the supervision of

DR. HARI SINGH

to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

.

Candidate's Declaration

I hereby declare that the work presented in this report entitled **" FEATURE EXTRACTION BASED SENTIMENT ANALYSIS "** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of **DR. HARI SINGH** Assistant Professor (SG).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
VIPUL SHARMA, 191266

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
DR. HARI SINGH
ASSISTANT PROFESSOR (SG)
Computer Science and Engineering/Information Technology
Dated: 25-04-23

ACKNOWLEDGEMENT

First of all, I would like to express my heartiest gratitude and gratefulness to almighty god for his divine blessings helped me to complete the project work successfully in the given period of time.

I extend my heartiest thanks to my project Supervisor DR. HARI SINGH, Assistant Professor (SG), Department of Computer Science & Engineering, Jaypee University of Information Technology, Waknaghat. Vast knowledge and keen interest of our supervisor in the field of Machine Learning helped me a lot to execute this project. His endless patience, scholarly guidance, continual encouragement, constant, energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all the stages have made it possible to complete this project.

I would also generously thank each one of those individuals who have helped me directly or indirectly in successfully carrying out the execution of this project. In this situation, I also want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Last but not the least, I must acknowledge with due respect the constant support and faith of my parents.

VIPUL SHARMA, 191266

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

- **NLP : Natural Language Processing**
- **TF-IDF : Term Frequency - Inverse document frequency**
- **BoW : Bag oF Words**
- **W2V : Word2vec**
- **AI : Artificial Intelligence**
- **ML : Machine learning**
- **IMDB : Internet movie database**
- **PCA : Principal Component Analysis**
- **LDA : Linear Discriminant Analysis**
- **BiGRU : Bidirectional Gated Repetitive Unit**

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The world is fast growing place, and so do the thoughts of people living in it. In this very fast growing world, there are a lot of ways available for people to express their thoughts and views on anything and everything. They share their thoughts in conversations, messages, letters, diaries, and on social media platforms like facebook, instagram, twitter, koo etc. There are a billion number of people in the world who use social media platforms to use as a medium to express their thoughts be it any kind of thought, for example; happiness, sadness, anger, hate, trolling some celebrity or a political leader, sharing some news, sharing something insightful, giving updates about their life, criticising, reviewing a product or anything else. Here we are going to talk about the tweets by people on twitter.

Twitter is a social media platform for microblogging and social networking, where people can post, share, comment, like and retweet anything and everything. A large number of people share their opinions on twitter. So now the question is how to identify which opinion has a positive impact and which one has a negative impact. This is where sentiment analysis comes into play.

A lot of research has been done in this field regarding the polarity of these opinions which are posted by people. These researches have been carried out on various types of data like reviews of movies, web series; example IMDB reviews, e-commerce websites; like amazon, myntra, etc., social media platforms like twitter, facebook etc., songs reviews, hotel reviews, airline reviews and the list goes on. In this project we have tried our hands on the Twitter reviews, using feature extraction based sentiment analysis. Natural language processing is used to apply the models on the dataset to check the performance of the models how well they can distinguish between the positive and negative tweets. The results were compared and the best model was obtained as a conclusion. Techniques that were compared were Bag of words feature, TF-IDF, Word2vec features.

CHAPTER - 1

INTRODUCTION

1.1 INTRODUCTION

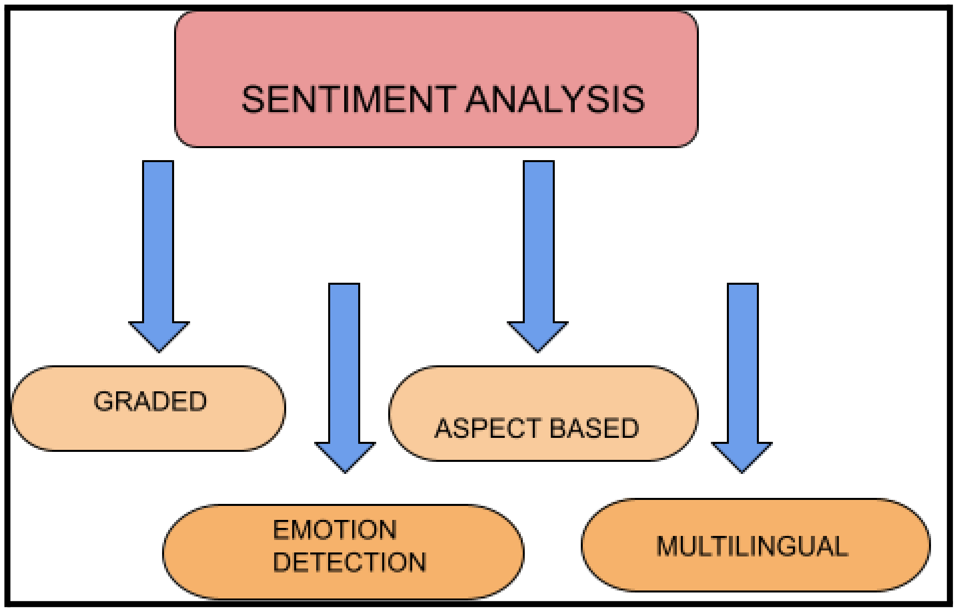
With the advancement of web technologies, population all over the world is able to express their thoughts and opinions on anything and everything from anywhere over the globe. Twitter has been most commonly used platform by people to post about their opinions on various topics. Twitter has a wide variety of genres on which people can express their thoughts and opinions, like technologies, science, movies, politics, their personal life, crisis in the world, wars and much more. Obviously it is a very difficult task for humans to read each and every tweet, analyse and tell its polarity. Therefore, we humans have assigned this task to machines. We can use natural language processing, to find out the sentiment of the tweet. NLP techniques have given way to various possibilities. Using sentiment analysis one can discover the patterns in the dataset, find the polarity of the textual dataset.

SENTIMENT ANALYSIS

Sentiment analysis is also referred to as opinion mining or emotion AI. It is basically the use of natural language processing techniques to discover, or extract the useful information from the dataset that we are given. It is used to understand the emotional tone of the part of a text or the entire sentence, for example any review on an e commerce website, how are we gooing to distinguish between the positive and the negative tone, how is a user supposed to make a decision to buy or not to buy the product. It is done on the basis of sentiment analysis. Another example of sentiment analysis is how do social media platforms identify whether a particular user is posting something positive or negative? Social media platforms use sentiment analysis to identify the polarity of the posts, comments, tweets etc to take actions if in any sense a particular user has violated their norms.

Consider the sentences; I like the way Hritik Roshan performs in his films. This is a positive sentiment because it has key word ‘like’ which states a positive sentiment. Similarly if it was I do not like the way Hritik Roshan performs in his films. This sentence would have been a negative sentiment, because the word used is ‘not’ and its suggests a negative meaning.

TYPES OF SENTIMENT ANALYSIS



(Fig- 1)

Types of Sentiment analysis

From the figure shown above it is clear that sentiment analysis is of four different types. They can be explained as follows:

- ***Graded Sentiment Analysis :*** This sentiment analysis model assists you with inferring extremity accuracy. You can direct a feeling examination across the accompanying extremity classifications: exceptionally good, good, nonpartisan, pessimistic, or extremely pessimistic. Graded Sentiment analysis is useful for the investigation of surveys and evaluations. For a rating scale from 1 to 5, you can think about 1 as exceptionally negative and five as extremely sure. For a scale from 1 to 10, you can think about 1-2 as exceptionally negative and 9-10 as extremely positive.

- ***Emotion Detection Sentiment Analysis :*** As the name proposes, emotion detection sentiment analysis assists you with recognizing feelings. This can incorporate displeasure, trouble, satisfaction, disappointment, dread, stress, alarm, and so forth. Feeling location frameworks ordinarily use dictionaries - an assortment of words that convey specific feelings. A few high level classifiers likewise use strong AI (ML) calculations. It's prescribed to involve ML over dictionaries since individuals express feelings in a heap of ways. Take this line, for instance: "This item is going to kill me." This line might communicate sensations of dread and frenzy. A comparative line - this item is killing it for me - has a completely unique and positive significance. In any case, "kill" may be related with dread or frenzy in the vocabulary. This might prompt wrong feeling identification.
- ***Aspect Based Sentiment Analysis :*** While graded sentiment analysis assists you with deciding the general extremity of your client audits, aspect based examination dives further. It assists you with deciding the specific perspectives individuals are discussing. Suppose; you're a cell phone producer, and you get a client survey expressing, "the camera does not work well in counterfeit lighting conditions." With viewpoint based examination, you can confirm that the commentator has remarked on something "negative" about the "camera."
- ***Multilingual Sentiment Analysis :*** Multilingual sentiment analysis can be troublesome. It includes a ton of preprocessing and assets. The majority of these assets are accessible on the web (for example feeling vocabularies), while others should be made , yet you'll have to know how to code to utilize them. On the other hand, you could distinguish language in messages consequently with a

language classifier, then, at that point, train a custom emotion mining model to order messages in your preferred language.

ADVANTAGES OF SENTIMENT ANALYSIS

- Sorting the data
- Real time analysis of the opinions
- Gives consistency criteria to users

FEATURE EXTRACTION

Feature extraction is a course of dimensionality decrease by which an underlying arrangement of important information is diminished to additional reasonable gatherings for handling. A trait of these enormous informational indexes is countless factors that require a ton of registering assets to process. Include extraction is the name for techniques that select and/or consolidate factors into highlights, actually diminishing how much information that should be handled, while still precisely and totally portraying the first informational index.

The course of feature extraction is helpful when you really want to lessen the quantity of assets required for handling without losing significant or important data. Highlight extraction can likewise lessen how much excess information for a given examination. Additionally, the decrease of the information and the machine's endeavors in building variable mixes (highlights) work with the speed of learning and speculation steps in the AI or machine learning cycle.

Feature extraction is important as having a large number of features or inputs in the data might increase the possibility of an overfitting model, this overfitting can in future lead to poor performance of the machine learning model because it will not be able to produce desired results. This can lead to lower accuracy on the testing dataset, and incorrect outputs on the processed data. So all of the preprocessing resources and time would be wasted.

APPLICATIONS OF FEATURE EXTRACTION IN REAL WORLD

- It is used in artificial intelligence
- It is used in Artificial neural networks
- It is used in Radiomics
- It is used in extraction methods
- It is used for image processing
- It is used for signal and time series data
- It is used for audio applications
- It is used for natural language processing

TYPES OF FEATURE EXTRACTION

Feature extraction is used in natural language processing to analyse and find the similarities and differences between the texts. Requirement of feature extraction procedures Artificail Intelligence calculations gain from a pre-characterized set of highlights from the preparation information to create yield for the test information. Yet, the primary issue in working with language handling is that machine learning calculations can't chip away at the crude text straightforwardly. Thus, we really want some component extraction strategies to change over text into a matrix(or vector) of elements. Probably the most well known techniques for feature extraction are :

- Bag of-Words
- TF-IDF

We will be discussing NLP , Bag-of-Words, and TF-IDF further in the report.
Other two most commonly used feature extraction techniques are :

- ***PCA*** : PCA stands for Principal Component Analysis. In straightforward words, PCA is a technique for getting significant factors (in type of parts) from a huge arrangement of factors accessible in an informational index. It will in general track down the bearing of most extreme variety (spread) in information. PCA is more helpful while managing 3 or higher-layered information.
- ***LDA*** : LDA stands for Linear Discriminant Analysis. The possibility of LDA is very basic. Given the preparation test set, LDA attempts to extend the example onto a straight line with the goal that the projection points of interclass tests are pretty much as close as could be expected and the projection points of the intraclass tests are as far separated as could really be expected. While characterizing another example, we extended it onto a similar line. The grouping of this example is resolved in view of the place of the projected point. Not at all like PCA, which attempts to keep up with information data however much as could be expected, LDA is to make the information focuses more recognizable after aspect decrease.

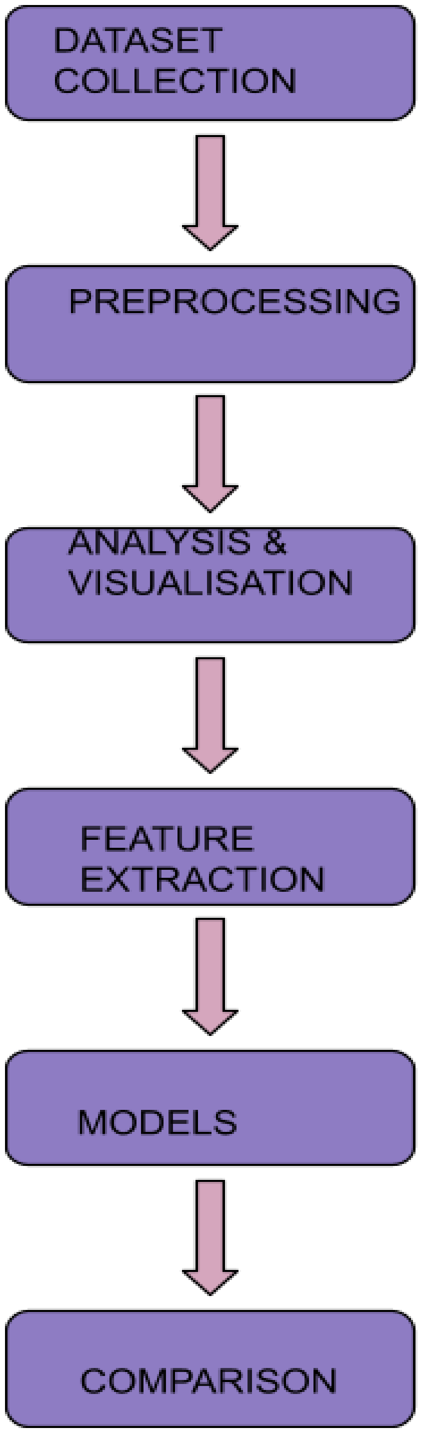
1.2 PROBLEM STATEMENT

Reviews and tweets present on social media platforms like twitter are are very large in number and it is difficult for the organization to find out what type of polarity they have. Sentiment Analysis can come into play at this stage in order to identify the tone of the reviews or comments posted about a particular topic. Feature Extraction enhances the performance of sentiment analysis in this case. It can help the websites to learn which type of comments are posted on their platform and how can they improve, or help their users, or take certain actions if some norms are violated. Model comparison is to be performed to find out which model performs better.

1.3 OBJECTIVES

- Understanding feature extraction based sentiment analysis
- Examining how to pre process data when NLP is being used.
- Implementing NLP techniques like bag of words, TF-IDF etc
- Comparing the performance of different methods.
- Using various machine learning libraries and understanding their uses
- Differentiating between polaritites of the text present in the dataset

1.4 METHODOLOGY



(Fig- 2)

Methodology of the project

On explaining step by step the procedures written in above figure all we can say is that the first step involved is the dataset selection, extraction from any desired source. The dataset used by us in this project is the twitter dataset, which contains tweets in the form of comments on the posts, retweets of some famous tweets etc, by different users of the twitter. It comprises of two csv files, namely test and train. As the name suggests test and train, the train file is used for training the model, and test file is the one on which outputs are tested.

Then the next step is preprocessing, it is a combination many small yet very very significant steps involved in training in any model. Preprocessing the part of machine learning where in all the changes in the model takes, how we process our data prior to executing it, how we prepare it is all that matters in the ML. If the preprocessing is perfect then our model will be perfect, there will be less errors, there will be less noise in the data, etc. The preprocessing steps that we have performed are as follows:

- Cleaning is an important part in the project as , if our data is not clean which means that our data has some malicious datapoints , some null points, some irrelevant values, then those values are either required to be removed to to be replaced in the dataset with some suitable values, like mean values, median, etc. In this project the data set is text base and cleaning is done like this; first of all both the files train and test are combined together so that the pre processing is easy to perform.
- Then all the redundant, irrelevant , numeric values, insignificant words, punctuation marks which are not useful are removed.
- Then all the tweets are tokenised. Tokenisation process is discussed further in the report. Basically it is nothing but simply spliiting the large sentences, paragraphs into smaller ones, by removing insignificant words, this id one so that the sentences can be easily interpreted.
- Next part is stemming, stemming is converting the words to their main or root word in order to understand their meaning from the dictionary very easily. Stemming will be discussed in detail in the upcoming part.

The next part is data analysis and visulaization. In this dataset is visualized using python libraries, like matplotlib, seaborn, pandas, numoy, etc. The visualizatiin part has the representation of the polarity in the form of a pie chart which is given in the dataset. The dataset has 2 polarities 0 and 1. They are visualized with the help of a pie chart as shown below in the figure.

Next is the checking the distribution of of the lengths of the tweets in both the files using a histogram. The histogram of the length comparison shown below in the figure.

The analysis of the dataset includes generation of the word clouds of the words in the dataset, first of all word cloud of all the words, then a separate one for positive words and the another one of the negative words, id generated using wordcloud library. Wordcliud is used to represent the frequency of the words in the dataset, larger the size of the word more is its frequency in the dataset.
The three wordclouds discussed above are shown in the figures below.
The next analysis is the analysis of the hashtags, because hashtags used in tweets are very helpful in understanding what all is the tweet actually about, so only these punctualtions marks, symbols are important for us to keep them in the dataset and rest of them can be removed. The usage of these hashtags are displayed with the help of bar plots. Two different barplots for positive and negative hashtags are used.

1.5 ORGANIZATION

The report is organised in five cahpters all the chapters constitute importatnt information regarding the project. First cahpter consists of the basic introduction of the project what is sentiment analysis. It is used to understand the emotional tone of the part of a text or the entire sentence, for example any review on an e commerce website, how are we gooing to distinguish between the positive and the negative tone, how is a user supposed to make a decision to buy or not to buy the product. Second chapter has the literature survey what all research papers we have used, we studied in the making of this project.Third chapter has the information about System development and the usage of the different softwares, dataset, feasibility study of the project, limitations, etc.

The fourth chapter has the performance analysis and the three methods used are compared on all aspects with each other. Chapter five has the conclusions of the the models applied and results are compared and future work in the project .

CHAPTER - 2

LITERATURE SURVEY

Author(s)	Journal/Conference, year	Published By (IEEE, Elsevier, Springer)	Methodology	Disadvantage
A. khan, S. Ahmad	2014	Researchgate	Searching, Criteria of Inclusion, exclusion and presentation	Feature space problem, redundancy
R. Ahuja, A. Chugh, S.Kohli, P.Ahuja	2019	Elseiver	Six preprocessing techniques and seven classifiers	PCA not used
H. Siqueria, F. Barros	2010	ICMC	Identification of parts of speech	Used less models

Table 1 for literature survey

The research papers used for the literature survey were some of the best works carried out in the field of sentiment analysis using feature extraction. In the past few years a lot of work has been done like analysis on the e commerce websites, covid-19 the latest trending thing of twitter, Online education review of classes, twitter analysis etc. All the work done has used similar techniques, TF-IDF, word2vector, bag of words, pca etc. to get their sentiment analysis done.

Author(s)	**Journal/Conference, year**	**Published By (IEEE, Elsevier, Springer)**	**Methodology**	**Disadvantage**
M.Avinash, E. Sivasankar	2019	Springer	TF-IDF on movie dataset	Feature space problem, redundancy
Z. Nanli, Z. Ping, L. I. Weiguo	2012	IEEE	Polarity	Used less models
N. Devasia, R. Sheikh	2016	IEEE	Recursive Deep Model	High dimensionality
Vedhavati N., Anil Kumar K.M	2022	Elseiver	TF-IDF W2V	PCA not used
Caglar G, Ismael A, Cem G	2022	Research gate	TF-IDF	High dimensionality

Table 2 for literature survey

The Twitter information can be utilized in many fields like wellbeing, training, economy and legislative issues. Be that as it may, breaking down a lot of text information is definitely not a minor errand and requires progressed AI strategies. Despite the fact that there are a few proposed techniques to beat this issue, it is viewed as that a complete investigation that incorporates the greater part of the most exceptional AI strategies is profoundly esteemed.[7]

In opinion examination task, the most noteworthy prescient exhibition has been gotten by Bidirectional Gated Repetitive Unit (BiGRU) engineering plan related to pretrained GloVe word implanting technique, with an order exactness of 95.35%. In subject characterization task, the most elevated prescient presentation has been gotten by Stacking gathering strategy utilizing together unigram and bigram word based Term

Recurrence Opposite Report Recurrence (TF-IDF) highlight extraction procedure, with a grouping exactness of 91.49%.[7]

Web based learning is likewise alluded to as E-realizing which has acquired gigantic consideration and drawn in the vast majority during the Coronavirus lockdowns. Because of the overabundance of online data, clients face extreme difficulties and hardships understanding the best course that is being cutthroat in the worldwide market. In this manner, it is important to foster a web-based suggestion framework that upholds the clients in choosing the best course with E-learning.The top course proposal is managed relying upon the closeness scores like Jaccard similitude, cosine comparability and euclidean similitude. Likewise, the misfortune capability in the classifier is decreased by advancing the weight boundaries utilizing the EAO approach. The presentation examination shows that the proposed suggestion model acquires further developed brings about terms of exactness of 99.98%, review of 99.81%, accuracy of 99.65%, and F-proportion of 99.95%. The relative examination display that the proposed EMRMR_EAO model accomplishes preferable execution over the other existing works in the writing.[6]

Fast expansion in web clients alongside developing force of online audit locales and virtual entertainment has brought forth Feeling examination or Assessment mining, which targets figuring out what others think and remark. Feelings or Assessments contain public created content about items, administrations, approaches and legislative issues. Individuals are normally intrigued to look for positive and negative conclusions containing different preferences, shared by clients for elements of specific item or administration. Thusly item highlights or angles have critical job in feeling investigation. Not withstanding adequate work being acted in message examination, highlight extraction in opinion examination is
presently turning into a functioning area of examination. This survey paper examines existing procedures and approaches for highlight extraction in feeling examination and

assessment mining. In this survey we have taken on a deliberate writing survey interaction to distinguish regions all around centered by specialists, least tended to regions are likewise featured giving an opportunity to analysts for additional work. We have additionally attempted to distinguish most and least normally utilized highlight choice methods to find research holes for future work.[1]

There are so many component extraction methods, for example, Sack of Words, TF-IDF, word installing, NLP(Natural Language Processing) based highlights like word count, thing count and so on. In this paper we examined the effect of two highlights TF-IDF word level and, N-Gram on SS-Tweet dataset of feeling examination. We found that by utilizing TF-IDF word level (Term Recurrence Backwards Record Recurrence) execution of opinion investigation is 3-4% higher than utilizing N-gram highlights, examination is finished utilizing six order algorithms(Decision Tree, Backing vector Machine, K-Closest Neighbor, Arbitrary Timberland, Calculated Relapse, Gullible Bayes) and taking into account F-Score, Exactness, Accuracy, and Review execution boundaries.[2]

While dissecting opinions from the emotional text utilizing AI strategies, highlight extraction turns into a huge part. We play out a concentrate on the presentation of element extraction strategies, TF-IDF utilizing Cornell film survey datasets, UCI feeling marked datasets, stanford film audit datasets, successfully ordering the message into good and pessimistic polarities by utilizing different preprocessing techniques like disposing of stop words and tokenization which expands the exhibition of opinion examination as far as exactness and time taken by the classifier. The highlights acquired in the wake of applying highlight extraction methods on the message sentences are prepared and tried utilizing the classifiers calculated relapse, support vector machines, K-closest neighbors, and Bernoulli Naive Bayes.[4]

An enormous number of suppositions are clients' surveys about items and administrations, as internet business turned out to be more well known. This pattern persuaded a few examination works and market applications focusing on the programmed investigation of the accessible feelings. Obviously, this data is urgent for administrators, who ought to work on the nature of the offered administrations in light of clients' perspectives. Concerning administrations given by ventures, for example, stores or lodgings, it is especially challenging to distinguish the elements being remarked on (e.g., nature of administration, conveyance cost, and so on). This work presents a space free cycle for highlight extraction which was utilized in the development of WhatMatter.[3]

CHAPTER - 3

SYSTEM DESIGN & DEVELOPMENT

3.1 ANALYSIS

Feasibility Study

• The main problem while working on a large dataset is time consuming.
• Moving to the twitter dataset the problem is to identify the nature of the text & comments on the twitter platform.
• Moving to the dataset the problem in the dataset is a different accuracy measure.

Problem Analysis

1. While working on the large dataset it consumes a lot of time in Uploading the dataset and running different algorithms it slows down the processing speed while working on a
single system.
2. We tried to find the best possible way to identify the nature of the tweets (positive negative , neutral) and identify the text belonging to which class.
3. We implemented the different feature extraction method and measured the accuracy , and
compared the accuracy and identified which algorithm is best for the model .

3.2 REQUIREMENTS

System requirement is the ability of the system to meet the condition desired by the users.
System requirement analysis is performed by grouping the needs into functional requirements and non functional requirements.

Functional Requirements

The functional requirements of the Sentiment Analysis System are as follows:

• The system can manage (add, read, update, and delete) twitter dataset , dictionary of root
words , dictionary of stop words .

• The system can perform the training process and display the model in the form of feature sets of the term data from the training data.

• The system can display a set of twitter dataset terms derived from tokenizing, filtering, and stemming processes.

Non-Functional Requirements

Meanwhile, the non-functional requirements of the Sentiment Analysis System are as follows:

The system can run in various web browsers which support the system environment.

The system gives a fast response.

The system has a user-friendly design.

PLATFORM

The coding platform used in the project is google colaboratory.

(Fig - 3)

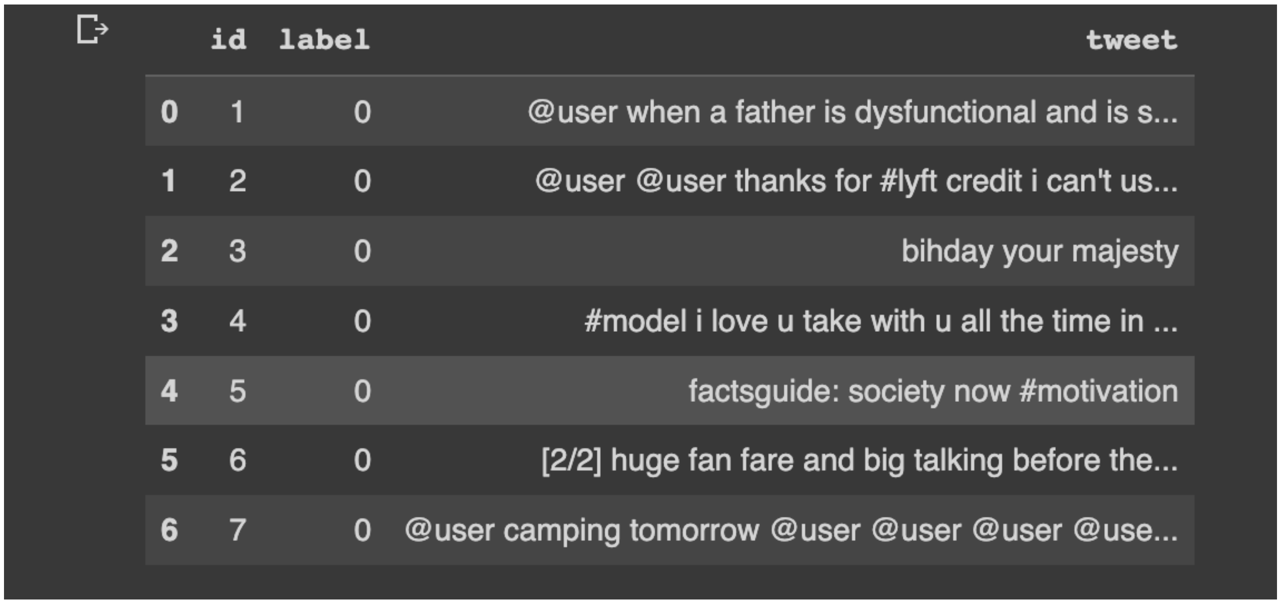
Colab is a completely cloud-based Jupyter notebook environment that is free to use. The notebooks you create can be simultaneously modified by your team members, exactly like you edit documents in Google Docs, and most significantly, it doesn't require any setup. Many well-known machine learning libraries are supported by Colab and are simple to load in your notebook.

Some of the things that colab offers to its users are :

- Create and run Python code
- Keep track of the code you use to support mathematical equations.
- Make, upload, and share notebooks
- Notebooks can be imported or saved to Google Drive.
- Publish and import notes from GitHub
- Add outside datasets, such those from Kaggle
- include Keras, TensorFlow, PyTorch, and OpenCV
- Free Cloud and GPU service

3.3 IMPLEMENTATION

The implementation is done in different phases. The collection of the data is the first step. the dataset is twitter dataset which is text based and it contains two files, one is train.csv and the other one is test.csv. Both of these files are used in the implementation of the sentiment analysis.



	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
5	6	0	[2/2] huge fan fare and big talking before the...
6	7	0	@user camping tomorrow @user @user @user @use...

(Fig - 4)

This is the snapshot of the datatset to getva vague idea about the features of the dataset. Then the next step is preprocessing, it is a combination many small yet very very significant steps involved in training in any model. Preprocessing the part of machine learning where in all the changes in the model takes, how we process our data prior to executing it, how we prepare it is all that matters in the ML. If the preprocessing is perfect then our model will be perfect, there will be less errors, there will be less noise in the data, etc. The preprocessing steps that we have performed are as follows:

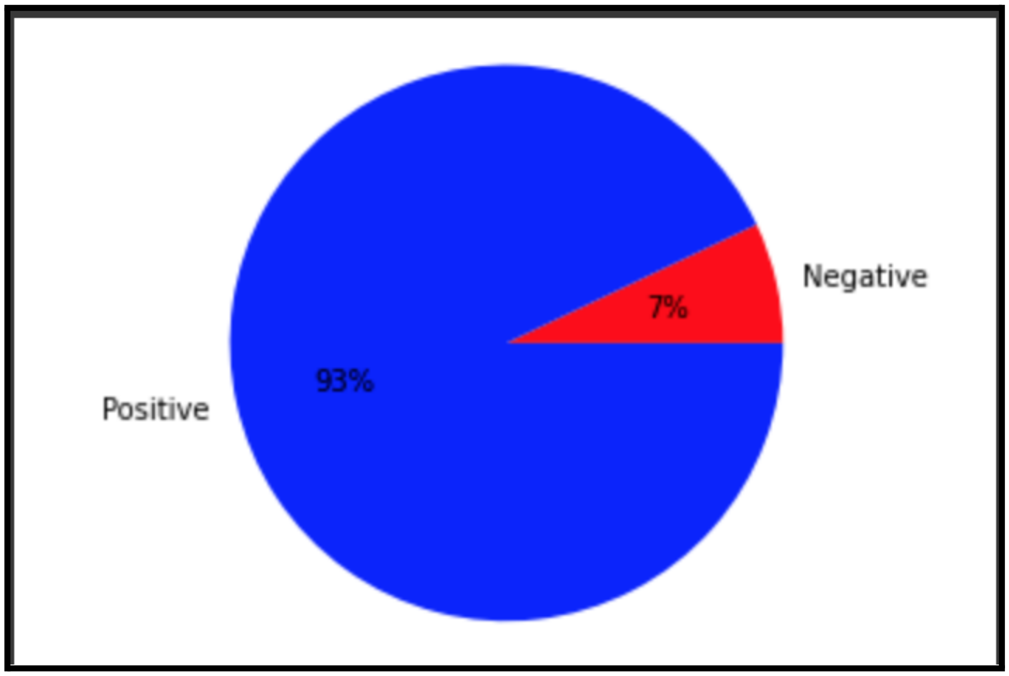
- Cleaning is an important part in the project as , if our data is not clean which means that our data has some malicious datapoints , some null points, some irrelevant values, then those values are either required to be removed to to be replaced in the dataset with some suitable values, like mean values, median, etc. In this project the data set is text base and cleaning is done like this; first of all both the files train and test are combined together so that the pre processing is easy to perform.
- Then all the redundant, irrelevant , numeric values, insignificant words, punctuation marks which are not useful are removed.
- Then all the tweets are tokenised. Tokenisation process is discussed further in

the report. Basically it is nothing but simply spliiting the large sentences, paragraphs into smaller ones, by removing insignificant words, this id one so that the sentences can be easily interpreted.

- Next part is stemming, stemming is converting the words to their main or root word in order to understand their meaning from the dictionary very easily. Stemming will be discussed in detail in the upcoming part.

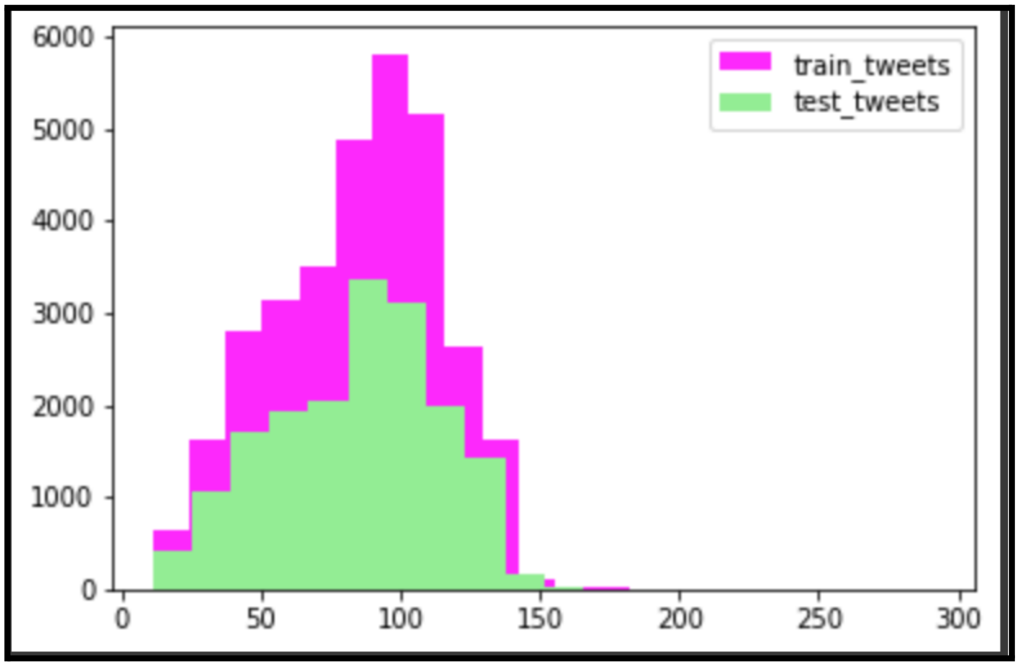
We normalize the text using the porter stemmer it helps in reducing the vocabulary size.

The next part is data analysis and visulaization. In this dataset is visualized using python libraries, like matplotlib, seaborn, pandas, numoy, etc. The visualizatiin part has the representation of the polarity in the form of a pie chart which is given in the dataset. The dataset has 2 polarities 0 and 1. They are visualized with the help of a pie chart as shown below in the figure.



(Fig - 5)
Pie chart of polarity

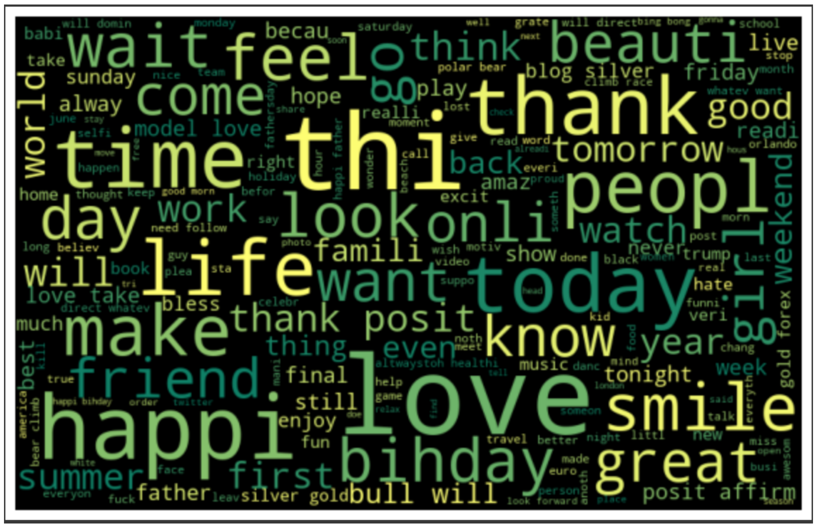
Next is the checking the distribution of of the lengths of the tweets in both the files using a histogram. The histogram of the length comparison shown below in the figure.



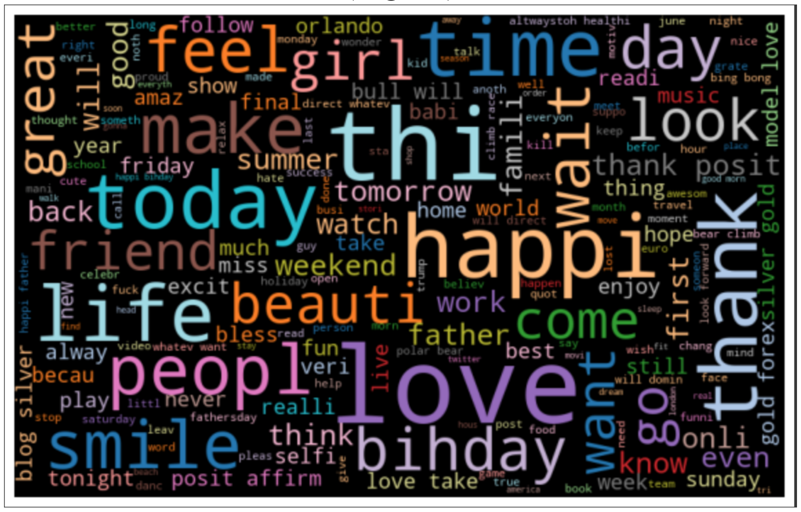
(Fig -6)
Histogram for length of tweets

The analysis of the dataset includes generation of the word clouds of the words in the dataset, first of all word cloud of all the words, then a separate one for positive words and the another one of the negative words, id generated using wordcloud library. Wordcliud is used to represent the frequency of the words in the dataset, larger the size of the word more is its frequency in the dataset.

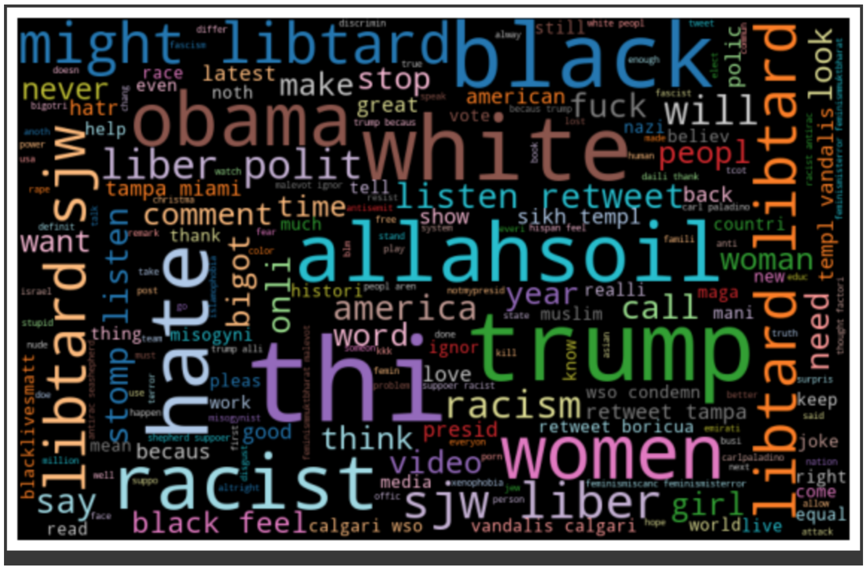
The three wordclouds discussed above are shown in the figures below.



Wordcloud of all the words
(Fig - 7)

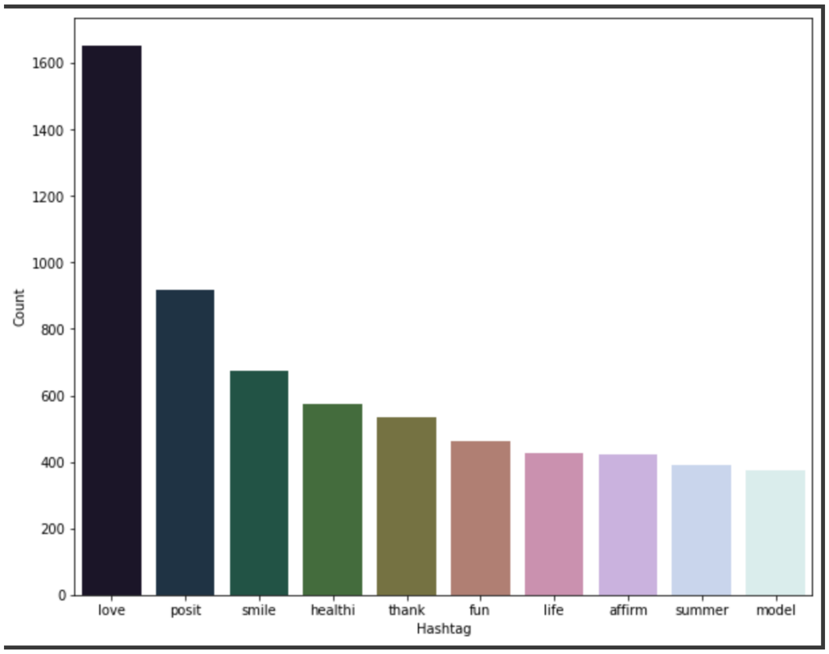


Wordcloud of positive words
(Fig - 8)



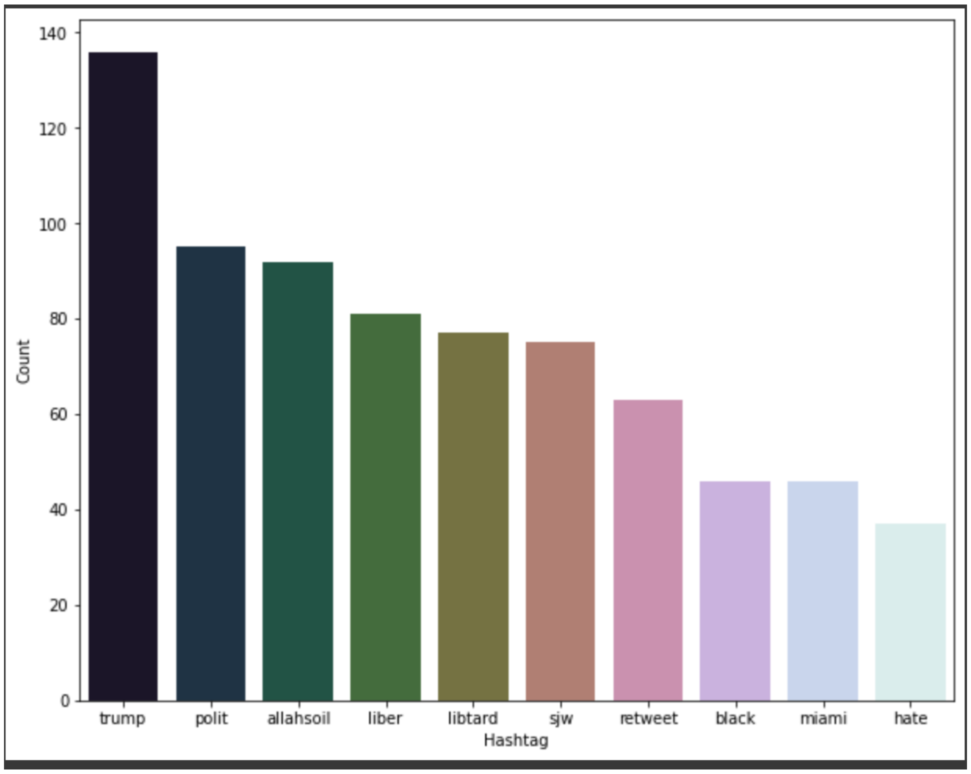
WordCloud of negative words
(Fig - 9)

The next analysis is the analysis of the hashtags, because hashtags used in tweets are very helpful in understanding what all is the tweet actually about, so only these punctualtions marks, symbols are important for us to keep them in the dataset and rest of them can be removed. The usage of these hashtags are displayed with the help of bar plots. Two different barplots for positive and negative hashtags are used.



Positive Hashtags

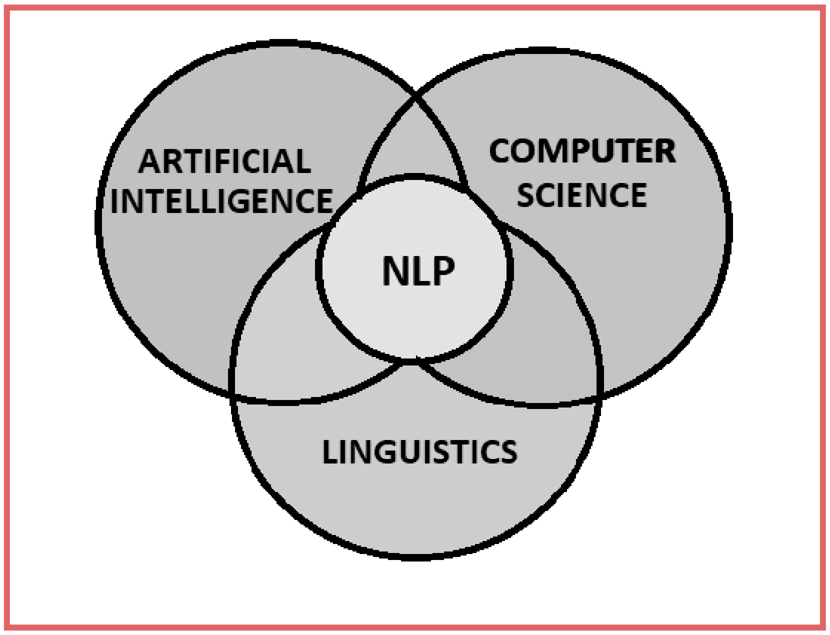
(Fig - 10)



Negative Hashtags

(Fig - 11)

NATURAL LANGUAGE PROCESSING



(Fig - 12)

NLP alludes to the part of software engineering — and all the more explicitly, the part of man-made reasoning or artificial intelligence — worried about empowering PCs to comprehend text and expressed words similarly people can.

NLP consolidates computational phonetics — rule-based demonstrating of human language — with factual, AI, and profound learning models. Together, these advancements empower PCs to handle human language as message or voice information and to 'comprehend' its full significance, complete with the speaker or essayist's aim and feeling.

NLP drives PC programs that decipher text starting with one language then onto the next, answer spoken orders, and sum up huge volumes of text quickly — even progressively. There's a decent opportunity you've cooperated with NLP as voice-worked GPS frameworks, computerized partners, discourse to-message transcription programming, client support chatbots, and other customer comforts. Yet, NLP likewise assumes a filling part in big business arrangements that assist with smoothing out business tasks, increment worker efficiency, and improve on strategic business processes.

BAG OF WORDS

It is an NLP technique used for the modelling of the text. It is nothing but a method of feature extraction in NLP. It is a very easy approach of extracting features from the data. A bag of words is a portrayal of text that depicts the event of words inside a report. We simply monitor word counts and dismissal the linguistic subtleties and the word request. It is known as a "bag" of words in light of the fact that any data about the request or construction of words in the report is disposed of. The model is just worried about whether realized words happen in the data, not where in the data.
One of the most serious issues with text is that it is muddled and unstructured, and AI calculations incline toward organized, distinct fixed-length inputs and by utilizing the bag of-Words strategy we can change over factor length texts into a fixed-length vector.

Likewise, at a much deeper level, the AI models work with mathematical information as opposed to literary information. So more specifically, by utilizing the bag of-words (BoW) strategy, we convert a text into its comparable vector of numbers.

LIMITATIONS OF BAG OF WORDS

- The model overlooks the area data of the word. The area data is a piece of vital data in the text. For instance "today is off" and "Is today off", have precisely the same vector portrayal in the BoW model.
- Bag of word models doesn't regard the semantics of the word. For instance, words 'soccer' and 'football' are much of the time utilized in a similar setting. Nonetheless, the vectors relating to these words are very disparate clinched of words model. The issue turns out to be more serious while displaying sentences.

Ex: "Purchase utilized vehicles" and "Buy old cars" are addressed by entirely unexpected vectors Clinched of-words model.

- The scope of jargon is a major issue looked by the Bag of-Words model. For instance, on the off chance that the model goes over another word it has not seen at this point, rather we say an intriguing, however educational word like Biblioklept(means one who takes books). The BoW model will likely wind up overlooking this word as this word has not been seen by the model yet.

TF-IDF

The scoring technique being utilized above takes the count of each word and addresses the word in the vector by the quantity of counts of that specific word. What does a word having high word count connote?

Does this imply that the word is significant in recovering data about archives? The response is NO. Allow me to make sense of, in the event that a word happens commonly in a record yet additionally alongside numerous different reports in our dataset, perhaps it is on the grounds that this word is only a regular word; not on the grounds that it is significant or significant.

One methodology is to rescale the recurrence of words by how frequently they show up in all reports so the scores for continuous words like "the" that are additionally successive across all records are punished. This approach is called term recurrence opposite record recurrence or without further ado known as Tf-Idf approach of scoring.TF-IDF is planned to reflect how important a term is in a given report. So how is Tf-Idf of a report in a dataset determined?

TF-IDF for a word in a report is determined by duplicating two distinct measurements:

The term frequency (TF) of a word in a report. There are multiple approaches to computing this recurrence, with the easiest being a crude count of cases a word shows up in a record. Then, there are alternate ways of changing the recurrence. For instance, by separating the crude count of cases of a word by one or the other length of the record, or by the crude recurrence of the most continuous word in the report. The equation to ascertain Term-Recurrence is

$TF(i,j)=n(i,j)/\Sigma\ n(i,j)$

Where,

$n(i,j)$= number of times nth word happened in a report

$\Sigma n(i,j)$ = complete number of words in a record.

The inverse document frequency (IDF) of the word across a bunch of reports. This recommends how normal or uncommon a word is in the whole record set. The nearer it is to 0, the more normal is the word. This measurement can be determined by taking the all out number of records, separating it by the quantity of reports that contain a word, and computing the logarithm**.**

WORD2VEC

Word2Vec is a new forward leap in the realm of NLP. Tomas Mikolov a Czech PC researcher and right now a scientist at CIIRC was one of the main donors towards the exploration and execution of word2vec. Word embeddings are a necessary piece of tackling numerous issues in NLP. They portray how people figure out language to a machine. You can envision them as a vectorized portrayal of text. Word2Vec, a typical technique for producing word embeddings, has different applications like message comparability, proposal frameworks, opinion investigation, and so on.

Before we get into word2vec, how about we lay out a comprehension of what word embeddings are. This is critical to know in light of the fact that the general outcome and result of word2vec will be embeddings related to every special word went through the calculation.

Word embeddings is a procedure where individual words are changed into a mathematical portrayal of the word (a vector). Where each word is planned to one vector, this vector is then educated in a way which looks like a brain organization. The vectors attempt to catch different attributes of that word with respect to the general text. These attributes can incorporate the semantic relationship of the word, definitions, setting, and so on. With these mathematical portrayals, you can do numerous things like distinguish comparability or uniqueness between words.

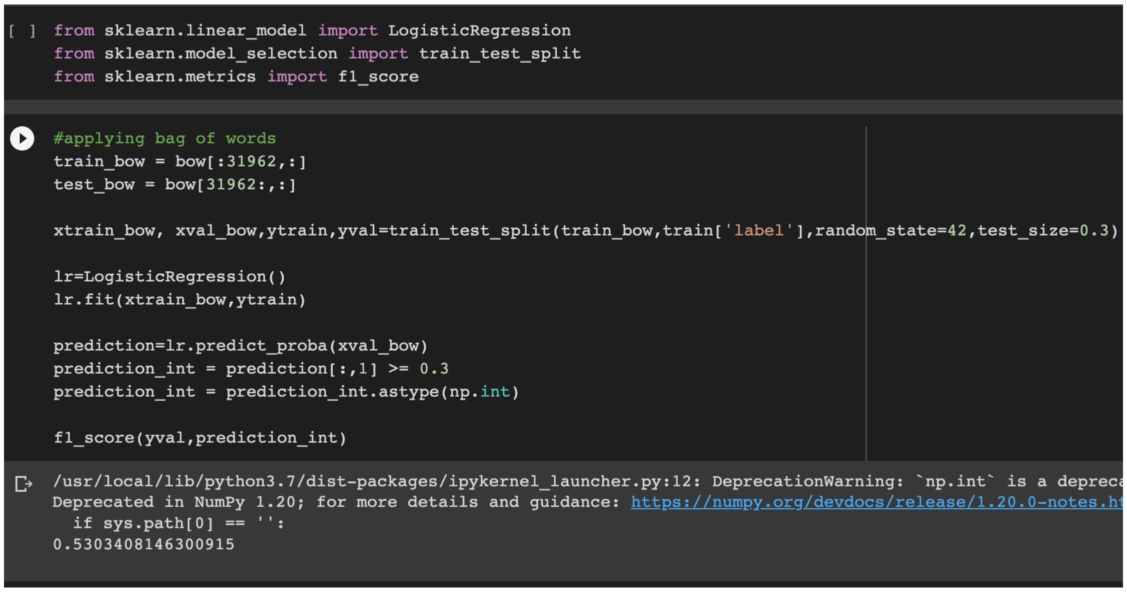
The adequacy of Word2Vec comes from its capacity to gather vectors of comparative words. Given an enormous enough dataset, Word2Vec can areas of strength for make about a word's significance in view of their events in the text. These assessments yield word relationship with different words in the corpus. For instance, words like "Ruler" and "Sovereign" would be basically the same as each other. While directing mathematical procedure on word embeddings you can track down a nearby guess of word likenesses. For instance, the 2 layered inserting vector of "ruler" - the 2 layered installing vector of "man" + the 2 layered implanting vector of "lady" yielded a vector which is exceptionally near the installing vector of "sovereign".

In our project we have applied all the three methods which are written above , and then we have compared their F1 scores and found out which one performs better in our case. After the implementation of all the above three models W2V comes out as a better performer than the other two models. TF-IDF and BoW were almost the same.

The respective F1 scores of the three models were as follows:

- Bow = 0.5303408146300915
- TF-IDF = 0.5451327433628319
- W2V = 0.6176688938381588

The snapshot of the output of these f1 scores is as shown below.

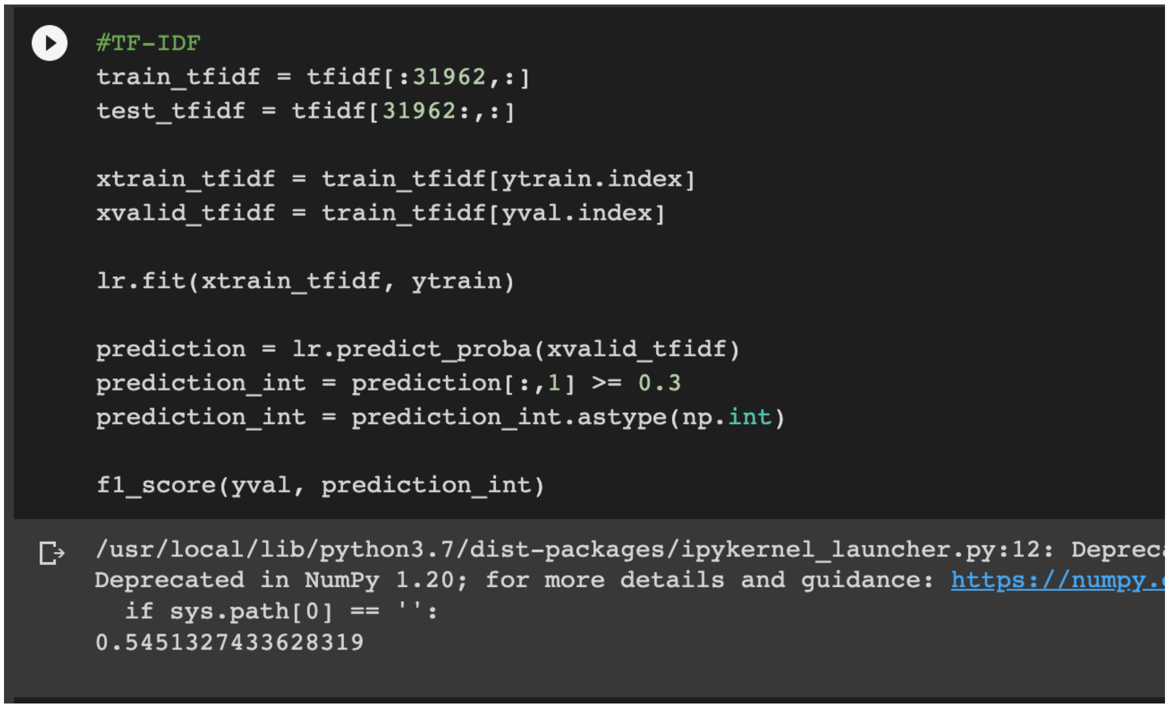


```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score
```

```
#applying bag of words
train_bow = bow[:31962,:]
test_bow = bow[31962:,:]

xtrain_bow, xval_bow,ytrain,yval=train_test_split(train_bow,train['label'],random_state=42,test_size=0.3)

lr=LogisticRegression()
lr.fit(xtrain_bow,ytrain)

prediction=lr.predict_proba(xval_bow)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

f1_score(yval,prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: DeprecationWarning: `np.int` is a depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.ht
  if sys.path[0] == '':
0.5303408146300915
```

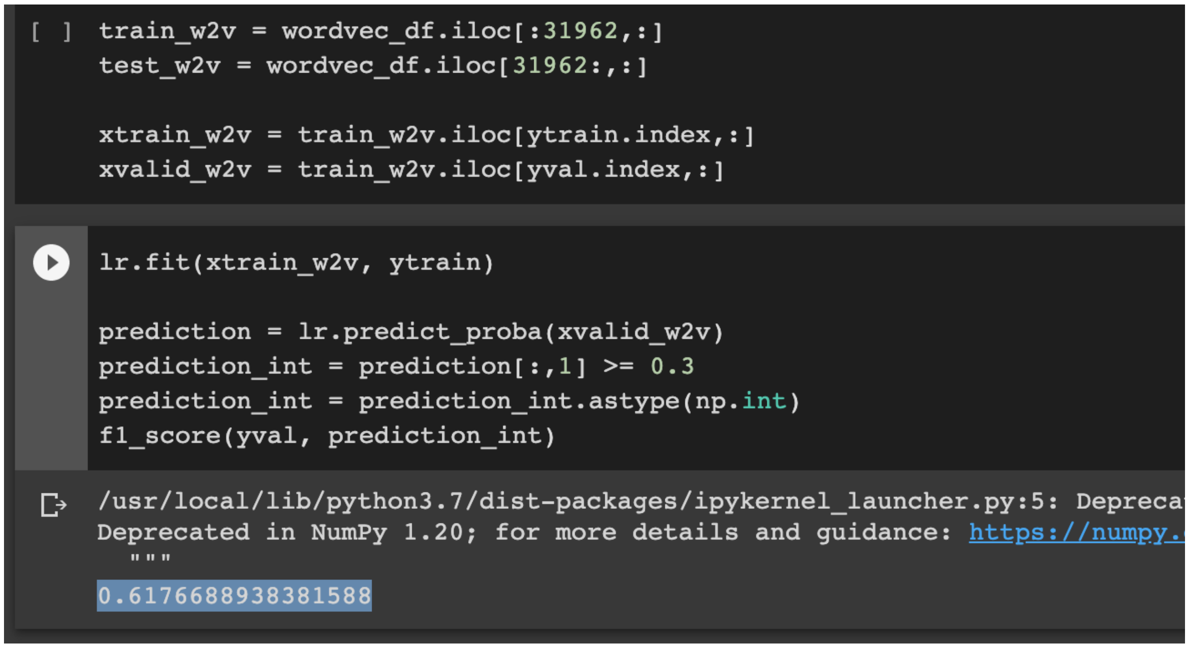
(Fig - 13)



```
#TF-IDF
train_tfidf = tfidf[:31962,:]
test_tfidf = tfidf[31962:,:]

xtrain_tfidf = train_tfidf[ytrain.index]
xvalid_tfidf = train_tfidf[yval.index]

lr.fit(xtrain_tfidf, ytrain)

prediction = lr.predict_proba(xvalid_tfidf)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

f1_score(yval, prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: Depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.
  if sys.path[0] == '':
0.5451327433628319
```

(Fig -14)



```
[ ] train_w2v = wordvec_df.iloc[:31962,:]
    test_w2v = wordvec_df.iloc[31962:,:]

    xtrain_w2v = train_w2v.iloc[ytrain.index,:]
    xvalid_w2v = train_w2v.iloc[yval.index,:]
```

```
lr.fit(xtrain_w2v, ytrain)

prediction = lr.predict_proba(xvalid_w2v)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
f1_score(yval, prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: Depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.
  """
0.6176688938381588
```

(Fig - 15)

Clearly from the above snapshots it is visible that the W2V has performed better because it has better f1 score.

MATHEMATICAL INTERPRETATION OF THE THREE APPROCAHES USED

- ***BAG OF WORDS*** **:** The fundamental concept behind the bag-of-words representation is to convert each important concept into one of the visual words that are frequently produced through clustering. By considering the mapping to a visual word, v_k, as a quantization function, $f_k(x)$:X, we can generalise this notion of quantization. We suppose that the quantization function $f_k(x)$ is randomly selected from a class of functions, denoted by F, via an unknown distribution P_F due to the uncertainty in vocabulary construction. We create the function F class in the manner shown below to capture the behaviour of quantization.

$$\mathcal{F} = \{f(\boldsymbol{x}; \boldsymbol{v}) | f(\boldsymbol{x}; \boldsymbol{v}) = I(\|\boldsymbol{x} - \boldsymbol{v}\| \leq \rho), \boldsymbol{v} \in \mathcal{X}\}$$

- ***TF-IDF* :** TF-IDF is calculated for each word in a document by multiplying two separate metrics: The number of times a word appears in a document. The simplest method of determining this frequency is to simply count the number of times a word appears in a document. The length of a document or the frequency of the word that appears the most frequently in a document are other ways to modify frequency. The word's average inverse document frequency in a group of documents. This refers to how prevalent or uncommon a word is throughout all documents. A word is more common the nearer it is to 0. To calculate this measure, multiply the overall number of documents by counting the instances of a term in documents and computing the logarithm. This number will therefore be close to zero if the word is widely used and appears in numerous papers. If not, it will go close to 1. The result of multiplying these two figures is the word's TF-IDF score in a document. The more relevant a word is in a given document, the higher the score. In more precise mathematical terms, the following is how the TF-IDF score for the word t in the document d from the document set D is determined:

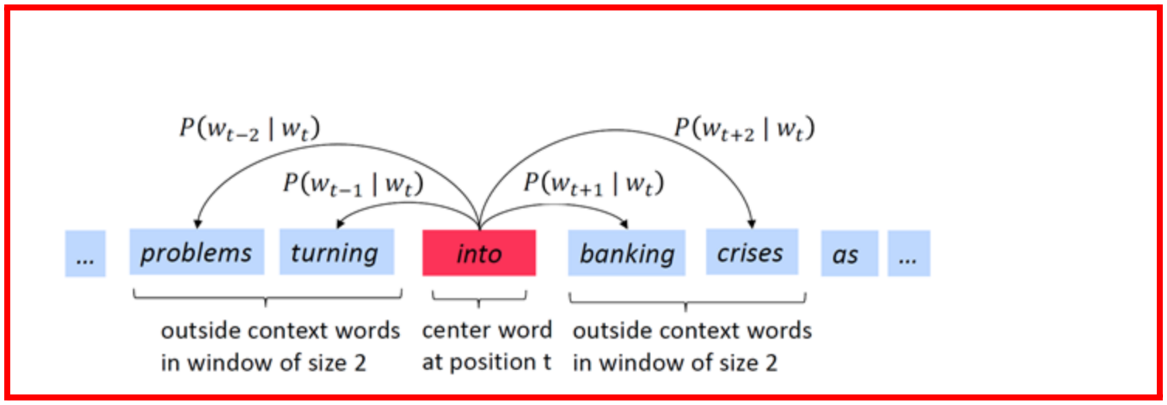
$$tf\,idf\,(t, d, D) = tf\,(t, d)\ .\ idf\,(t, D)$$

Where:

$$tf\,(t, d) = log\,(1 + freq\,(t, d))$$

$$idf\,(t, D) = log\,\left(\frac{N}{count\,(d \in D{:}t \in d)}\right)$$

- ***Word2Vec* :** Word2Vec can be implemented using an algorith called the Skip-gram Algorithm : The primary principle of the approach is to initialise the vectors for each word in the vocabulary at random first. The centre word at each location t will then be defined as c, and its context word will be defined as o, as we move through the positions t one by one. We will define a window of size m, which indicates that our model will look at words in position t-m to t+m as the context, in order to detect the words.



Some formulae used in this algorithm are displayed using the snapshots give below:

$$L(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

The above formula is used to maximize the likelihood of the context words.

TOKENISATION

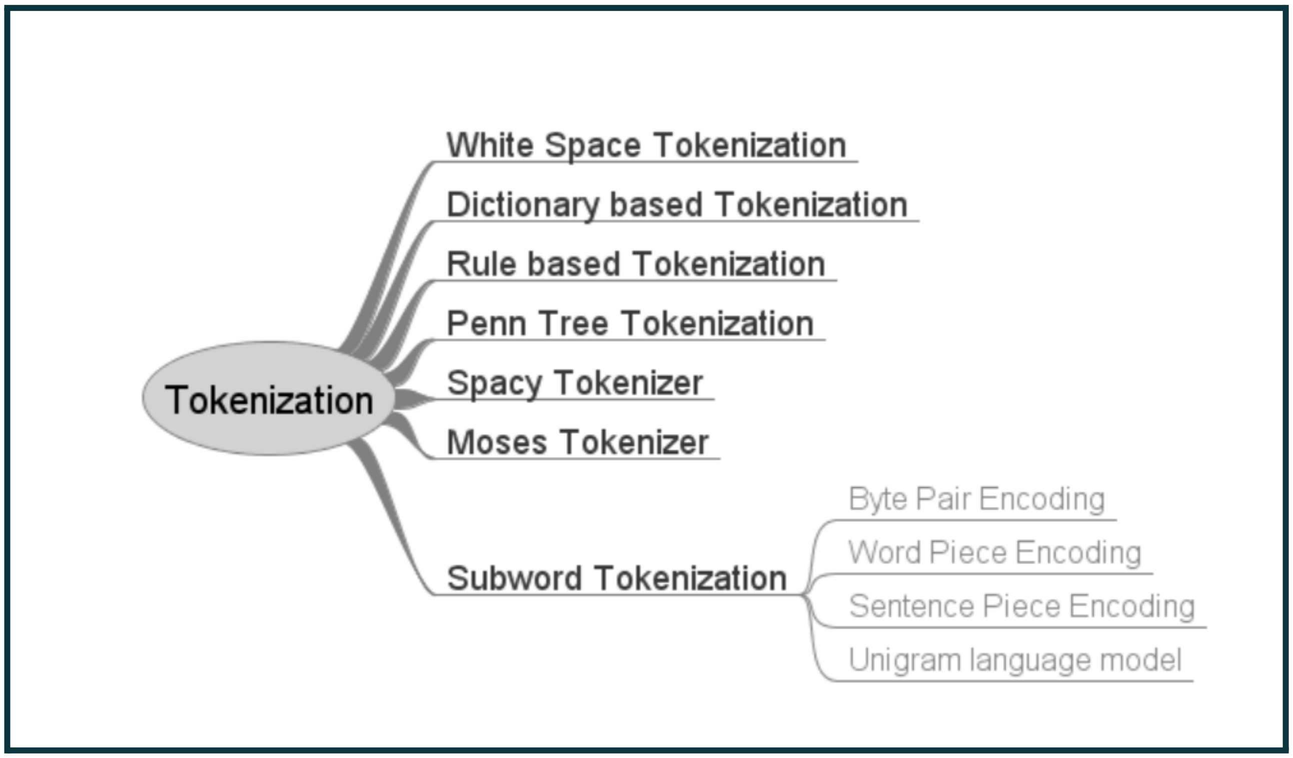
Tokenization involves cutting the raw text into manageable pieces. Tokenization divides the original text into tokens, which are words and sentences. These tokens aid in context comprehension or model development for NLP. By examining the word order in the text, tokenization aids in comprehending the text's meaning.

The words "It is hot," for instance, can be tokenized into "It," "is," and "hot."

Tokenization can be done using a variety of tools and frameworks. Some of the libraries that can be utilised to do the work include NLTK, Gensim, and Keras.

You can tokenize individual words or entire sentences. The process of breaking up text into words using a technology is known as word tokenization, while the process of doing the same for sentences is known as sentence tokenization.

Stop words are those in the text that don't provide any context to the sentence and removing them won't change how the content is processed for the intended purpose. To reduce noise and the size of the feature set, they are eliminated from the lexicon.



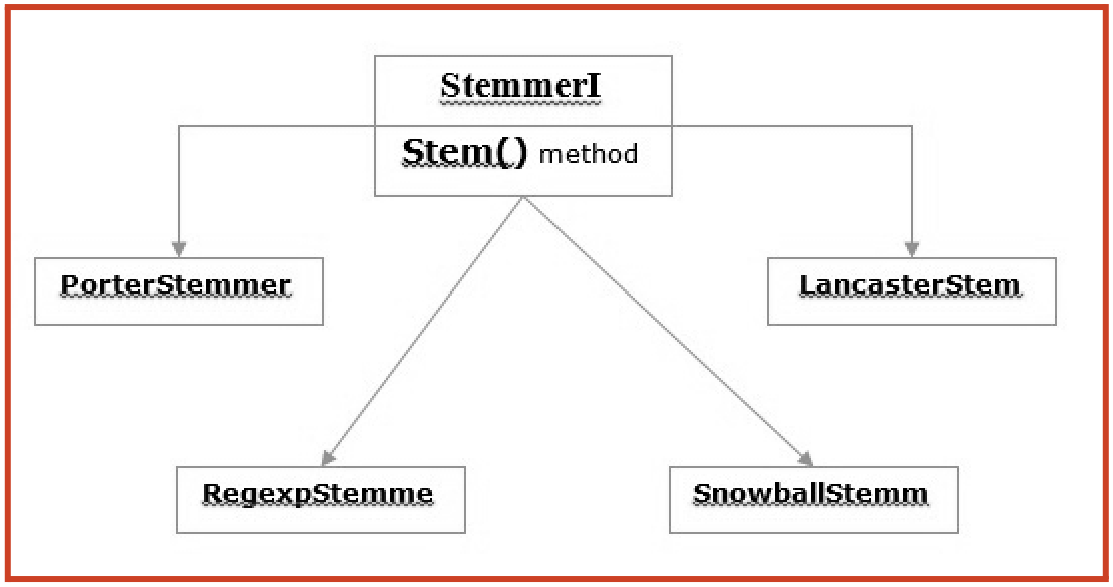
(Fig - 16)

Types of Tokenisation

STEMMING

Stemming is a method for eliminating affixes from words to reveal their basic form. It is analogous to trimming a tree's branches back to the trunk. For instance, the word eat is the root of the verbs eating, eats, and eaten.

Stemming is used by search engines to index the words. Because of this, a search engine can only record the stems of a word rather than all of its variations. Stemming does this by reducing the size of the index and improving retrieval precision.
All of the stemmers that we will examine in the following section are contained in NLTK's stemmerI interface, which provides a stem() method. Let's comprehend it using the diagram below.



(Fig -17)

Stemming

Porter's algorithm for stemming

One of the most popular stemming algorithms basically aims to replace well-known suffixes of English words with other suffixes.

Class PorterStemmer

For the word we want to stem, the Porter Stemmer methods can be quickly implemented using the NLTK's PorterStemmer class. This class is familiar with a variety of regular word forms and suffixes that it can use to change the input word into the final stem. Frequently, a shorter term with the same root meaning is the resulting stem.

Algorithm for Lancaster stemming

It is another widely used stemming method, it was created at Lancaster University.

Class LancasterStemmer

For the term we want to stem, the NLTK includes a LancasterStemmer class that makes it simple to implement Lancaster Stemmer algorithms.

Lemmatization - What Is It?

Similar to stemming is the lemmatization approach. The final product of lemmatization is referred to as a "lemma," which is a root word as opposed to a root stem, the final product of stemming. We will obtain a valid term with the same meaning after lemmatization.

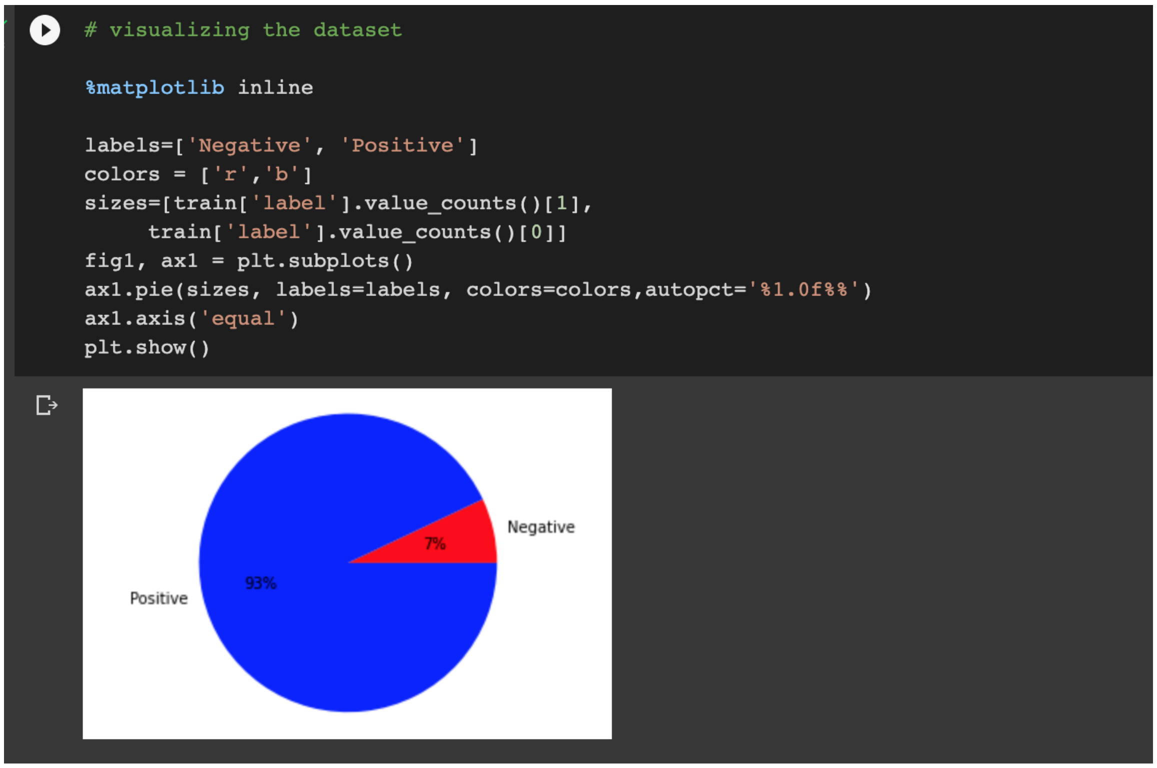
The WordNetLemmatizer class from NLTK is a simple wrapper for the WordNet corpus. To locate a lemma, this class employs the WordNet CorpusReader class's morphy() function.

CHAPTER - 4

EXPERIMENTS AND RESULT ANALYSIS

Initially the data is analysed on the basis of the polarity of the tweets.) and 1 are the values of the polarity. In our dataset it is 0 for the positive tweets and 1 for the negative tweets.

This result is displayed with the help of a pie chart using the matplot library of the python. Below is the snapshot of the piechart and its code snippet.

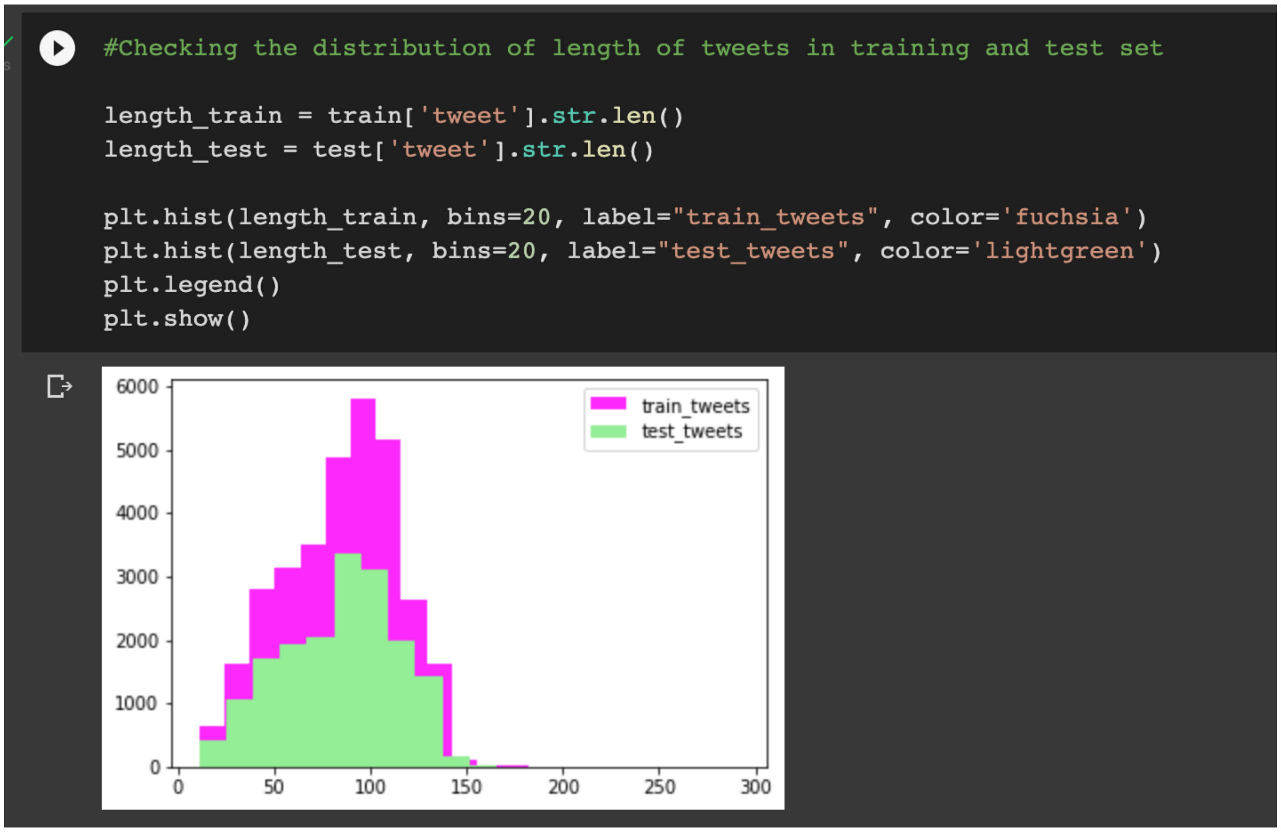


```
# visualizing the dataset

%matplotlib inline

labels=['Negative', 'Positive']
colors = ['r','b']
sizes=[train['label'].value_counts()[1],
     train['label'].value_counts()[0]]
fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, colors=colors,autopct='%1.0f%%')
ax1.axis('equal')
plt.show()
```

(Fig - 18)

Analysis of dataset

The next result that we are trying to find out using the analysis of the dataset is the length of the tweets compared in test and train csv files. As we can clearly see in the figure that the length distribution the tweets is some what same in both the files.

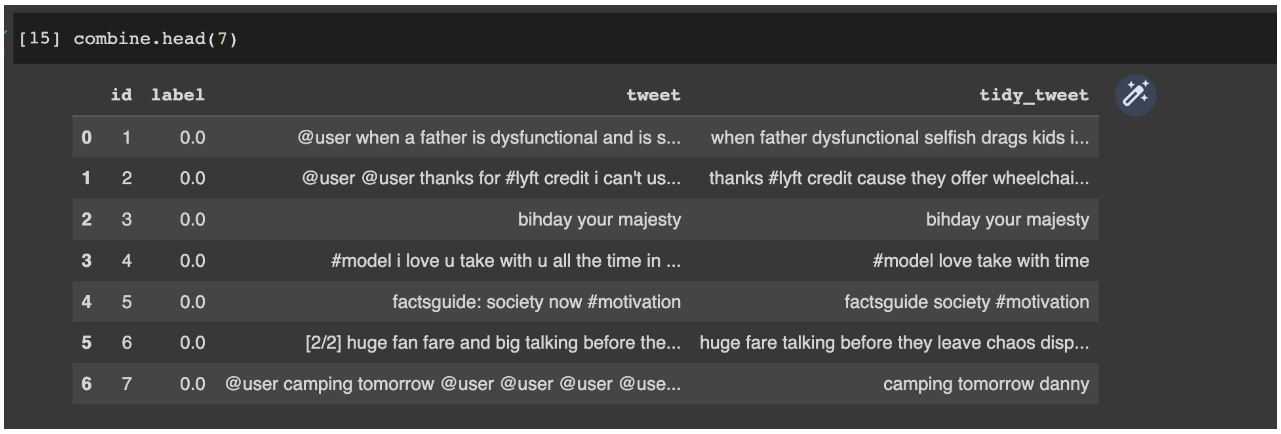


```
#Checking the distribution of length of tweets in training and test set

length_train = train['tweet'].str.len()
length_test = test['tweet'].str.len()

plt.hist(length_train, bins=20, label="train_tweets", color='fuchsia')
plt.hist(length_test, bins=20, label="test_tweets", color='lightgreen')
plt.legend()
plt.show()
```

(Fig - 19)

Comparison of the lengths of tweets implementation

In the preprocessing steps we first combined the test and train files and got the result in the form of new csv file which is named combine. We have pasted a sample of the file below.



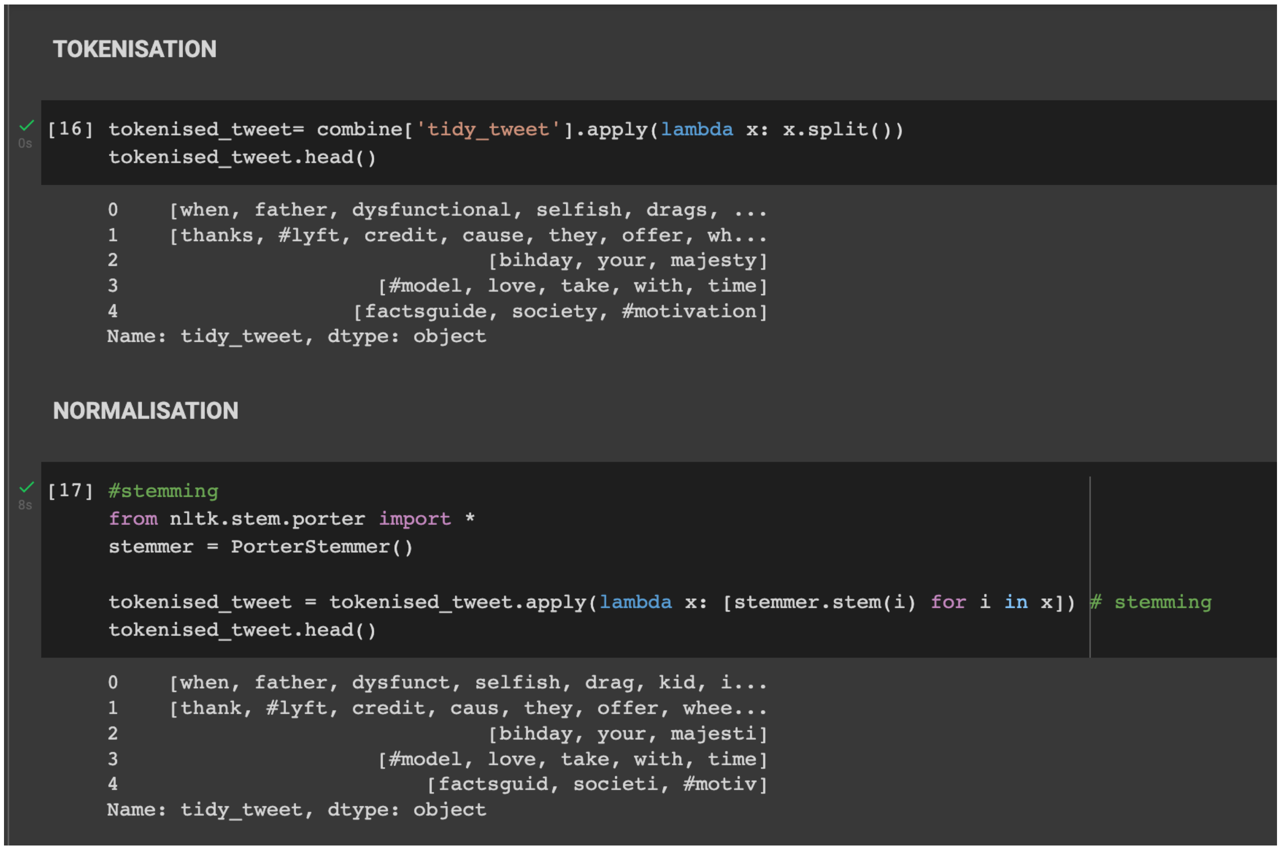
```
[15] combine.head(7)
```

	id	label	tweet	tidy_tweet
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0.0	bihday your majesty	bihday your majesty
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguide society #motivation
5	6	0.0	[2/2] huge fan fare and big talking before the...	huge fare talking before they leave chaos disp...
6	7	0.0	@user camping tomorrow @user @user @user @use...	camping tomorrow danny

(Fig - 20)

Combined files

There were some more steps involved in the process of preprocessing they were tokenisation, stemming, etc. There results are shown below.



TOKENISATION

```
[16] tokenised_tweet= combine['tidy_tweet'].apply(lambda x: x.split())
     tokenised_tweet.head()
```

```
0    [when, father, dysfunctional, selfish, drags, ...
1    [thanks, #lyft, credit, cause, they, offer, wh...
2                               [bihday, your, majesty]
3                      [#model, love, take, with, time]
4                    [factsguide, society, #motivation]
Name: tidy_tweet, dtype: object
```

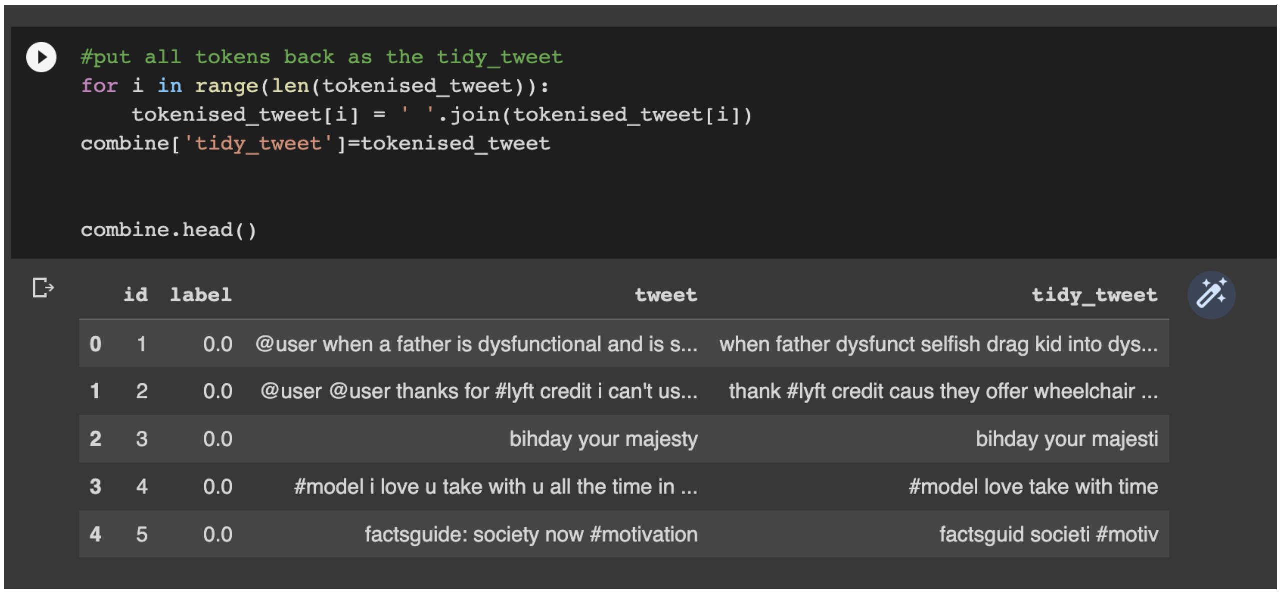
NORMALISATION

```
[17] #stemming
     from nltk.stem.porter import *
     stemmer = PorterStemmer()

     tokenised_tweet = tokenised_tweet.apply(lambda x: [stemmer.stem(i) for i in x]) # stemming
     tokenised_tweet.head()
```

```
0    [when, father, dysfunct, selfish, drag, kid, i...
1    [thank, #lyft, credit, caus, they, offer, whee...
2                               [bihday, your, majesti]
3                      [#model, love, take, with, time]
4                          [factsguid, societi, #motiv]
Name: tidy_tweet, dtype: object
```

(Fig - 21)

Tokenisation and normalisation



```
#put all tokens back as the tidy_tweet
for i in range(len(tokenised_tweet)):
    tokenised_tweet[i] = ' '.join(tokenised_tweet[i])
combine['tidy_tweet']=tokenised_tweet


combine.head()
```

	id	label	tweet	tidy_tweet
0	1	0.0	@user when a father is dysfunctional and is s...	when father dysfunct selfish drag kid into dys...
1	2	0.0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit caus they offer wheelchair ...
2	3	0.0	bihday your majesty	bihday your majesti
3	4	0.0	#model i love u take with u all the time in ...	#model love take with time
4	5	0.0	factsguide: society now #motivation	factsguid societi #motiv

(Fig - 22)

These results show us that the preprocessing has been done effectively and there are no chances of the mistakes in the dataset now. Both the tokenisation and normalisation are the important steps in the preprocessing and hence their results should be staisfcatory, otherwise there will be a lot of repercussions while we try to run our model. As we can see in the figures that the length of the sentences have been reduced after the tokeinsation process of the tweets, these sentences are more easily iterpretable and they canbe easily processed in the model.

After this word clouds were generated and hashtags were analysed for positive and negative tweets. There results are discusses earlier also in the report.

Text Features Extraction

Features of Bag of Words include the creation of a matrix of the distinct tokens and their corresponding frequencies in all documents in a corpus. This will just detect the word's presence and not its context. Reference

Features of TF-IDF: similar to BOW, but gives unusual words more weight. In other words, terms that frequently appear in a small number of tweets are given greater weight than words that appear in all tweets. Reference

Word Embeddings (Word2Vec) are text representations that maintain semantics (in a way). Here, similar representations will be assigned to words with similar meanings. Word2Vec provides a word's numerical representation while maintaining the connection between words (such as synonyms or antonyms).
First of all features are extracted using the above three methods nd then these models are implemented using simple logistic regression to compare the result which method gives the best result on this particular dataset.
From our implementations we have derived the conclusion that Word2vec has outperformed both the other models in the performance. The performance measure that we have used in F1 score. F1 score is used because it gives more precise results than any other performance measure.

F1 SCORE

The accuracy of a model on a dataset is gauged by the F-score, also known as the F1-score. It's employed to assess binary categorization schemes that label examples as "positive" or "negative."

The harmonic mean of the model's precision and recall is known as the F-score, which is a method of combining the model's precision and recall.

The F-score is frequently used to assess machine learning models, particularly those employed in natural language processing, as well as information retrieval systems like search engines.

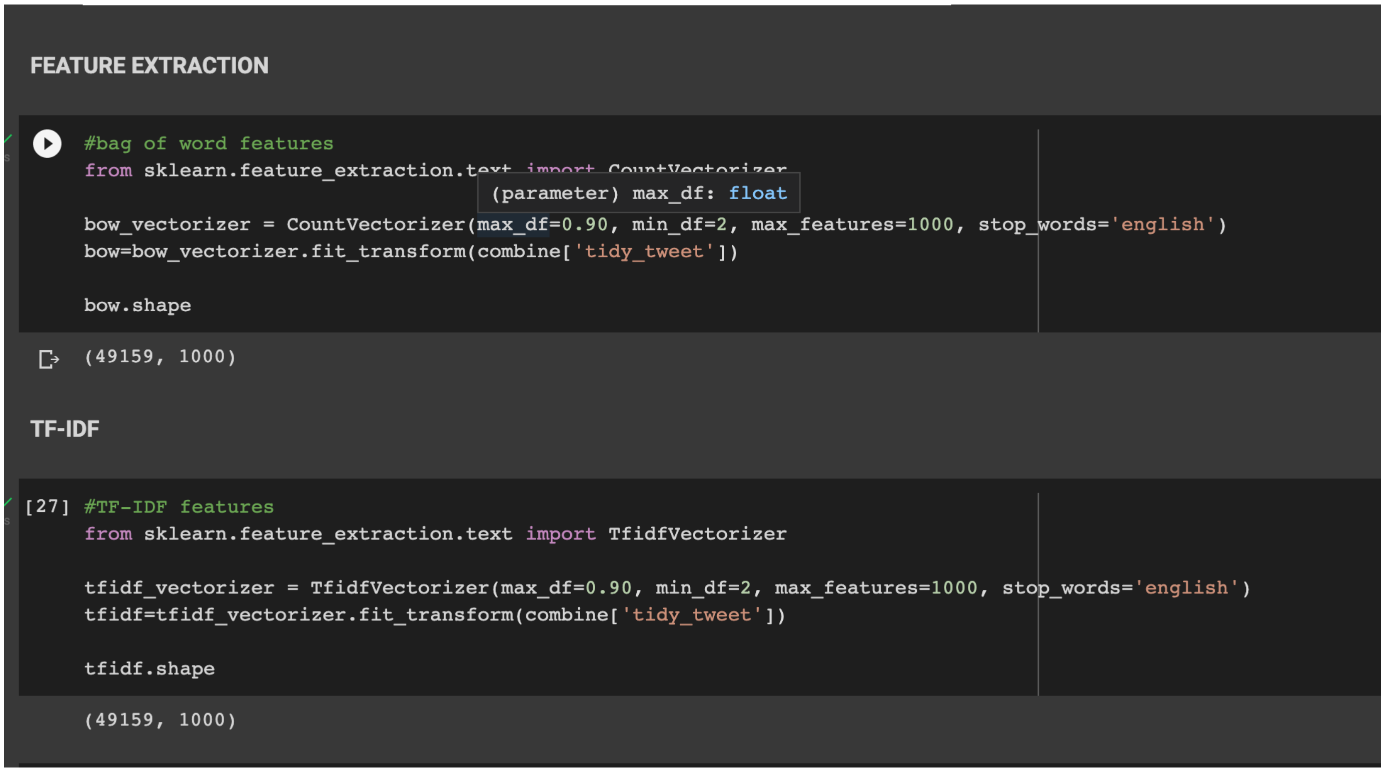
It is possible to change the F-score such that precision is valued more highly than recall, or the other way around. Along with the conventional F1-score, common adjusted F-scores include the F0.5-score and the F2-score.

Precision : The proportion of real positive examples among those that the model classified as positive is known as precision. To put it another way, the ratio of genuine positives to false positives + true positives.

Recall : The percentage of examples out of all positive examples that are categorised as positive is called recall, also known as sensitivity. Alternatively stated, the ratio of genuine positives to true positives

$$\text{F1 score} = 2 \bullet \frac{\text{Precision} \bullet \text{Recall}}{\text{Precision} + \text{Recall}}$$

The screenshots of the results are shown below:



FEATURE EXTRACTION

```
#bag of word features
from sklearn.feature_extraction.text import CountVectorizer
                                 (parameter) max_df: float
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
bow=bow_vectorizer.fit_transform(combine['tidy_tweet'])

bow.shape
```

```
(49159, 1000)
```

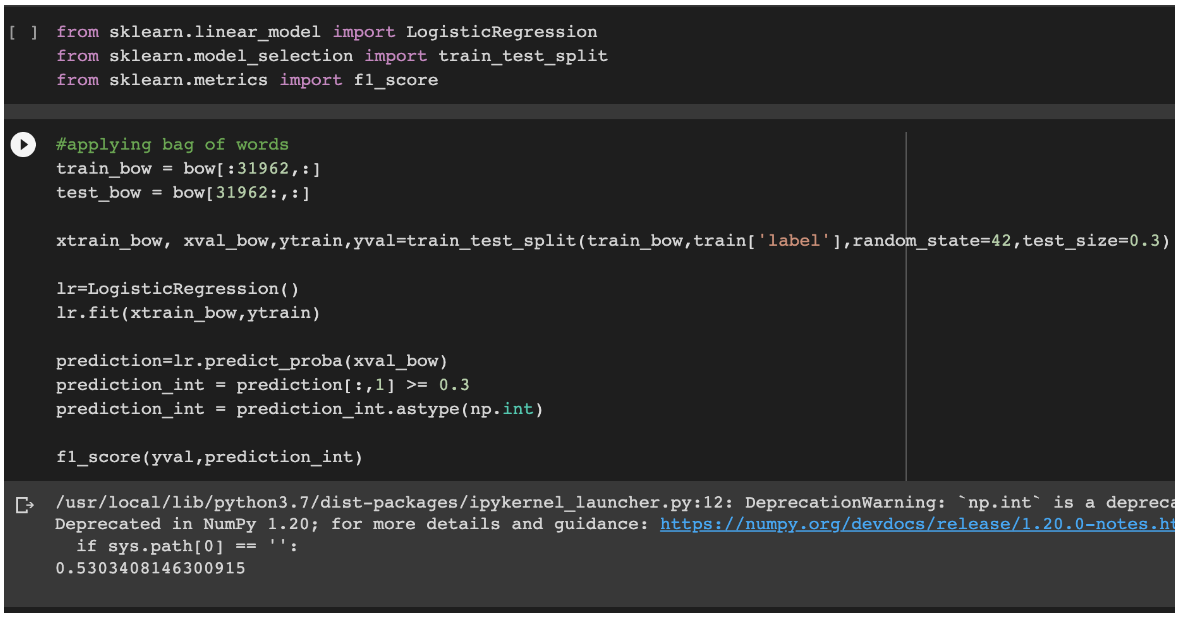
TF-IDF

```
[27] #TF-IDF features
     from sklearn.feature_extraction.text import TfidfVectorizer

     tfidf_vectorizer = TfidfVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
     tfidf=tfidf_vectorizer.fit_transform(combine['tidy_tweet'])

     tfidf.shape
```

```
(49159, 1000)
```

(Fig - 23)

IMPLEMENTATION OF MODELS

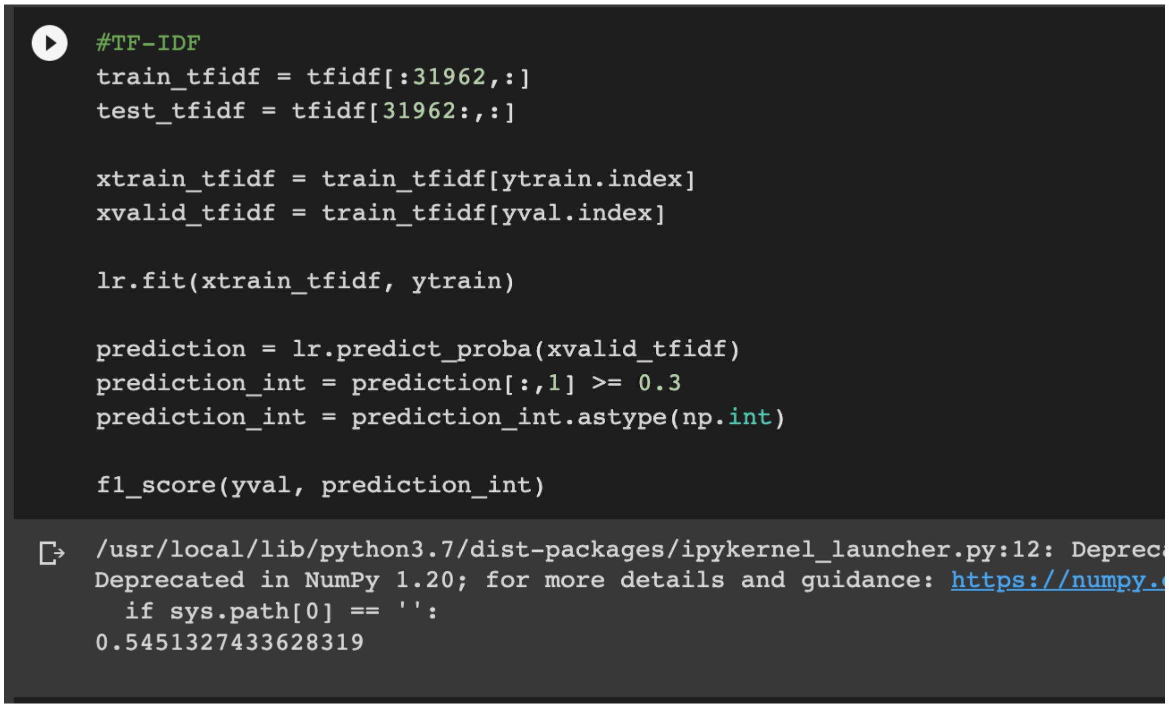


```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import f1_score
```

```
#applying bag of words
train_bow = bow[:31962,:]
test_bow = bow[31962:,:]

xtrain_bow, xval_bow,ytrain,yval=train_test_split(train_bow,train['label'],random_state=42,test_size=0.3)

lr=LogisticRegression()
lr.fit(xtrain_bow,ytrain)

prediction=lr.predict_proba(xval_bow)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

f1_score(yval,prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: DeprecationWarning: `np.int` is a depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.ht
  if sys.path[0] == '':
0.5303408146300915
```

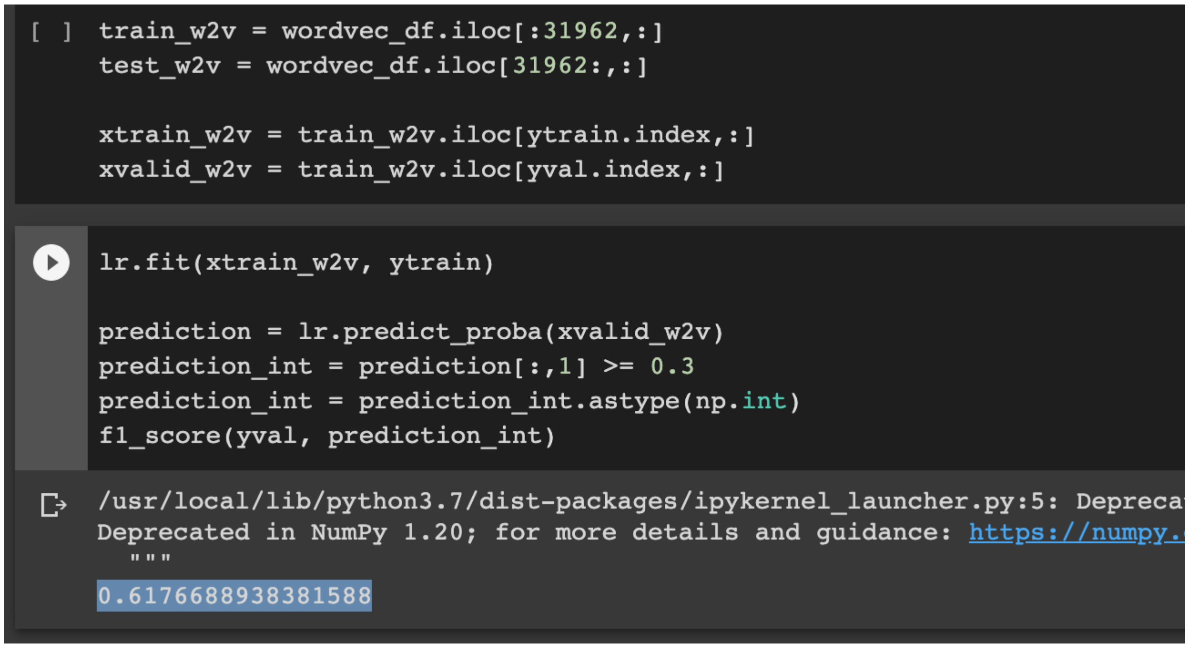
(Fig - 24)



```
#TF-IDF
train_tfidf = tfidf[:31962,:]
test_tfidf = tfidf[31962:,:]

xtrain_tfidf = train_tfidf[ytrain.index]
xvalid_tfidf = train_tfidf[yval.index]

lr.fit(xtrain_tfidf, ytrain)

prediction = lr.predict_proba(xvalid_tfidf)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

f1_score(yval, prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: Depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.
  if sys.path[0] == '':
0.5451327433628319
```

(Fig - 25)



```
train_w2v = wordvec_df.iloc[:31962,:]
test_w2v = wordvec_df.iloc[31962:,:]

xtrain_w2v = train_w2v.iloc[ytrain.index,:]
xvalid_w2v = train_w2v.iloc[yval.index,:]
```

```
lr.fit(xtrain_w2v, ytrain)

prediction = lr.predict_proba(xvalid_w2v)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
f1_score(yval, prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: Depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.
  """
0.6176688938381588
```

(Fig - 26)

F1 score

MATPLOTLIB

For 2D displays of arrays, Matplotlib is a fantastic Python visualisation library. A multi-platform data visualisation package called Matplotlib was created to deal with the larger SciPy stack and is based on NumPy arrays. In the year 2002, John Hunter first presented it.

One of visualization's biggest advantages is that it gives us visual access to vast volumes of data in forms that are simple to understand. There are numerous plots in Matplotlib, including line, bar, scatter, histogram, etc.

It can be implemented as follows :

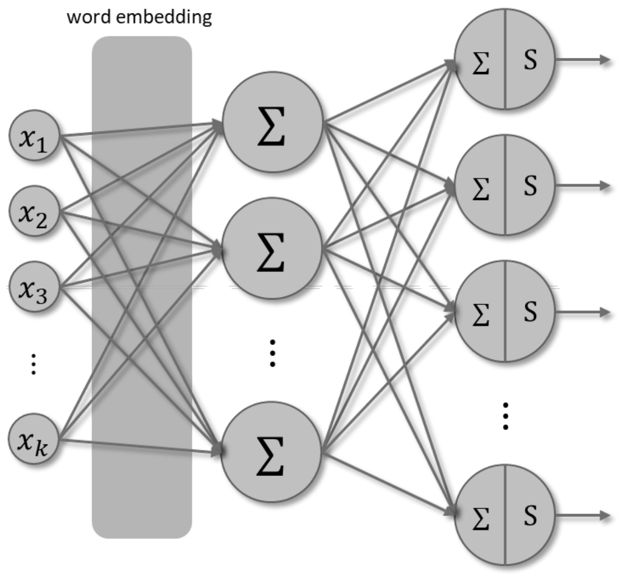
```
from matplotlib import pyplot as plt
or
import matplotlib.pyplot as plt
```

There are numerous plot types available in Matplotlib. Plots aid in recognising trends, patterns, and relationships. They are frequently tools for reasoning about quantitative data.

Some of them are :

- Line plot
- Bar Plot
- Histogram
- Scatter Plot
- Pie chart

The performance analysis of the three models shows that W2V is a better performer as it uses a dense neural network with a single hidden layer which has not activation function.



(Fig - 27)

Neural Network

Some Limitations of bag of words are :

- It's terrible news when you have a huge corpus because the feature dimension is directly dependent on the number of unique tokens. To conserve space, you could eliminate some of the least frequent tokens, but doing so would also eliminate some of the data's potentially relevant information.
- You can see that the first and last papers from the data example above are different documents even if they share the same feature vector. This is so that associations between tokens can no longer be preserved by the bag of words.

TF-IDF is a mapping for word documents (with some normalization). It disregards word order and outputs a nxm matrix (or mxn matrix, depending on implementation), where n is the number of vocabulary words and m is the number of documents. On the other hand, Word2Vec generates a distinct vector for each word depending on the words that appear in its vicinity. Simple linear algebra is used to obtain the TF-IDF.

The hidden layer of a two-layered neural network yields Word2Vec. It is possible to assign vectors to words or texts using TF-IDF. Word2Vec can be used directly to assign a vector to a word, but additional processing is required to obtain the vector representation of a document. In contrast to TF-IDF, Word2Vec considers the placement of words.

CHAPTER - 5

CONCLUSIONS

The conclusion that we were able to draw from this project are that, natural language processing can be implemented using the feature extraction techniques. Sentiment analysis of the dataset provided in this project gave ok values of the f1 score. Which meant that the model worked fine.

On the other hand we were able to compare the the values of f1 scores of the three models and as discussed in the performance analysis the Word2Vec method which used neural networks was the best out of the three.

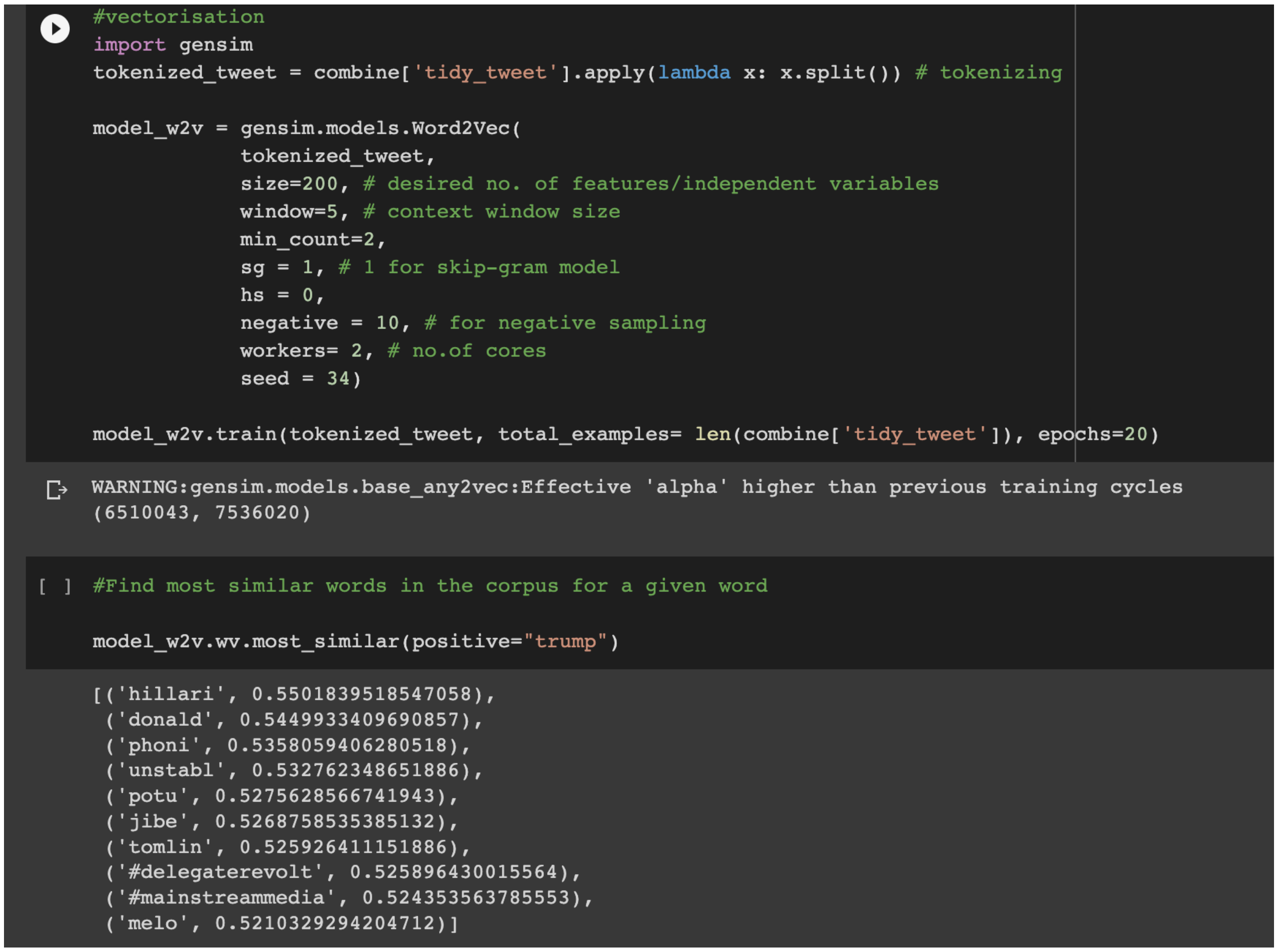
More better values of the f1 score can be obtained if some more features are looked upon in the dataset and if some other models are employed and they are compared.

The sentiments of the tweets are correctly predicted and gives satisfactory values.

Word cloud results were also satisfying as they gave us the idea to understand which word in both the categories have more frequencies can be easily judged and then can be compared with each other. Hashtags (both positive and negative) analysis was done to understand even before looking at the tweet and most used hashtags were displayed using the histograms in matplotlib library.. Vectorisation helped in grouping similar words together and the vectorisation also helped in easy implementation of the three methods used in sentiment analysis using feature extraction.

In sentiment analysis, we find tweets with a bad attitude, such as those that are racist, sexist, or just plain hateful. Here, tweets labelled "1" indicate negative tweets, while tweets labelled "0" indicate that there was no hate speech in the message.

To train a supervised machine learning model on textual data, feature extraction must be done. Instead than focusing on improving the model's performance, our major objective in this notebook is to investigate various feature extraction techniques.



```
#vectorisation
import gensim
tokenized_tweet = combine['tidy_tweet'].apply(lambda x: x.split()) # tokenizing

model_w2v = gensim.models.Word2Vec(
            tokenized_tweet,
            size=200, # desired no. of features/independent variables
            window=5, # context window size
            min_count=2,
            sg = 1, # 1 for skip-gram model
            hs = 0,
            negative = 10, # for negative sampling
            workers= 2, # no.of cores
            seed = 34)

model_w2v.train(tokenized_tweet, total_examples= len(combine['tidy_tweet']), epochs=20)
```

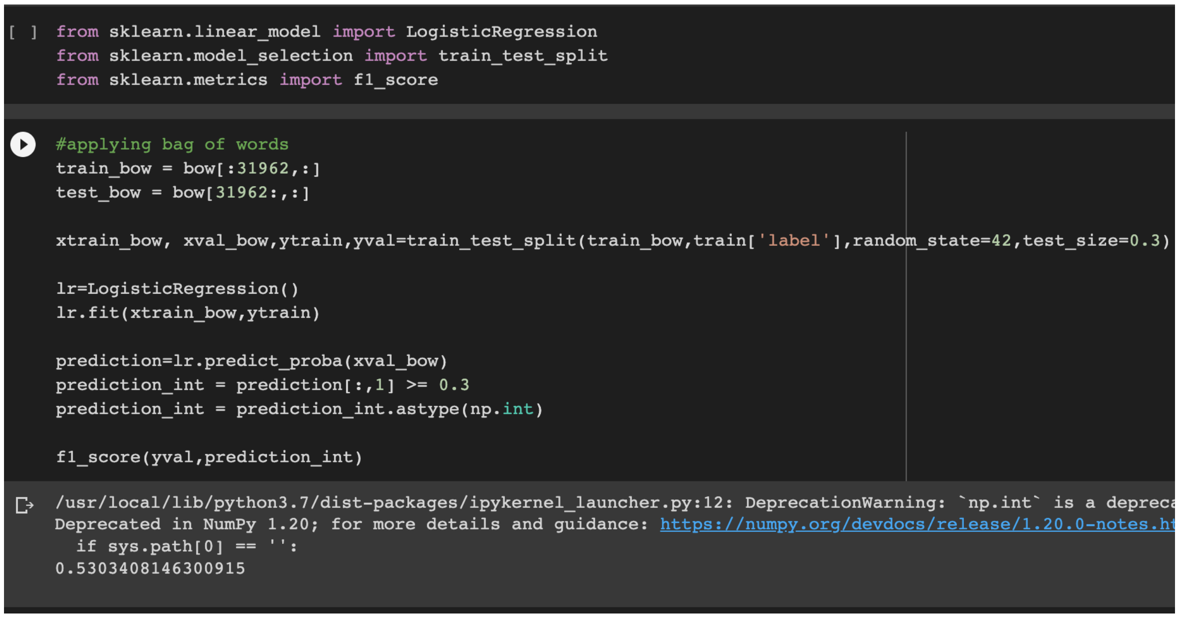
```
WARNING:gensim.models.base_any2vec:Effective 'alpha' higher than previous training cycles
(6510043, 7536020)
```

```
[ ] #Find most similar words in the corpus for a given word

    model_w2v.wv.most_similar(positive="trump")
```

```
[('hillari', 0.5501839518547058),
 ('donald', 0.5449933409690857),
 ('phoni', 0.5358059406280518),
 ('unstabl', 0.532762348651886),
 ('potu', 0.5275628566741943),
 ('jibe', 0.5268758535385132),
 ('tomlin', 0.525926411151886),
 ('#delegaterevolt', 0.525896430015564),
 ('#mainstreammedia', 0.524353563785553),
 ('melo', 0.5210329294204712)]
```

(Fig 28)

The results obtained were satisfactory and upto the mark and the implementation was successful.

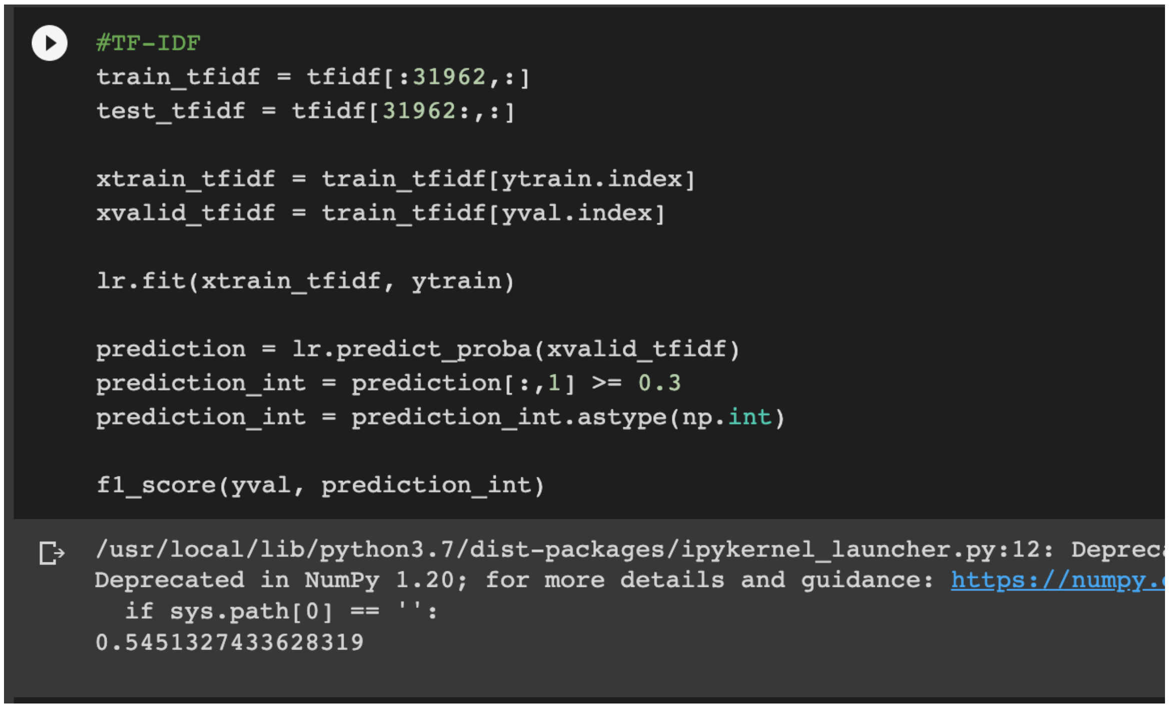


```
[ ] from sklearn.linear_model import LogisticRegression
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import f1_score
```

```
#applying bag of words
train_bow = bow[:31962,:]
test_bow = bow[31962:,:]

xtrain_bow, xval_bow,ytrain,yval=train_test_split(train_bow,train['label'],random_state=42,test_size=0.3)

lr=LogisticRegression()
lr.fit(xtrain_bow,ytrain)

prediction=lr.predict_proba(xval_bow)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

f1_score(yval,prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: DeprecationWarning: `np.int` is a depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.ht
  if sys.path[0] == '':
0.5303408146300915
```

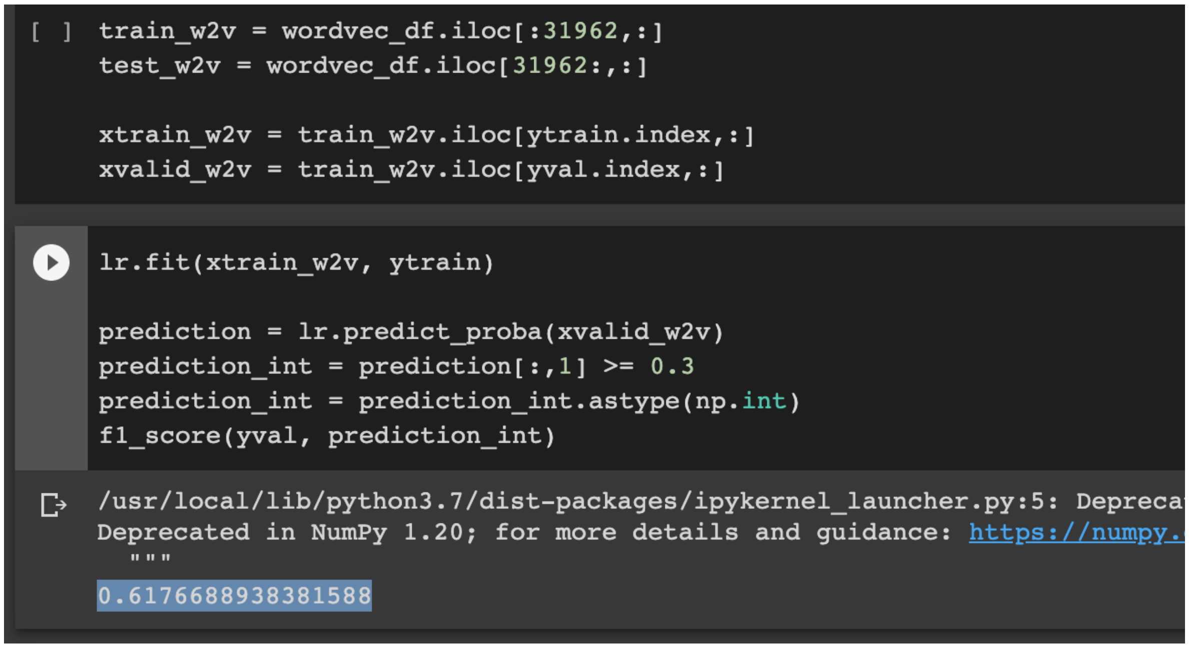
(Fig - 29)



```
#TF-IDF
train_tfidf = tfidf[:31962,:]
test_tfidf = tfidf[31962:,:]

xtrain_tfidf = train_tfidf[ytrain.index]
xvalid_tfidf = train_tfidf[yval.index]

lr.fit(xtrain_tfidf, ytrain)

prediction = lr.predict_proba(xvalid_tfidf)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)

f1_score(yval, prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:12: Depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.
  if sys.path[0] == '':
0.5451327433628319
```

(Fig - 30)



```
[ ]  train_w2v = wordvec_df.iloc[:31962,:]
     test_w2v = wordvec_df.iloc[31962:,:]

     xtrain_w2v = train_w2v.iloc[ytrain.index,:]
     xvalid_w2v = train_w2v.iloc[yval.index,:]
```

```
lr.fit(xtrain_w2v, ytrain)

prediction = lr.predict_proba(xvalid_w2v)
prediction_int = prediction[:,1] >= 0.3
prediction_int = prediction_int.astype(np.int)
f1_score(yval, prediction_int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: Depreca
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.
  """
0.6176688938381588
```

(Fig - 31)

References

A) Conferences and Conference Proceedings

[1] Muhammad Zubair Asghar , Aurangzeb Khan , Shakeel Ahmad , Fazal Masud Kundi, “A Review of Feature Extraction in Sentiment Analysis”, Journal of Basic and Applied Scientific Research, 2014

[2] Ravinder Ahuja, Akash Chugh, Shruti Kohli, Shaurya Gupta, Pratyush Ahuja, “The Impact of Feature Extraction On Sentiment Analysis”, Elsevier , 2019

[3] Henrique Siqueira and Flavia Barros , “A Feature Extraction Process for Sentiment Analysis of Opinions on Services”, Centro de Inform´atica (CIn) - Universidade Federal de Pernambuco (UFPE) Recife - PE - Brazil, ICMC, 2010

[4] M. Avinash, E. Sivasankar, “A study of feature extraction techniques for sentiment analysis”, Springer, Part of the Advances in Intelligent Systems and Computing book series (AISC,volume 814), 2019

[5] Zhu Nanli; Zou Ping; Li Weiguo; Cheng Meng, “Sentiment Analysis :A literature review”, IEEE, 2012

[6] Neena Devasia, Reshma Sheik, “Feature extracted sentiment analysis of customer product reviews”, 2016 International Conference on Emerging Technological Trends (ICETT), IEEE

[7] Vedhavati N. , Anil Kumar K.M. , “E-learning course recommendation based on sentiment analysis using hybrid Elman similarity”, Elsevier, 2022

[8] Caglar G, Ismael A, Cem G, “COVID-19 Related Tweets Classification and Sentiment Analysis Based on Machine Learning Approaches and Deep Learning Architecture Designs: A Comprehensive Analysis”, Research Gate, 2022

B) Datasets

Ashika Trivedi, “Feature Extraction for Sentiment analysis”, https://www.kaggle.com/code/aashkatrivedi/feature-extraction-for-sentiment-analysis/data, (Train.csv, Test. csv), 2020