Department of Computer Science and Engineering,
NMAMIT, Nitte.

Project Phase-II

Evaluation Report

on

Sentimental Analysis of Amazon Reviews

Submitted to

NMAM INSTITUTE OF TECHNOLOGY, NITTE
(An Autonomous Institution under VTU, Belagavi)

In partial fulfillment of the requirements for the award of the

Degree of Bachelor of Engineering
in
Computer Science and Engineering

by

M RamyaPrabhu	**4NM17CS094**
Manasa	**4NM17CS099**
Avinash B	**4NM18CS401**
Meghashree S	**4NM18CS409**

Under the guidance of

Dr. RoshanFernandes
Assoc. Professor
Dept. of CSE, NMAMIT, NITTE

NITTE EDUCATION TRUST | NMAM INSTITUTE OF TECHNOLOGY

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC

CERTIFICATE

Certified that the project work entitled **Sentimental Analysis of Amazon Reviews**is a bonafide work carried out by

M RAMYA PRABHU	**(4NM17CS094)**
MANASA	**(4NM17CS099)**
AVINASH B	**(4NM18CS401)**
MEGHASHREE S	**(4NM18CS409)**

in partial fulfillment for the award of Degree of Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi during the year 2020-21.It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project Phase- 1 prescribed for the said Degree.

Name & Signature of Guide **Signature of HODSignature of the Principal**

Dr. RoshanFernandes
Assoc. Professor
Dept. of CSE, NMAMIT, NITTE

External Viva

Name of the Examiners	**Signature with Date**
1. _____	_____
2. _____	_____

ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without the mention of all the people, without whom this endeavour would have been a difficult one to achieve. Their constant blessings, encouragement, guidance and suggestions have been a constant source of inspiration

First and foremost, my gratitude to my project guide**Dr.RoshanFernandes** for the constant guidance throughout the course of this Project Phase-1 and for the valuable suggestions.

I also take this opportunity to express a deep sense of gratitude to the project coordinators for their valuable guidance and support.

My thanks to our beloved HOD, **Dr. Uday Kumar Reddy,** for extending support in carrying out this project in the department and providing us with all necessary facilities.

My sincere thanks to our beloved principal, **Dr. Niranjan N Chiplunkar**for permitting us to carry out this project at our college and providing us with all needed facilities.

Finally, thanks to staff members of the Department of Computer Science and Engineering and our friends for their honest opinions and suggestions throughout the course of our project Phase-1.

M RamyaPrabhu (4nm17cs094)
Manasa (4nm17cs099)
Avinash B (4nm18cs401)
Meghashree S(4nm18cs409)

ABSTRACT

With the fast growth of e-commerce, large number of products is sold online, and a lot more people are purchasing products online. People while buying also give feedback of product purchased in form of reviews. The user generated reviews for products and services are largely available on internet. Since information available on internet is so widespread we need to extract the needful information for which we make use of sentimental analysis. Sentiment analysis extracts abstract and to the point information required for source materials by applying concept of Natural language processing. It is used to deal with identification and aggregation of the opinions given by the customers. These reviews play vital role in determining potential customer for the products as well as market trend for product. This report provides summary of reviews for products by classifying these reviews as positive, negative or neutral. Information on internet is highly unstructured, machine learning approaches are applied including naïve Bayes and support vector machine algorithms by first taking inputs as unstructured product reviews, performs preprocessing, calculates polarity of reviews, extracts features on to which comments are made and also plots graph for the result. The algorithms precision, recall and accuracy are measured finally.

CHAPTER 1

INTRODUCTION

Online shopping has been growing for 20 years and many e-commerce websites such as Amazon, have been created to meet the increasing demand. Consequently, a specific product can be bought on several websites and the prices may vary. As customers usually want the best quality for the lowest price but can't directly check it, reviews from other customers seem to be the most reliable way to decide whether to buy the product or not. Therefore, sentiment analysis has proven essential to understand a product's popularity among the buyers all over the world

CHAPTER 2

LITERATURE SURVEY

Reference Paper (Name,Author,Year)	Dataset	Techniques/ Algorithm	Advantages	Disadvantages
1)Sentiment analysis of product based reviews - ManveeChauhan, DivakarYadav. (December 2015)	Collected from different sites like consumerreview.com, cnet.com, download.com, zdnet.com. It consists of 13094 product reviews where 12094 are of training and 1000 for testing	To develop an interface Microsoft visual studio is used .It is possible to test and train datasets, extract features out of it. Either naïve bayes or support vector is used to work upon the data and predict polarity of opinions.	Naïve bayes gives better accuracy that is 84.02% .	SVM is less accurate compared to naive bayes that is 80.2%. Large text files take long time for computation.
2)Sentiment Analysis and Sentiment Classification using NLP-G.Divya, R.Suresh (July-2016)	1)The Multi Domain Sentiment dataset:contains product reviews from Amazon.com that includes books,DVDs,Electronics and Kitchen appliances with 1000 positive and 1000 negative reviews for each domain.(http://www.cs.jhu.edu/mdredze/data sets/sentiment) 2)Another review data is available. http://www.cs.uic.edu/	Machine learning techniques like Maximum entropy,Naïve Bayes and Support vector machines for text categorization.Other well known machine learning algorithms in NLP are K-Nearest Neighborhood,ID3,C5,centroid classifier,winnow classifier and Ngram model. Basic idea is estimating probabilities of categories using joint probabilities of words and categories.	Maximum entropy,Naïve Bayes and SVM help achieve great success in text categorization.	Opinion word considered positive in one situation may also be considered negative in another situation. Different people express opinion in different ways. Sentence which lacks context can be difficult to understand.Forex: ”That show was as good as its last show” is dependent entirely on what the person expressing the opinion thought of the previous show.

	liub/FBS/CustomerReviewData.zip: Consists of reviews of 5 electronic products downloaded from Amazon and Cnet. 3)Movie review data is available. (http://www.cs.cornell.edu/People/pabo/movi e-review-data) 4)Blogs,reviewers data collected from ecommerce websites like Amazon,yelp,CNET,dpreview, zdnet			
3)Sentiment Analysis for Amazon.com reviews-LeventGuner,EmilieCoyne,JimSmit (March 2019)	Dataset containing 60,000 product reviews from Amazon.com which are randomly selected from dataset available from Kaggle containing 4 million reviews.	The performance of 3 different algorithms were compared: Multinomial Naïve Bayes(MNB), Linear Support Vector Machine(LSVM), Long short-term memory network(LSTM). For classification with MNB and LSVM,a TF-IDF vectorizer is used.Whereas for classification with LSTM a method called tokenization is applied.	With tokenization numerical data represents whole sentence which is convenient for a recurrent neural network like LSTM.Hence LSTM networks are most suitable for binary sentiment analysis on Amazon.com product reviews.	In this study the assumption that the amount of stars corresponds with the sentiment in the review is made.However it could be the case that a very positive review is given one star or vice versa,polluting the dataset.
4)Twitter Sentiment Analysis Based on Ordinal RegressionShihabElbagir,Jing Yang (October 2019)	Data is collected from Twitter using Twitter API which contains 5000 positive tweets and 5000 negative tweets	Count vectorizer and term frequency-inverse document frequency is used for feature extraction.Machine learning techniques such as Multinomial logistic regression,Suppor	Experimental results conclude that the proposed model can detect ordinal regression with a good accuracy result.	----------

		t Vector Regression,Decision Trees and Random Forest used to build and study machine learning classifier.		
5) Sentiment analysis on online product reviewRaheesasafrin ,K.R. sharmila,T.S.shrisubangi ,E.A. vimal. (April 2017)	The dataset contains online product reviews along with their associated binary sentiment polarity labels comments are taken as review and it is considered as a dataset for our project. The number of entries in the dataset is 3100	Novel incremental diffusive algorithm is being used to extract features from online product descriptions, and then employ association rule mining and the k nearest neighbour	The POS tagging is used to extract the most relevant features to get better results in classifying the sentence as positive or negative. This positive and negative separation of comments is used to analyze the quality of the online products. ANN is used to predict the comments. ANN provides more accuracy than the support vector algorithm	----------
6) Machine Learning-Based Sentiment Analysis for Text MessagesAbhishekBhagat,AkashSharma, Sarat Kr. Chettri (June-2020)	a)The IMDB dataset: a large movie review dataset which consists of 50,000 polarized movie review posts. b) The Sentiment 140 dataset:Consists of 1.6 million Twitter messages. c) The SemEval-2013 dataset:Consists of comments taken from a variety of topics discussed on Twitter. d) The SemEval-2014 dataset:Deals with product reviews made by consumers.	Naïve Bayes, Decision Tree and Support Vector Machine (SVM) for sentiment analysis.	We find that the results obtained from the Decision Tree and SVM have a lower mean square error or higher accuracy with most of the datasets and are considered to be good classifiers.	Drawback of using a machine learning approach to opinion mining is that each opinion is treated as a single uniform statement and assigns a sentiment score to the post as a whole.Also since the social networking data can be accessible in various languages,it becomes an obstacle to sentiment analysis.

7)Sentimental Analysis of Tweets using Naive Bayes Algorithm M. Vadivukarassi, N. Puviarasan, P. Aruna (2017)	2 kinds of APIs to extract the tweets:Search API used for dumping old tweets and Streaming API used for dumping live Tweets.	In this paper, the keywords are collected from Twitter using Twitter API.Theextracted raw data is preprocessed using Natural Language Toolkit techniques. Algorithms usedare Naïve Bayes& Chi-Square test.	The proposed system would be easy for user to obtain the summarized report about the opinion from Twitter. It also supports them in the decision making process in their daily life activities.	Application settings that should always be kept private.
8)Natural Language Processing for Sentiment Analysis- Wei Yen Chong, BhawaniSelvaretnam , Lay-Ki Soon (2014)	A total of 1513 tweets were extracted from Twitter and manually labelled.	SVM,Decision Tree ,Naïve bayes. The processed tweets will proceed to sentiment classification, in order to predict the sentiment of tweets. a)Subjectivity Classification b)Semantic Association c)Polarity classification.	The results are tabulated in confusion matrix. It records the predicted result and the actual result.	It is also noticeable that due to the highly appeared misspelled words and slangs in tweets, it is not easy to extract the sentiment lexicons if it is not preprocessed to formal language.
9) Analysis of Feature Selection Methods for Text Classification using Multiple Datasets ArchitAggarwal, BhavyaGola,TusharSankla (June 2020)	a)20Newgroups Dataset b)Polarity Dataset c)Reuters21578 Dataset	Naive Bayes, Bagging, Random Forest and Naive Bayes Multinomial classifiers for text classification on various datasets.	It usually gives more often than not, better or equally good results without using any feature selection as compared to using feature selectors taking into account all the evaluation measures.	.....
10) Sentiment analysis on hotel reviews using Multinomial Naïve Bayes classifier - Arif Abdurrahman Farisi, YuliantSibaroni ,Said Al Faraby. (2019)	In this research the dataset is derived from Data finitis's Business Database which contains hotel reviews of as many as 5000 English sentences in CSV File.	Multi-Nomial Naïve Bayes Model K-Fold Cross Validation	The use of preprocessing in figure 1 greatly affects the performance of the system so it can be seen if using preprocessing F1-Score average results can improve performance optimally.	By doing a test scenario will result in different performance because each scenario can affect the model built. Each scenario is validated using 10 fold cross validation each time

CHAPTER 3

PROBLEM DEFINITION

The aim of this project is to make an application in the field of natural language processing in order to find and implement a novel algorithm to solve the problem of measuring real-time comments made by users on products. For this we have collected the reviews from amazon that have been written by different users about a particular product and then the polarity of the each feature of the mentioned product is determined that either it is negative, positive and neutral. For determining the polarity of the feature of product, the sentiment of feature word (noun) of the text is calculated and corresponding scores are given to public opinions of the product's feature which could being used to compare between the product based on a particular feature and a statistical output was produced to show the results. The methods used in this project can be used for any specific product with public opinions. The customer's reviews reflect the customer's sentiments and have a substantial significance for the products being sold online including electronic gadgets, movies, house hold appliances and books. Hence, extracting the exact features of the products by analyzing the text of reviews requires a lot of efforts and human intelligence.

CHAPTER 4

SYSTEM REQUIREMENTS SPECIFICATION

Software Requirements:

- Operating system: Windows 7 and above
- Anaconda Navigator(anaconda3.x)
- Tool used : Jupyter Notebook
- .Python Libraries such as Numpy,Matplotlib,pandas,condas,nltk

Hardware Requirements:

- Processor – i3 and above
- Hard Disk – 500 GB
- Memory – 4 GB RAM

CHAPTER 5

SYSTEM DESIGN

Sentiment analysis is a type of natural language processing for tracking the sentiments of the public about a particular product or topic. Sentiment analysis, which is also called opinion mining, involves in building a system to collect and examine opinions about the product made in blog posts, comments, reviews or tweets.

Method:

1)Data acquisition

The dataset used for training consisted of a big dataset (4 million reviews) available on Kaggle. This Kaggle dataset consists of Amazon customer reviews (input text)

Categories	Number of reviews
Data science book	20648
Electronic products	35633
Musical Instruments	10259
Fine foods	10000

Data specifications

2)Importing required Libraries

```
In [1]: import pandas as pd
        import numpy as np
        import nltk
```

3)Load the Dataset

```
In [ ]: df = pd.read_csv("FINAL_DATASET.csv")
```

4)Validating the Dataset

Reading First 5 Records

```
In [5]: df.head()
```

Out[5]:

	AMAZON_TEXT_REVIEWS	CATAGORIES
0	I order 3 of them and one of the item is bad q...	Amazon Electronics Products
1	Bulk is always the less expensive way to go fo...	Amazon Electronics Products
2	Well they are not Duracell but for the price i...	Amazon Electronics Products
3	Seem to work as well as name brand batteries a...	Amazon Electronics Products
4	These batteries are very long lasting the pric...	Amazon Electronics Products

Reading Last 5 Records

```
In [7]: df.tail()
```

Out[7]:

	AMAZON_TEXT_REVIEWS	CATAGORIES
76534	Am disabled, retired RN, but always wished I w...	Data Science Book
76535	At one point I considered library work as oppo...	Data Science Book
76536	Overall, I think this is an excellent resource...	Data Science Book

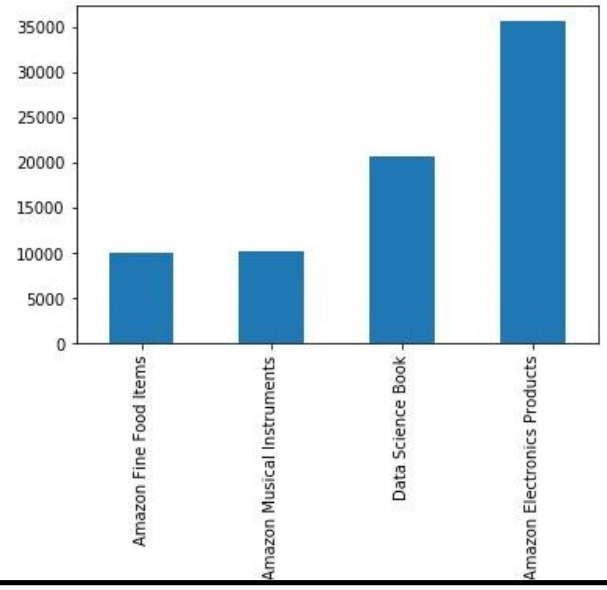
a)Check for NULL values

```
In [10]: print(df.isnull().sum())

AMAZON_TEXT_REVIEWS    0
CATAGORIES             0
dtype: int64
```

Since There is No Null Values Found in Our Dataset

b)Representing Dataset through Bar graph

```
In [12]: # plotting Bar graph for CATAGORIES
         df["CATAGORIES"].value_counts().sort_values().plot.bar()

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x1cd05dc0048>
```



5) Data pre-processing

Raw reviews are full of noise, misspellings, and contain numerous abbreviations and slang words. Such noisy characteristics often influence the performance of sentiment analysis approaches. Thus, some preprocessing approaches are applied prior to feature extraction.The preprocessing of reviews includes the following steps:

Step 5: Data Pre-Processing

DATA NORMALIZATION

1. Converting Emojis to their Respective Emotions
2. Removing all Punctuations in the Reviews
3. Removing StopWords
4. Tokenize the Data
5. Stemming
6. Lemmatization
7. Bag of Words(BOW)

1. Converting Emojis to their Respective Emotions

```
In [36]: df["AMAZON_TEXT_REVIEWS"] = df["AMAZON_TEXT_REVIEWS"].replace(["\:\)", "\:\-\)", "\:\-\}", "\;\-\}", "\:\-\>", "\;\-\)"], ["Happy
         df["AMAZON_TEXT_REVIEWS"] = df["AMAZON_TEXT_REVIEWS"].replace(["\:\-\(", "\:\(", "\:\-\|", "\;\-\(", "\;\-\<", "\|\-\{"], ["Sad",
         df["AMAZON_TEXT_REVIEWS"] = df["AMAZON_TEXT_REVIEWS"].replace(["\:\D", "\:\`\-\)", "\:\`\-\("], ["laugh", "tear of joy", "tear of
```

a)Converting Emojis to their Respective Emotions in the reviews.

The use of emoji on the internet rapidly increased in recent years.People often use them when it is difficult to describe their expressions only with

words.A single Emoji character may enhance the expressivity of a text message. A name of a city has no sentiment value when it is posted alone. However, if the user used an Emoji along with this name, the text may have a sentiment value. For example, a smiling face Emoji character can express someone's positive feeling towards the city. In contrast, using the angry face Emoji along with some brand name may reveal negative feelings towards the brand.In this model we convert Emojis to their respective emotions.

1. Converting Emojis to their Respective Emotions

```
In [36]: df["AMAZON_TEXT_REVIEWS"] = df["AMAZON_TEXT_REVIEWS"].replace([":)", ":-)", ":-}", ";-}", ":->", ";-)"], ["Happy
         df["AMAZON_TEXT_REVIEWS"] = df["AMAZON_TEXT_REVIEWS"].replace([":-(", ":(", ":-|", ";-(", ";-<", "|-{"], ["Sad",
         df["AMAZON_TEXT_REVIEWS"] = df["AMAZON_TEXT_REVIEWS"].replace([":D", ":'-)", ":`-("], ["laugh", "tear of joy", "tear of
```

b)Removing all Punctuations in the Reviews:

2. Removing all punctuations in the Reviews

```
In [14]: import string
         exclude = set(string.punctuation)
         final = exclude

         print(len(exclude))
         print(len(final))
```

```
32
32
```

```
In [15]: print(final)
```

```
{'~', '!', '\\', '^', '_', '+', '#', '<', ',', '@', '"', '&', '/', ':', '(', ']', "'", ')', ';', '>', '|', '*', '[', '$', '{',
'?', '`', '%', '}', '-', '.', '='}
```

```
In [16]: for i in final:
           df['AMAZON_TEXT_REVIEWS'] = df['AMAZON_TEXT_REVIEWS'].str.replace(i, ' ', regex=True)
```

```
In [20]: #Output after Removing Punctuation
         df["AMAZON_TEXT_REVIEWS"].tolist()
```

```
 'Seem to hold up as well as any name brand and at a much better price',
 'Works just as well as every other brand of battery with a better price  Powers all of my sons toys for months at a time ',
 'Just got them so i really wont kno for a few eeeks or so how good these batteries are',
 'These last as well as Energizer  half the price ',
 'These batteries are horrible  We depend on AA batteries for many wireless microphone and have found that brand name batteri
es last about a month per our usage  We switched to Amazon basics to try it out when a subscription for our preferred brand w
as no longer available and have found that  under the same usage  these batteries last about a week  That s a 1 4 capacity ',
 '3rd purchase of these AA from Amazon  all have expiration of 2027  Ones I ve used thus far last as Long as name brands',
 'I am 100 satisfied with my purchase of these batteries',
 'I put these batteries in 10 of the 12 window candles that I bought  I put my last 2 Duracell in the other 2  The Amazon Ba
tteries lasted for about 10 days and the Duracell lasted for 12  Since then I ve experimented to see if this is consistent
It is  the Duracell last an average of 17 longer ',
 'First time I bought these they worked well and lasted almost as long as the name brands  Not the same this time around  alm
ost no charge  4 6 hours of run time  Junk batteries ',
```

c)RemovingStopwords:

Stopwords are often added to sentences to make them grammatically correct, for example, words such as a, is, an, the, and etc.These should be removed so machine learning algorithms can better focus on words which define the meaning/idea of the text

```
In [21]: import nltk
         nltk.download('stopwords')

         [nltk_data] Downloading package stopwords to
         [nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
         [nltk_data]   Package stopwords is already up-to-date!

Out[21]: True

In [22]: !pip install stop_words

         Requirement already satisfied: stop_words in c:\users\dell\anaconda3.x\lib\site-packages (2018.7.23)

In [23]: from stop_words import get_stop_words
         from nltk.corpus import stopwords

         stop_words = list(get_stop_words('en'))         #About 900 stopwords
         nltk_words = list(stopwords.words('english')) #About 179 stopwords
         stop_words.extend(nltk_words)

         len(stop_words)

Out[23]: 353
```

```
In [24]: print(stop_words)

['a', 'about', 'above', 'after', 'again', 'against', 'all', 'am', 'an', 'and', 'any', 'are', "aren't", 'as', 'at', 'be', 'becau
se', 'been', 'before', 'being', 'below', 'between', 'both', 'but', 'by', "can't", 'cannot', 'could', "couldn't", 'did', "did
n't", 'do', 'does', "doesn't", 'doing', "don't", 'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had', "hadn't", 'h
as', "hasn't", 'have', "haven't", 'having', 'he', "he'd", "he'll", "he's", 'her', 'here', "here's", 'hers', 'herself', 'him',
'himself', 'his', 'how', "how's", 'i', "i'd", "i'll", "i'm", "i've", 'if', 'in', 'into', 'is', "isn't", 'it', "it's", 'its', 'i
tself', "let's", 'me', 'more', 'most', "mustn't", 'my', 'myself', 'no', 'nor', 'not', 'of', 'off', 'on', 'once', 'only', 'or',
'other', 'ought', 'our', 'ours', 'ourselves', 'out', 'over', 'own', 'same', "shan't", 'she', "she'd", "she'll", "she's", 'shoul
d', "shouldn't", 'so', 'some', 'such', 'than', 'that', "that's", 'the', 'their', 'theirs', 'them', 'themselves', 'then', 'ther
e', "there's", 'these', 'they', "they'd", "they'll", "they're", "they've", 'this', 'those', 'through', 'to', 'too', 'under', 'u
ntil', 'up', 'very', 'was', "wasn't", 'we', "we'd", "we'll", "we're", "we've", 'were', "weren't", 'what', "what's", 'when', "wh
en's", 'where', "where's", 'which', 'while', 'who', "who's", 'whom', 'why', "why's", 'with', "won't", 'would', "wouldn't", 'yo
u', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'yourself', 'yourselves', 'i', 'me', 'my', 'myself', 'we', 'our',
'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'hi
s', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'the
mselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be',
'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'bec
ause', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'be
fore', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 't
hen', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'so
me', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "d
on't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'did
n', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "migh
tn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
'won', "won't", 'wouldn', "wouldn't"]
```

In [25]:
```
#Before Removing Stopwords
print(df)
```

```
                                      AMAZON_TEXT_REVIEWS  \
0      I order 3 of them and one of the item is bad q...
1      Bulk is always the less expensive way to go fo...
2      Well they are not Duracell but for the price i...
3      Seem to work as well as name brand batteries a...
4      These batteries are very long lasting the pric...
...                                                  ...
76534  Am disabled  retired RN  but always wished I w...
76535  At one point I considered library work as oppo...
76536  Overall  I think this is an excellent resource...
76537                                              GREAT
76538                                  Execellent info

                        CATAGORIES
0      Amazon Electronics Products
1      Amazon Electronics Products
2      Amazon Electronics Products
3      Amazon Electronics Products
4      Amazon Electronics Products
...                            ...
76534           Data Science Book
76535           Data Science Book
76536           Data Science Book
76537           Data Science Book
76538           Data Science Book

[76539 rows x 2 columns]
```

In [33]:
```
#Data Cleaning
from nltk.corpus import stopwords
stop = set(stopwords.words('english'))
from nltk.corpus import stopwords
def remove_stopword(word):
    return word not in words

#StopWords Removed
df['StopWReviews'] = df['AMAZON_TEXT_REVIEWS'].str.lower().str.split()
df['StopWReviews'] = df['StopWReviews'].apply(lambda x : [item for item in x if item not in stop])
```

```
                         CATAGORIES  \
0      Amazon Electronics Products
1      Amazon Electronics Products
2      Amazon Electronics Products
3      Amazon Electronics Products
4      Amazon Electronics Products
...                             ...
76534            Data Science Book
76535            Data Science Book
76536            Data Science Book
76537            Data Science Book
76538            Data Science Book

                                                 StopWReviews
0      [order, 3, one, item, bad, quality, missing, b...
1      [bulk, always, less, expensive, way, go, produ...
2                        [well, duracell, price, happy]
3      [seem, work, well, name, brand, batteries, muc...
4                [batteries, long, lasting, price, great]
...                                                    ...
76534  [disabled, retired, rn, always, wished, librar...
76535  [one, point, considered, library, work, oppose...
76536  [overall, think, excellent, resource, anyone, ...
76537                                            [great]
76538                                 [execellent, info]

[76539 rows x 3 columns]
```

Output after removing stopwords

d)Tokenize the data

Tokenisation is the process of breaking up a given text into units called tokens. Tokens can be individual words, phrases or even whole sentences.These tokens are very useful for finding such patterns as well as is considered as a base step for stemming and lemmatization. Stemming and Lemmatization both generate the root form of the inflected words obtained from tokenisation.

4. Tokenize the Data

In [37]:
```
import nltk
nltk.download('punkt')

df['AMAZON_TEXT_REVIEWS'] = df.apply(lambda row: nltk.word_tokenize(row['AMAZON_TEXT_REVIEWS']), axis=1)
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\DELL\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [38]:
```
df.AMAZON_TEXT_REVIEWS.tolist()
```

Out[38]:
```
[['I',
  'order',
  '3',
  'of',
  'them',
  'and',
  'one',
  'of',
  'the',
  'item',
  'is',
  'bad',
  'quality',
  'Is',
  'missing',
  'backup',
```

e)Stemming

Stemming is a method of removing the suffix of the word and bringing it to a base word. Stemming is the normalization technique used in Natural language processing that reduces the number of computations required. We can do stemming in NLP using libraries such as PorterStemming, Snowball Stemmer, etc.For example, we have the word ‘Eating, we remove the suffix ‘ing’ and bring that word to a base word ‘eat’.

Stemming is mainly used to reduce the dimensionality of data. In simple words, if there we have words like walk, walks, waited, waiting that are different but similar contextually. We bring these words base word ‘walk’ by removing suffixes from all the words.

In [66]:
```
!pip install corpus
```

```
Requirement already satisfied: corpus in c:\users\dell\anaconda3.x\lib\site-packages (0.4.2)
```

In [80]:
```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
corpus = []
for i in range(0, 76539):
    review=df['AMAZON_TEXT_REVIEWS'][i]
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)
# stemming for whole set AMAZON_TEXT_REVIEWS
```

```
In [82]: print(corpus)

['I order 3 one item bad qualiti Is miss backup spring I put pc aluminum make batteri work', 'bulk alway less expens way go pro
duct like', 'well duracel price happi', 'seem work well name brand batteri much better price', 'these batteri long last price g
reat', 'bought lot batteri christma amazonbas cell good I notic differ brand name batteri amazon basic brand just lot easier pu
rchas arriv hous hand will buy', 'ive problam batteri order past pleas', 'well look cheap non recharg batteri last quit perfect
noth say', 'these hold amount high power juic like energ duracel half price', 'amazonbas AA aaa batteri done well appear good s
helf life I buy', 'I find amazon basic batteri equal superior name brand one can believ I start buy sooner the packag larg pric
e great', 'when I first start get amazon basic batteri I realli like with recent purchas seem last like mayb mix bag inconsist
last better other I done test feel brand may last longer howev price hard beat', 'use fish tank light night work great I love e
asili switch want guest', 'got em I realli comment good job good price quick deliveri put two one keyboard go year say three da
y', 'mani thing need aa batteri great', 'thank I abl find amazon great price even better ship arriv perfect condit exactli I ne
ed great purchas would purchas', 'I know I would buy thu brand seem like last long duracel', 'In opinion last anywher near long
duracel thing like led candl crazi trail camera camera expos cold temp less batteri WE buy bulk north hous amazon basic great t
hing like sheet beed towel In opinion batteri life larg packag aaa aa size purchas lack', 'they last long brand name good enoug
h consid much cheaper', 'bought batteri christma gift month decemb last like 2month toy need replac batteri I also use doorbel
need replac batteri Tv remot control still work batteri last long', 'thi second order seem work good name brand ship door', 'th
i second purchas amazon batteri work great just good even better name brand batteri half price thi way I purchas batteri', 'the
y last long duracel batteri xbox one control none explod like review said', 'seem last long name brand name disappoint', 'these
last long cheap batteri happi', 'use yet batteri batteri good price', 'these amazon batteri job although I gave 4star I I would
say hand full batteri strong pretti weak box 48 batteri I definit buy priceim pretti well satisfi thank', 'We order time caus h
appi product these good qualiti We also happi price I would recommend', 'have not had chanc TO use all OF them but hope all OF
them will remain IN good order for A reason amount OF time mean A few year out', 'light thought fit light arriv nice compani ba
tteri need aa ok fault know thank', 'I use batteri game camera work great last sever week', 'veri conveni box batteri good pric
e No longer need compar price store should done long time ago', 'not much say batteri except work good price have hundr let sha
re kid grand kid', 'they seem work okay far price great', 'I use xbox control last pretti long great cheap price', 'seem hold w
ell name brand much better price', 'work well everi brand batteri better price power son toy month time', 'just got realli wont
kno eeek good batteri', 'these last well energ half price', 'these batteri horribl We depend AA batteri mani wireless microphon
found brand name batteri last month per usag We switch amazon basic tri subscript prefer brand longer avail found usag batteri
```

```
In [101]: from nltk.corpus import stopwords
          from nltk.stem.porter import PorterStemmer
          from nltk.tokenize import word_tokenize

          ps=PorterStemmer()
          for line in df.AMAZON_TEXT_REVIEWS:
               for w in word_tokenize(line):
                  if w not in stopwords.words('english'):
                       print(w, " : " , ps.stem(w))
           # output for particular word shows before steminng and after stemmin in  AMAZON_TEXT_REVIEWS
```

```
I  :  I
order  :  order
3  :  3
one  :  one
item  :  item
bad  :  bad
quality  :  qualiti
.  :  .
Is  :  Is
missing  :  miss
backup  :  backup
spring  :  spring
I  :  I
put  :  put
pcs  :  pc
aluminum  :  aluminum
make  :  make
```

As there is no accuracy in stemming,we will not get meaningful word ,it only cuts off the suffix.So we are doing Lemmatization to overcome this issue.

f)Lemmatization

Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.
For example, runs, running, ran are all forms of the word run, therefore run is the lemma of all these words. Because lemmatization returns an actual word of the language, it is used where it is necessary to get valid words.

g)Bag of Words

Bag of Words (BOW) is a method to extract features from text documents. These features can be used for training machine learning algorithms. It creates a vocabulary of all the unique words occurring in all the documents in the training set

```
In [114]: print(words)

[['I', 'order', '3', 'of', 'them', 'and', 'one', 'of', 'the', 'item', 'is', 'bad', 'quality', 'Is', 'missing', 'backup', 'sprin
g', 'so', 'I', 'have', 'to', 'put', 'a', 'pcs', 'of', 'aluminum', 'to', 'make', 'the', 'battery', 'work'], ['Bulk', 'is', 'alwa
ys', 'the', 'less', 'expensive', 'way', 'to', 'go', 'for', 'products', 'like', 'these'], ['Well', 'they', 'are', 'not', 'Durace
ll', 'but', 'for', 'the', 'price', 'i', 'am', 'happy'], ['Seem', 'to', 'work', 'as', 'well', 'as', 'name', 'brand', 'batterie
s', 'at', 'a', 'much', 'better', 'price'], ['These', 'batteries', 'are', 'very', 'long', 'lasting', 'the', 'price', 'is', 'grea
t'], ['Bought', 'a', 'lot', 'of', 'batteries', 'for', 'Christmas', 'and', 'the', 'AmazonBasics', 'Cell', 'have', 'been', 'goo
d', 'I', 'haven', 't', 'noticed', 'a', 'difference', 'between', 'the', 'brand', 'name', 'batteries', 'and', 'the', 'Amazon', 'B
asic', 'brand', 'Just', 'a', 'lot', 'easier', 'to', 'purchase', 'and', 'have', 'arrive', 'at', 'the', 'house', 'and', 'have',
'on', 'hand', 'Will', 'buy', 'again'], ['ive', 'not', 'had', 'any', 'problame', 'with', 'these', 'batteries', 'have', 'ordere
d', 'them', 'in', 'the', 'past', 'been', 'very', 'pleased'], ['Well', 'if', 'you', 'are', 'looking', 'for', 'cheap', 'non', 're
chargeable', 'batteries', 'that', 'last', 'quite', 'a', 'while', 'then', 'these', 'are', 'perfect', 'Nothing', 'more', 'to', 's
ay'], ['These', 'do', 'not', 'hold', 'the', 'amount', 'of', 'high', 'power', 'juice', 'like', 'energizer', 'or', 'duracell', 'b
ut', 'they', 'are', 'half', 'the', 'price'], ['AmazonBasics', 'AA', 'AAA', 'batteries', 'have', 'done', 'well', 'by', 'me', 'ap
pear', 'to', 'have', 'a', 'good', 'shelf', 'life', 'I', 'll', 'buy', 'them', 'again'], ['I', 'find', 'amazon', 'basics', 'batte
ries', 'to', 'be', 'equal', 'if', 'not', 'superior', 'to', 'name', 'brand', 'ones', 'Can', 't', 'believe', 'I', 'didn', 't', 's
tart', 'buying', 'them', 'sooner', 'The', 'packages', 'are', 'large', 'and', 'the', 'price', 'is', 'great', 'too'], ['When',
'I', 'first', 'started', 'getting', 'the', 'Amazon', 'basic', 'batteries', 'I', 'really', 'liked', 'them', 'With', 'recent', 'p
urchases', 'they', 'do', 'not', 'seem', 'to', 'last', 'like', 'they', 'had', 'or', 'maybe', 'a', 'mixed', 'bag', 'inconsisten
t', 'with', 'some', 'lasting', 'better', 'than', 'others', 'I', 'have', 'not', 'done', 'any', 'tests', 'but', 'feel', 'some',
'other', 'brands', 'may', 'last', 'longer', 'However', 'the', 'price', 'is', 'hard', 'to', 'beat'], ['Use', 'it', 'for', 'my',
'fish', 'tank', 's', 'light', 'at', 'night', 'and', 'works', 'great', 'I', 'love', 'how', 'you', 'can', 'easily', 'switch', 'i
t', 'off', 'and', 'on', 'if', 'you', 'want', 'it', 'on', 'while', 'guests', 'are', 'there'], ['just', 'got', 'em', 'so', 'I',
'can', 't', 'really', 'comment', 'on', 'how', 'good', 'the', 'do', 'the', 'job', 'good', 'price', 'quick', 'delivery', 'but',
'have', 'only', 'put', 'two', 'into', 'one', 'of', 'my', 'keyboards', 'but', 'they', 'can', 'go', 'up', 'to', 'a', 'year', 's
```

After lemmatization our dataset looks like this, we have added new column 'StopWreviews' which is Normalized column (cleaned data)

We are going to apply different algorithms to this column in our next process

PROJECT PHASE -II

IMPLEMENTATION OF ALGORITHM

We have implemented 5 Algorithms and tested the accuracy for each by splitting the dataset into test and train

Process after Lemmatization

- The 'TextBlob' Library is used to get the subjectivity and polarity of each review.It gives some positive values and some negative values for each review.
- To achieve this we should load Normalize the column 'StopWreviews'
- Renaming 'StopWreviews' column into 'CleanedText' .
  →df['CleanedText']=df['StopWReviews']
  Output of the same

```
In [20]: df.head()
```

Out[20]:

	AMAZON_TEXT_REVIEWS	CATAGORIES	StopWReviews	CleanedText
0	I order 3 of them and one of the item is bad q...	Amazon Electronics Products	order one item bad quality miss backup spring ...	order one item bad quality miss backup spring ...
1	Bulk is always the less expensive way to go fo...	Amazon Electronics Products	bulk always less expensive way go product like	bulk always less expensive way go product like
2	Well they are not Duracell but for the price i...	Amazon Electronics Products	well duracell price happy	well duracell price happy
3	Seem to work as well as name brand batteries a...	Amazon Electronics Products	seem work well name brand battery much good price	seem work well name brand battery much good price
4	These batteries are very long lasting the pric...	Amazon Electronics Products	battery long lasting price great	battery long lasting price great

- The 'CleanedText' is the column containing normalized data and it is in the text format.
- On loading this column to the TextBlob,we get Polarity and Subjectivity as shown below:

Out[8]:

	AMAZON_TEXT_REVIEWS	CATAGORIES	StopWReviews	CleanedText	tb_Pol	tb_Subj
0	I order 3 of them and one of the item is bad q...	Amazon Electronics Products	order one item bad quality miss backup spring ...	order one item bad quality miss backup spring ...	-0.700000	0.666667
1	Bulk is always the less expensive way to go fo...	Amazon Electronics Products	bulk always less expensive way go product like	bulk always less expensive way go product like	-0.333333	0.383333
2	Well they are not Duracell but for the price i...	Amazon Electronics Products	well duracell price happy	well duracell price happy	0.800000	1.000000

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion.

- It is available in the NLTK package and can be applied directly to unlabeled text data.

- VADER sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores.
- The sentiment score of a text can be obtained by summing up the intensity of each word in the text.
- We check only if the text expresses a positive, negative or neutral opinion.
- We need not check for subjective fact or opinion
- VADER's SentimentIntensityAnalyzer() takes in a string as input and returns a dictionary of scores in each of four categories:
  1) negative
  2) neutral
  3) positive
  4) compound
- We have applied VaderSentimentAnalyzer() to get labels like neg,pos,neu since all the algorithm needs labeled dataset for further implementation.
- Our dataset contains 2 columns i,e AMAZON_TEXT_REVIEWS(text reviews) and CATEGORIES(Types of Reviews combined)
- We check if the values we got for 'VaderSentimentAnalyzer' after loading the normalized column is same as we got for 'TextBlob' or not.
- It is successful that we got same positive and negatives values for each rows on implementing the libraries .Hence comparison done successfully.
- Below is the screenshot for the same.

Out[10]:

	AMAZON_TEXT_REVIEWS	CATAGORIES	StopWReviews	CleanedText	tb_Pol	tb_Subj	compound	neg	neu	pos
0	I order 3 of them and one of the item is bad q...	Amazon Electronics Products	order one item bad quality miss backup spring ...	order one item bad quality miss backup spring ...	-0.700000	0.666667	-0.6249	0.298	0.702	0.000
1	Bulk is always the less expensive way to go fo...	Amazon Electronics Products	bulk always less expensive way go product like	bulk always less expensive way go product like	-0.333333	0.383333	0.3612	0.000	0.737	0.263
2	Well they are not Duracell but for the price i...	Amazon Electronics Products	well duracell price happy	well duracell price happy	0.800000	1.000000	0.7003	0.000	0.256	0.744

- We have defined another 2 columns for better understanding.

1)Sentiment -1 means negative review and 1 means positive review.

2)Label 'pos' means positive review and 'neg' means negative review.

- We have appended the above 2 columns named sentiment and Label,which gives a clear picture of the analysis for checking the customer reviews ‘positive or negative’ .
- In this step we have achieved our task of checking the ‘sentiment analysis of amazon reviews’ i.e positive or negative.Below is the screenshot for the same.

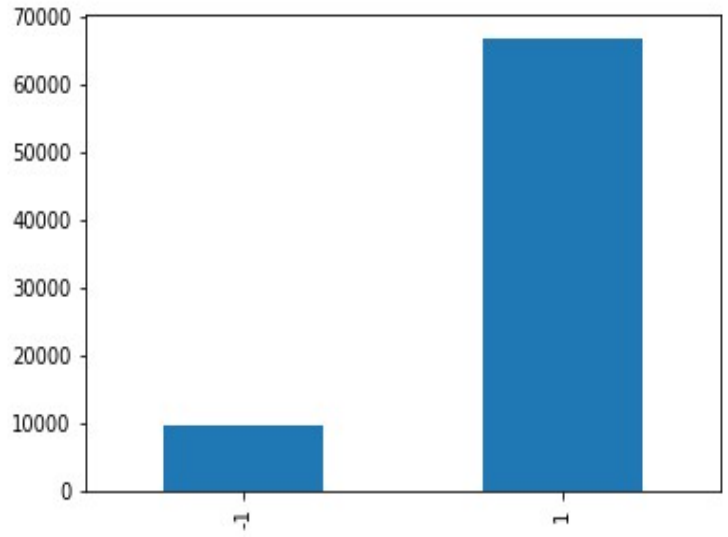
In [11]:
```
df.loc[df['neg'] < df['pos'], 'sentiment'] = '1'
df.loc[df['neg'] >= df['pos'], 'sentiment'] = '-1'
df.loc[df['neg'] < df['pos'], 'Label'] = 'pos'
df.loc[df['neg'] >= df['pos'], 'Label'] = 'neg'
df.head(100)
```

Out[11]:

	AMAZON_TEXT_REVIEWS	CATAGORIES	StopWReviews	CleanedText	tb_Pol	tb_Subj	compound	neg	neu	pos	sentiment	Label
0	I order 3 of them and one of the item is bad q...	Amazon Electronics Products	order one item bad quality miss backup spring ...	order one item bad quality miss backup spring ...	-0.700000	0.666667	-0.6249	0.298	0.702	0.000	-1	neg
1	Bulk is always the less expensive way to go fo...	Amazon Electronics Products	bulk always less expensive way go product like	bulk always less expensive way go product like	-0.333333	0.383333	0.3612	0.000	0.737	0.263	1	pos
2	Well they are not Duracell but for the price i...	Amazon Electronics Products	well duracell price happy	well duracell price happy	0.800000	1.000000	0.7003	0.000	0.256	0.744	1	pos
3	Seem to work as well as name brand batteries a...	Amazon Electronics Products	seem work well name brand battery much good price	seem work well name brand battery much good price	0.700000	0.600000	0.6124	0.000	0.583	0.417	1	pos
4	These batteries are very long lasting the pric...	Amazon Electronics Products	battery long lasting price great	battery long lasting price great	0.250000	0.383333	0.6249	0.000	0.494	0.506	1	pos
...	...	...	...	...	...	...	...	...	...	...	...	...

Graph representation of total number of positive and negative reviews,we have taken the sentiment column to plot

```
In [37]: df["sentiment"].value_counts().sort_values().plot.bar()
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x29532abb108>
```



- We need only 2 columns to work with all the algorithms,to get accuracy of each.
- For this we have taken ‘CleanedText’ column data.
- ‘Sentiment’ column contains -1 and 1 scores that represents positive and negative reviews.
- For better understanding we have re-named those 2 columns into ‘reviews’ and ‘score’

```
In [42]: newdf = df[["CleanedText", "sentiment"]]
         newdf.columns = ["reviews", "score"]
         newdf.head(100)
Out[42]:
```

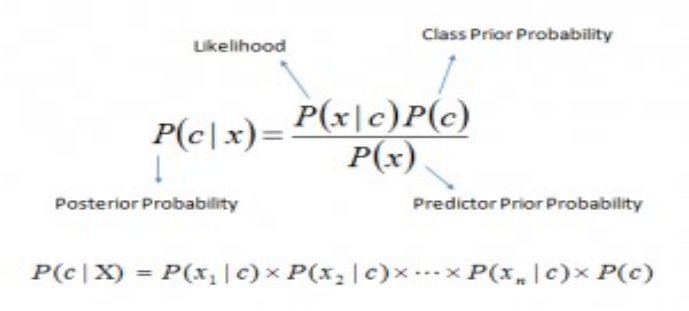
	reviews	score
0	order 3 one item bad quality missing backup sp...	-1
1	bulk always less expensive way go products like	1
2	well duracell price happy	1
3	seem work well name brand batteries much bette...	1
4	batteries long lasting price great	1
...	...	...
95	buy amazonbasics comes batteries long lasting ...	1
96	leappads xbox controllers remotes use batterie...	1
97	buy batteries great long lasting batteries	1
98	know crazy cheap batteries light practically h...	1
99	ps3 remote control television remote controls ...	1

100 rows × 2 columns

1. Naive Bayes Implementation

Naive Bayes is a supervised learning algorithm that's typically used for classification problems. Naive Bayes is simple, intuitive, and yet performs well

- It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors.
- They are probabilistic, which means that they calculate the probability of each tag for a given text, and then output the tag with the highest one.
- The way they get these probabilities is by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.
- In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.It predicts the tag of text.
- Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



,

P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).

P(c) is the prior probability of class.

P(x|c) is the likelihood which is the probability of predictor given class.

P(x) is the prior probability of predictor.

Process of implementation

- Splitting the dataset into Train and Test By using sklearn.model_selection On importing train_test_split.
- train_test_split is a function in Sklearn model selection for splitting data arrays into two subsets: for training data and for testing data. With this function, we don't need to divide the dataset manually. By default, Sklearn train_test_split will make random partitions for the two subsets.
- Given two sequences, like x and y here, train_test_split() performs the split and returns four sequences (in this case NumPy arrays) in this order:

  1) x_train: The training part of the first sequence (x)
  2) x_test: The test part of the first sequence (x)
  3) y_train: The training part of the second sequence (y)
  4) y_test: The test part of the second sequence (y)

- By specifying the train_size as 0.75, we aim to put 75% of the data into our training set, and test_size as 0.25, we aim to put 25% of the data into our testing set. Because we only have 10 data points, the program automatically rounded the ratio to 7:3. It's okay to omit the test_size parameter, if we already got the train_size specified.

  Why Random State is used?

  random_state is basically used for reproducing our problem the same every time it is run. If we do not use a random_state in train_test_split, every time we make the split we might get a different set of train and test data points and will not help us in debugging, in case we may get an issue.

  Below is the code snippet for spliiting the dataset

```
In [28]: from sklearn.model_selection import train_test_split
         X = newdf['reviews']
         y = newdf['score']
         x_train,x_test,y_train,y_test = train_test_split(X,y ,train_size=0.75,test_size=0.25, random_state=101)
```

- Import the sklearn packages- CountVectorizer and TfidfTransformer
- Using Tf-Idf transformer it is possible to systematically compute word counts using **CountVectorizer** and Inverse Document Frequency (**IDF**) values and then compute the **Tf-idf**scores.It computes the word counts, **IDF** values, and **Tf-idf** scores all using the same dataset.
- For Naive Bayes we have imported 'naive_bayes import MultinomialNB'

- To proceed further with the algorithms ,we have taken the train set .

Looking at distribution of 'positives' & 'negatives' samples in train dataset

```
In [46]: train.head()
```

Out[46]:

	reviews	score
54099	love amp reasonslaughsolidly made everything h...	1
3684	work great everything	1
26957	use kindle daily basis good size reading bed	1
31985	smooth movement im glad arent million buttons ...	1
34119	pain butt would recommend	-1

Gives Total No.of Pos and Neg in the train set

```
In [47]: import collections
         %matplotlib inline
         collections.Counter(train['score'])
```

```
Out[47]: Counter({'1': 65472, '-1': 9528})
```

- After importing all the libraries ,we implement naive bayes.
- We have taken train set and fit it to the naive bayes classifier.

Code snippet for the above is shown:

1.Naive Bayes(with precision & Recall, F1-score,confusion matrix)

```
In [305]: from sklearn.feature_extraction.text import CountVectorizer
          from sklearn.feature_extraction.text import TfidfTransformer
          from sklearn.pipeline import Pipeline
          from sklearn.naive_bayes import MultinomialNB
```

```
In [306]: from sklearn.model_selection import train_test_split
          X = newdf['reviews']
          y = newdf['score']
          # Perform a 70-30 time based splitting (shuffle = False)
          X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size = 0.3, shuffle = False)
```

```
In [307]: pipeNB = MultinomialNB()
```

```
In [308]: # Create an object of class CountVectorizer
          bow = CountVectorizer()
          # Call the fit_transform method on training data
          X_train = bow.fit_transform(X_train_raw.values)
```

```
In [261]: X_train.shape
```

```
Out[261]: (53577, 29072)
```

```
In [309]: # Call the transform method on the test dataset
          X_test = bow.transform(X_test_raw.values)

In [263]: X_test.shape

Out[263]: (22962, 29072)

In [310]: std = StandardScaler(with_mean=False)
          X_train = std.fit_transform(X_train)
          X_test = std.transform(X_test)

In [311]: pipeNB.fit(X_train, y_train)

Out[311]: MultinomialNB()
```

- We have used 'Standardscalar' function for standardizing the train set into X-train and y_train
- y=newdf['score'] is the train set,fitting that column as x and y axis ,we get pipeNB.fit(x_train,y_train).
- We tried to predict testing data along with checking accuracy of Precision, Recall & F1-score.(Overall output of NB accuracy and precision,recall and F1-score)
- Here we append our model pipeNB(contains train set) load x_test to the variable pred.
- At last we print y_test and predNB variables with the help of classification module which gives overall accuracy.

precision & Recall , F1-score

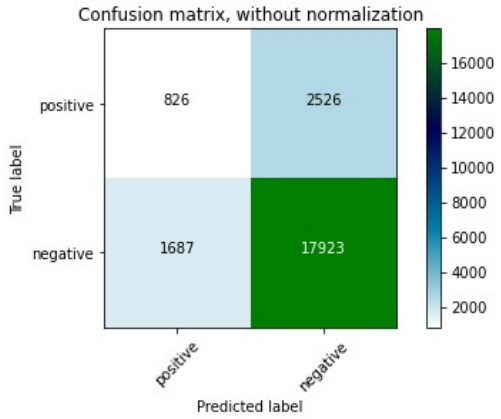
```
In [312]: predNB = pipeNB.predict(X_test) #predict testing data

          from sklearn.metrics import classification_report
          print(classification_report(y_test,predNB))
```

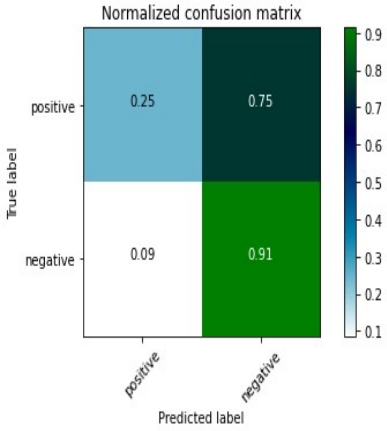
	precision	recall	f1-score	support
-1	0.33	0.25	0.28	3352
1	0.88	0.91	0.89	19610
accuracy			0.82	22962
macro avg	0.60	0.58	0.59	22962
weighted avg	0.80	0.82	0.81	22962

- We got Accuracy of 0.82 on Naive Bayes .

Confusion matrix for Naive bayes(Without Normalized and Normalized)

```
Confusion matrix, without normalization
[[  826  2526]
 [ 1687 17923]]
Normalized confusion matrix
[[0.25 0.75]
 [0.09 0.91]]
```





ROC (Receiver Operating Characteristic curve) For Naïve Bayes.

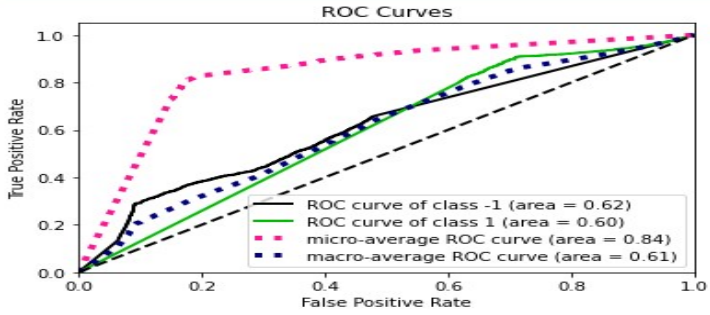
The plot of 'True Positive Rate' (Sensitivity/Recall) against the 'False Positive Rate' (1-Specificity) at different classification thresholds. The area under the ROC curve (AUC) measures the entire two-dimensional area underneath the curve. It is a measure of how well a parameter can distinguish between two diagnostic groups. Often used as a measure of quality of the classification models. A random classifier has an area under the curve of 0.5, while AUC for a perfect classifier is equal to 1.

- We caluculated the AUC Score

```
In [94]: rf_prob = pipeNB.predict_proba(X_test)[:,-1]
         rf_auc = roc_auc_score(y_test,rf_prob)
         print("NaiveByaes AUC: ", rf_auc)

         NaiveByaes AUC:  0.5985900476961854
```

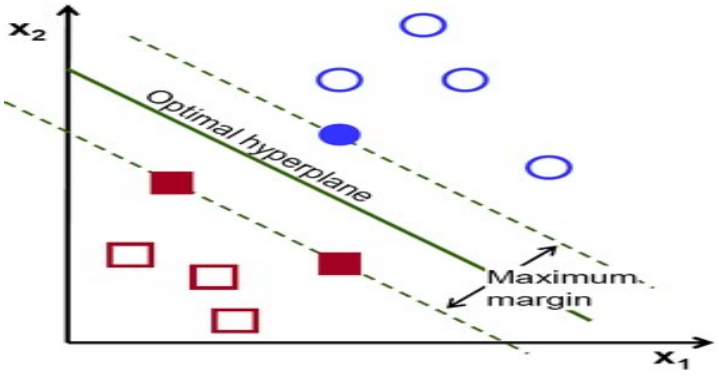
ROC curve

```
In [95]: #ROC
         import scikitplot as skplt #to make things easy
         y_pred_proba =pipeNB.predict_proba(X_test)
         skplt.metrics.plot_roc_curve(y_test, y_pred_proba)
         plt.show()

         C:\Users\Avinash\anaconda3\lib\site-packages\sklearn\utils
         ted; This will be removed in v0.5.0. Please use scikitplot
           warnings.warn(msg, category=FutureWarning)
```



2. Support Vector Machine(SVM)

- A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems.
- After giving an SVM model sets of labeled training data for each category, they're able to categorize new text. So we're working on a text classification problem.
- Objective is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.



- There are many possible hyperplanes that could be chosen.
- Our objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes.
- Here we use LenearSVC method to implement SVM

Process of implementation

- Here we have taken train set as it contains huge samples
- In Naive Bayes we have taken tf-idf classifier but in SVM we have implemented using a method called 'LenearSVC.'
- Based on research SVM always gives most accurate result than Naive Bayes.
- At first we have imported all the packages we need for SVM,the code snippet for implementation as shown below.

Code snippet of implementation

5. SUPPORT VECTOR MACHINE(LinearSVC)

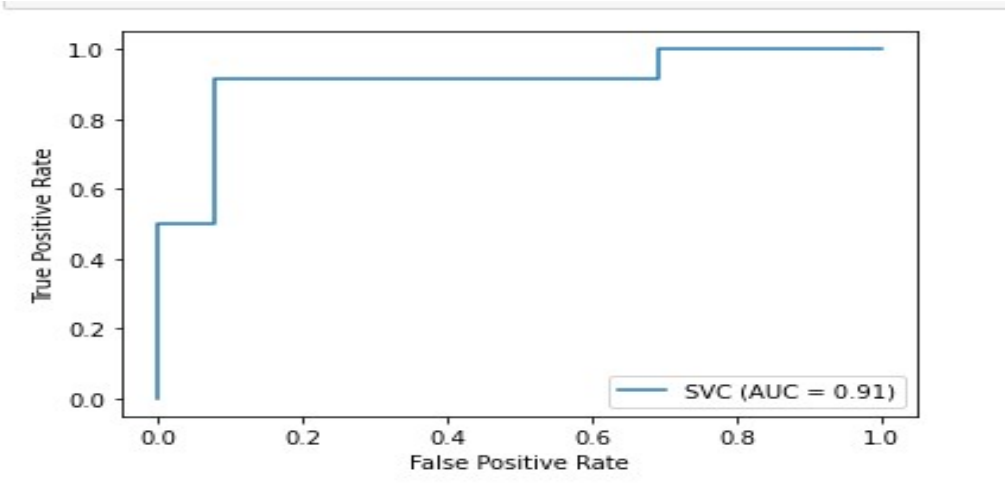
```
In [283]: from sklearn.model_selection import train_test_split
          X = newdf['reviews']
          y = newdf['score']
          X_train,X_test,y_train,y_test = train_test_split(X,y ,train_size=0.75,test_size=0.25, random_state=101)
```

```
In [284]: X_train_raw.shape, y_train.shape, X_test_raw.shape, y_test.shape
Out[284]: ((53577,), (57404,), (22962,), (19135,))
```

ROC with AUC score(fitted train in clf.fit)

```
In [285]: import matplotlib.pyplot as plt
          >>> from sklearn import datasets, metrics, model_selection, svm
          >>> X, y = datasets.make_classification(random_state=0)
          >>> X_train, X_test, y_train, y_test = model_selection.train_test_split(
          ...     X, y, random_state=0)
          >>> clf = svm.SVC(random_state=0)
          >>> clf.fit(X_train, y_train)
          SVC(random_state=0)
          >>> metrics.plot_roc_curve(clf, X_test, y_test)
          plt.show()
```

ROC Curve (SVM)



- Above is the ROC curve showing AUC score-0.91

Precision,Recall,F1-score for SVM

- Here clf.fit(contains train set) appended X-test to the variable pred.
- At last printed with classification module by adding y_test and pred variables which has both test and train.

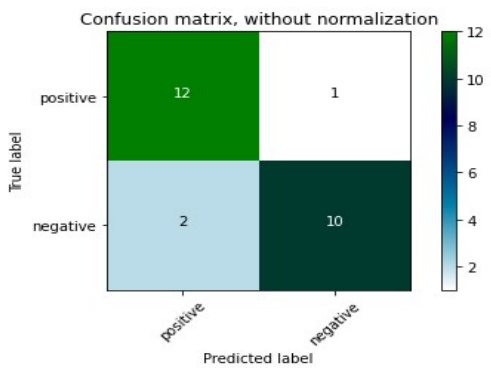
```
In [286]: pred = clf.predict(X_test) #predict testing data

          from sklearn.metrics import classification_report
          print(classification_report(y_test,pred))
```

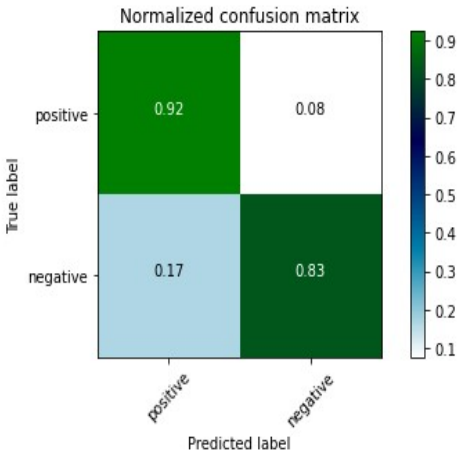
	precision	recall	f1-score	support
0	0.86	0.92	0.89	13
1	0.91	0.83	0.87	12
accuracy			0.88	25
macro avg	0.88	0.88	0.88	25
weighted avg	0.88	0.88	0.88	25

- Here we get 0.88 accuracy on LenearSVC method applied on SVM.
- We got 0.81 on Naïve Bayes But in SVM we got 0.88 ,So we can conclude that SVM is most Accurate than Naive Bayes

Confusion matrix For SVM (Without Normalized and Normalized)

```
Confusion matrix, without normalization
[[12  1]
 [ 2 10]]
Normalized confusion matrix
[[0.92 0.08]
 [0.17 0.83]]
```





3.Logistic Regression(Method used BOW)

- Logistic Regression is a 'Statistical Learning' technique categorized in 'Supervised' Machine Learning (ML) methods dedicated to 'Classification' tasks.
- Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature.
- Logistic regression is most commonly used when the data in question has binary output, so when it belongs to one class or another, or is either a 0 or 1.

Process of implementation

- Import all the required libraries.
- As with above algorithms we have taken df['score'] column for further implementation of algorithm ,which is train set column as it has huge data samples.
- Here also we used standardScalar for column standardization.

The code snippet and outputs for the same is shown below

```
In [66]: newdf['score'].value_counts()

Out[66]: 1     66820
         -1     9719
         Name: score, dtype: int64

In [67]: X = newdf['reviews']
         y = newdf['score']

In [68]: # Perform a 70-30 time based splitting (shuffle = False)
         X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size = 0.3, shuffle = False)

In [69]: # Create an object of class CountVectorizer
         bow = CountVectorizer()
         # Call the fit_transform method on training data
         X_train = bow.fit_transform(X_train_raw.values)

In [70]: X_train.shape

Out[70]: (53577, 35760)

In [71]: # Call the transform method on the test dataset
         X_test = bow.transform(X_test_raw.values)

In [72]: X_test.shape

Out[72]: (22962, 35760)
```

- Performing column standardization with the transform function for X_train and X_test.
- Standardization means the process of putting different variables on the same scale

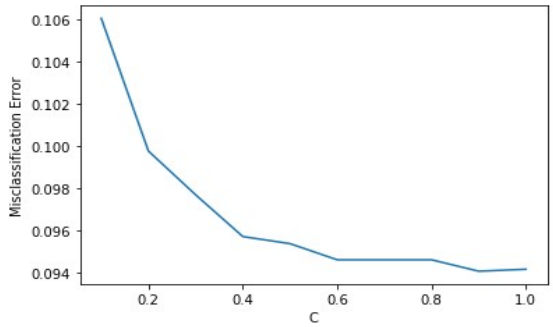
- Basically it is used in Regression analysis
- Cross validation and Hyper Parameter tuning with Logistic Regression. Implementation code is shown below.

```
Perform Column Standardization

In [73]: std = StandardScaler(with_mean=False)
         X_train = std.fit_transform(X_train)
         X_test = std.transform(X_test)
```

- We have created an object of the class LogisticRegression with balanced class weights.
- Assigned to the variable ‘clf ’i.e-->clf = LogisticRegression(C = c, class_weight = 'balanced').
- Perform 5-fold cross validation,which returns the cv accuracy for each fold in a list.
- It is then assigned to the variable ‘score’ i.e--> scores = cross_val_score(clf, X_train, y_train, cv=5, scoring='accuracy').
- Storing the mean of the accuracies from all the 5 folds and loading it to the variable ‘cv_score’ i.e-->cv_scores.append(scores.mean()).
- Calculate misclassification error from accuracy (error = 1 - accuracy) i.ecv_error = [1 - x for x in cv_scores]

Plot misclassification error vs C

```
In [75]: # plot misclassification error vs C
         plt.plot(C_values, cv_error)
         plt.xlabel('C')
         plt.ylabel('Misclassification Error')
         plt.show()
```



- Below code snippet is final step of Logistic regression implementation to find accuracy
- Lf_mod is the class of Logistic regression, we fit our model and store it in the variable ‘Lf_mod.fit.
- Below is Code which gives All the matrics values

Precision,Recall,F1-score for Logistic Regression

```
In [273]: lg_mod = LogisticRegression()
          lg_mod.fit(X_train, y_train)

          print("Train score: ",lg_mod.score(X_train, y_train))

          pred_lg = lg_mod.predict(X_test)
          print('Test score:',lg_mod.score(X_test, y_test))

          print("Classification report: ",classification_report(pred_lg,y_test))
```

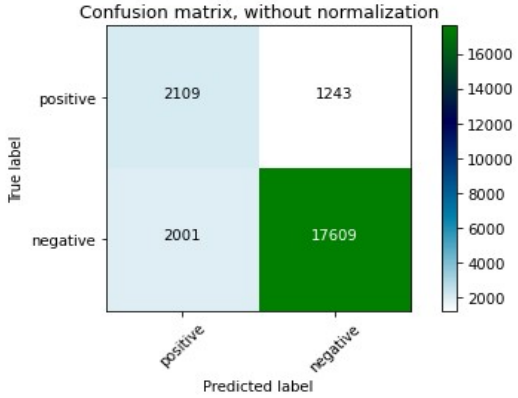
```
Train score:  0.9980028743677324
Test score: 0.8587231077432279
Classification report:                precision    recall  f1-score   support

          -1       0.63      0.51      0.57      4112
           1       0.90      0.93      0.92     18850

    accuracy                           0.86     22962
   macro avg       0.76      0.72      0.74     22962
weighted avg       0.85      0.86      0.85     22962
```

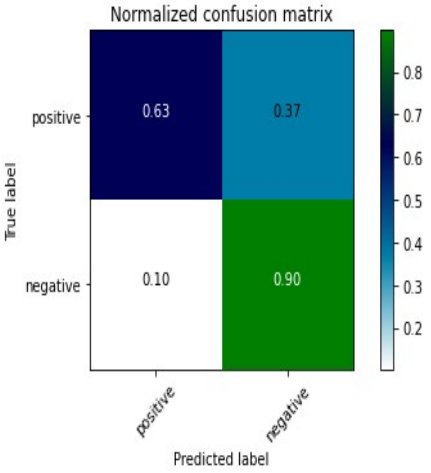
- We got 86% of accuracy for Logistic Regression algorithm
- Let's Check by applying normal method ,along with accuracy of precision and recall & F1-score.
- Here lg_mod is the class that contains X_train and Y_train(Train set).
- We load X_test and store it to the variable and print using classification module along with y_test.

- The accuracy for Logistic Regression is 86%.

Confusion matrix For Logistic Regression(Without Normalized and Normalized)

```
Confusion matrix, without normalization
[[ 2109  1243]
 [ 2001 17609]]
Normalized confusion matrix
[[0.63 0.37]
 [0.1  0.9 ]]
```



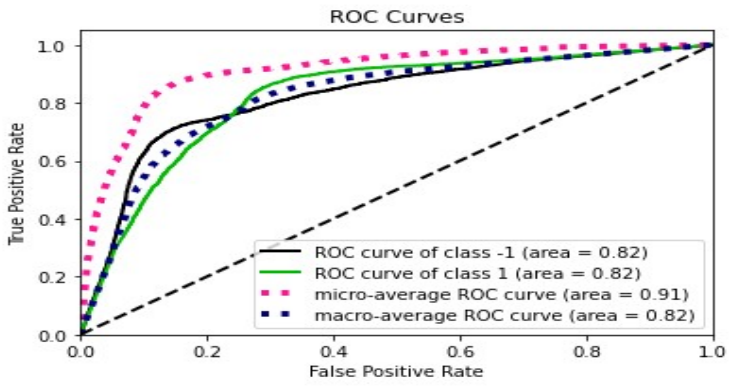


AUC Score(Logistic Regression)

```
In [182]: lg_prob = lg_mod.predict_proba(X_test)[:,-1]
          lg_auc = roc_auc_score(y_test, lg_prob)
          print("Logistic Regression AUC: ",lg_auc)

          Logistic Regression AUC:  0.8224521136505534
```

ROC curve(Logistic Regression)

```
In [183]: #ROC
          import scikitplot as skplt #to make things easy
          y_pred_proba =lg_mod.predict_proba(X_test)
          skplt.metrics.plot_roc_curve(y_test, y_pred_proba)
          plt.show()
```



4.Decison Tree Algorithm

- Decision Trees are a type of Supervised Machine Learning (that is we can explain what the input is and what the corresponding output is in the training data) where the data is continuously split according to a certain parameter.
- Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes.
- The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.
- To get corresponding output we take training data samples.
- Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.
- The logic behind the decision tree can be easily understood because it shows a tree-like structure.
- But here we are just implementing decision to get accuracy.
- Decision trees can be divided into two types; categorical variable and continuous variable decision trees.
- Decision tree contains categorical data(Yes/No)and numerical data.
- As a standard practice, you may follow 70:30 to 80:20 as needed.But we should estimate how accurately the classifier predicts the outcome. The accuracy is computed by comparing actual test set values and predicted values.

There are several advantages of using decision tree for predictive analysis:

- Decision trees can be used to predict both continuous and discrete values i.e. they work well for both regression and classification tasks.
- They require relatively less effort for training the algorithm.
- They can be used to classify non-linearly separable data

Process of implementation

- Import required libraries i.e→ DecisionTreeclassifier

Performing The decision tree analysis using scikit learn.

Splitting the dataset

2. DECISION TREE

```
In [96]: from sklearn.tree import DecisionTreeClassifier

         from sklearn.preprocessing import StandardScaler

         from sklearn.metrics import accuracy_score, confusion_matrix
         from sklearn.model_selection import train_test_split, cross_val_score
```

```
In [97]: from sklearn.model_selection import train_test_split
         X = newdf['reviews']
         y = newdf['score']
         # Perform a 70-30 time based splitting (shuffle = False)
         X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size = 0.3, shuffle = False)
```

- Here alos we have performed column standardization with' StandardScalar'
- We have Fitted our Train model -→dtree.fit
- Called dtree=DecisonTreeClassifie()
- Below is the code for the same

Perform Column Standardization

```
In [103]: std = StandardScaler(with_mean=False)
          X_train = std.fit_transform(X_train)
          X_test = std.transform(X_test)
```

```
In [104]: import pandas as pd
          from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
          from sklearn.model_selection import train_test_split # Import train_test_split function
          from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation
```

```
In [272]: # Create Decision Tree classifer object
          dtree = DecisionTreeClassifier(max_depth=4)

          # Train Decision Tree Classifer
          dtree = dtree.fit(X_train,y_train)

          #Predict the response for test dataset
          y_pred = dtree.predict(X_test)

          # Model Accuracy, how often is the classifier correct?
          print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

          Accuracy: 0.853976134483059
```

- We got accuracy 0.85 that is 85% accurate

Precision, Recall and F1-score

Precisison and Recall

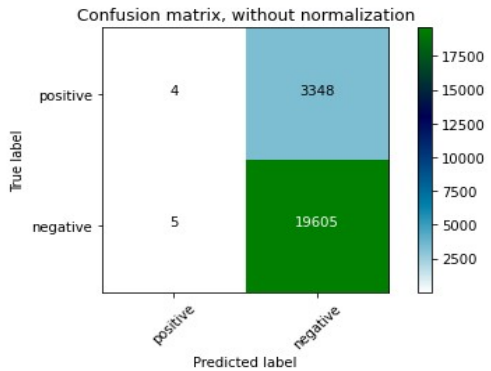
```
In [147]: pred = dtree.predict(X_test) #predict testing data

          from sklearn.metrics import classification_report
          print(classification_report(y_test,pred))
```

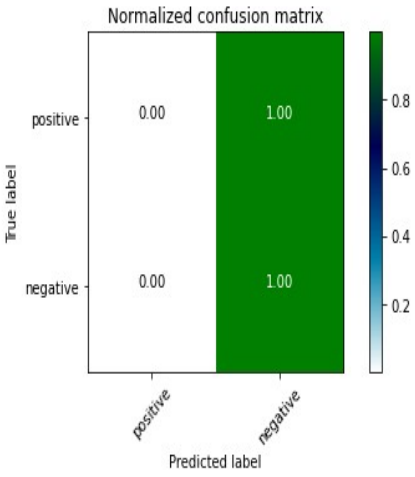
	precision	recall	f1-score	support
-1	0.44	0.00	0.00	3352
1	0.85	1.00	0.92	19610
accuracy			0.85	22962
macro avg	0.65	0.50	0.46	22962
weighted avg	0.79	0.85	0.79	22962

- On calculating using the normal method of classification report we got an accuracy of 85%.

Confusion Matrix(without Normalized And Normalized)

```
Confusion matrix, without normalization
[[    4  3348]
 [    5 19605]]
Normalized confusion matrix
[[1.19e-03 9.99e-01]
 [2.55e-04 1.00e+00]]
```



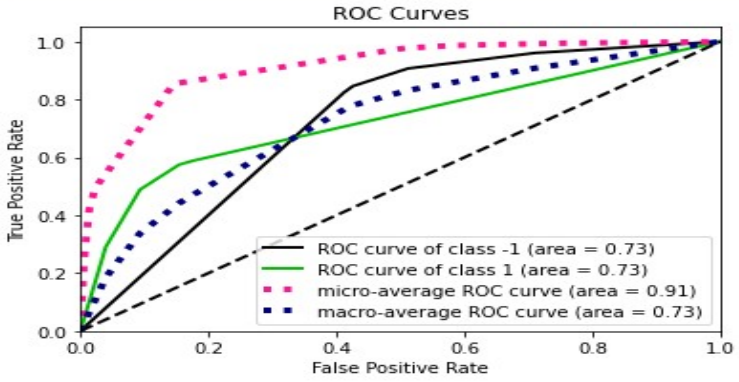


AUC Score(Decision Tree)

```
In [158]: from sklearn.metrics import roc_auc_score, roc_curve

In [159]: rf_prob = dtree.predict_proba(X_test)[:,-1]
          rf_auc = roc_auc_score(y_test,rf_prob)
          print("DecisionTree AUC: ", rf_auc)

          DecisionTree AUC:  0.7262486475533039
```

ROC Curve(Decision Tree)

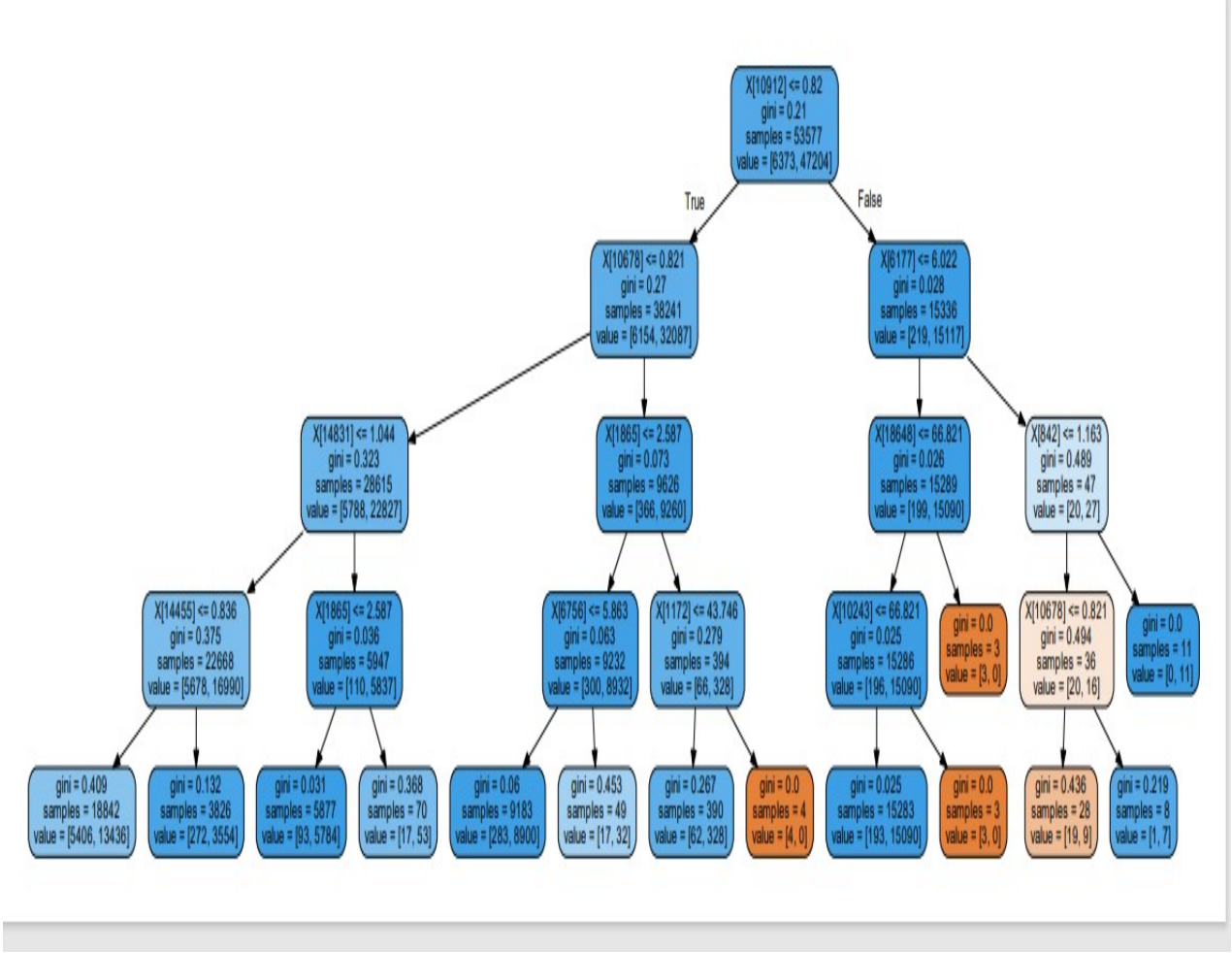
```
In [160]: #ROC
          import scikitplot as skplt #to make things easy
          y_pred_proba =dtree.predict_proba(X_test)
          skplt.metrics.plot_roc_curve(y_test, y_pred_proba)
          plt.show()

          C:\Users\Avinash\anaconda3\lib\site-packages\sklearn\utils\deprecatio
          ted; This will be removed in v0.5.0. Please use scikitplot.metrics.pl
            warnings.warn(msg, category=FutureWarning)
```



We have Plotted decision Tree Visual representation for our model Below is the code

```
In [162]: from sklearn.tree import export_graphviz
          from sklearn.externals.six import StringIO
          from IPython.display import Image
          import pydotplus

In [161]: from sklearn.tree import export_graphviz

          export_graphviz(
                  dtree,
                  out_file="AMAZON_TEXT_REVIEWS_DTree.dot",
                  rounded=True,
                  filled=True
              )

In [58]: conda install graphviz
```

Decision Tree Visualization(With Depth value=4)



5. Random Forest Algorithm

- Random forest is a supervised learning algorithm which is used for both classification as well as regression.
- It creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting.
- Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.
- Random forest has nearly the same hyperparameters as a decision tree or a bagging classifier.
- Random forest adds additional randomness to the model, while growing the trees
- But as stated, a random forest is a collection of decision trees.With that said, random forests are a strong modeling technique and much more robust than a single decision tree.

Process of implementation

- Load Required libraries from sklearn.ensemble import RandomForestClassifier
- Splitted our dataset into X and Y

Below is the code snippet for the same

```
In [185]: X = newdf['reviews']
          y = newdf['score']

In [186]: # Perform a 70-30 time based splitting (shuffle = False)
          X_train_raw, X_test_raw, y_train, y_test = train_test_split(X, y, test_size = 0.3, shuffle = False)

In [187]: X_train_raw.shape, y_train.shape, X_test_raw.shape, y_test.shape
Out[187]: ((53577,), (53577,), (22962,), (22962,))

In [188]: import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt

          import sqlite3    ## SQL Interface
          import pickle     ## Used to save your data - Converts objects to byte stream and vice versa
          import time

          from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

          from sklearn.linear_model import LogisticRegression
          from sklearn.preprocessing import StandardScaler

          from sklearn.metrics import accuracy_score, confusion_matrix
          from sklearn.model_selection import train_test_split, cross_val_score
```

- Here also we have used ‘standardScalar’ for colum means we done column standardization.
- Below is the code snippet for the same.

```
In [189]: # Create an object of class CountVectorizer
          bow = CountVectorizer()
          # Call the fit_transform method on training data
          X_train = bow.fit_transform(X_train_raw.values)

In [190]: X_train.shape
Out[190]: (53577, 29072)

In [191]: # Call the transform method on the test dataset
          X_test = bow.transform(X_test_raw.values)

In [192]: X_test.shape
Out[192]: (22962, 29072)

In [193]: std = StandardScaler(with_mean=False)
          X_train = std.fit_transform(X_train)
          X_test = std.transform(X_test)
```

Fitting our model-→(rf_mod.fit)

```
In [195]: rf_mod = RandomForestClassifier(n_estimators=200, criterion='entropy', random_state= 10, verbose= 1)
          rf_mod.fit(X_train, y_train)

          print("Train score: ",rf_mod.score(X_train, y_train))

          pred_rf = rf_mod.predict(X_test)
          print('Test score:',rf_mod.score(X_test, y_test))

          [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
          [Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:  4.3min finished
          [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
          [Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    7.9s finished
          [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

          Train score:  0.9999813352744648

          [Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    3.7s finished
          [Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

          Test score: 0.8957843393432627

          [Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    3.7s finished
```

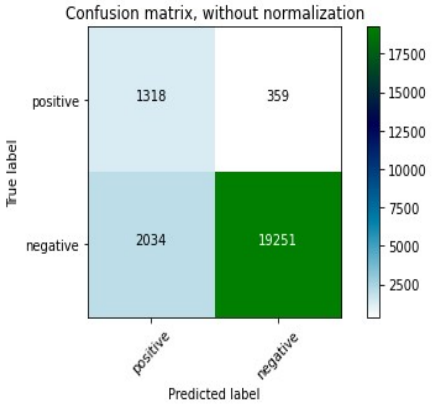
Precision And Recall,F1-score

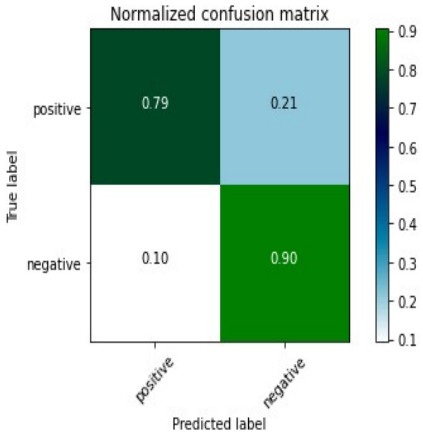
```
In [196]: print("Classification report: ",classification_report(pred_rf,y_test))

          Classification report:                precision    recall  f1-score   support

                    -1       0.39      0.79      0.52      1677
                     1       0.98      0.90      0.94     21285

              accuracy                           0.90     22962
             macro avg       0.69      0.85      0.73     22962
          weighted avg       0.94      0.90      0.91     22962
```

- We got 0.90 that is 90% accurate

Confusion Matrix(without Normalized And Normalized)

```
Confusion matrix, without normalization
[[ 1318   359]
 [ 2034 19251]]
Normalized confusion matrix
[[0.79 0.21]
 [0.1  0.9 ]]
```





AUC Score

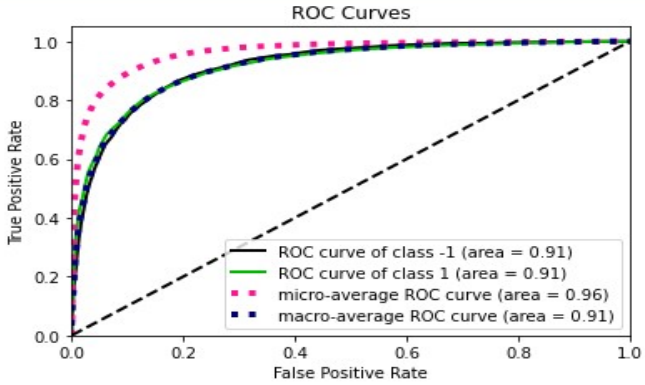
In [208]:
```
rf_prob = rf_mod.predict_proba(X_test)[:,-1]
rf_auc = roc_auc_score(y_test,rf_prob)
print("RandomForest AUC: ", rf_auc)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
RandomForest AUC:  0.9149075985293169
```

ROC Curve

In [209]:
```
#ROC
import scikitplot as skplt #to make things easy
y_pred_proba =rf_mod.predict_proba(X_test)
skplt.metrics.plot_roc_curve(y_test, y_pred_proba)
plt.show()
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 200 out of 200 | elapsed:    3.7s finished
```



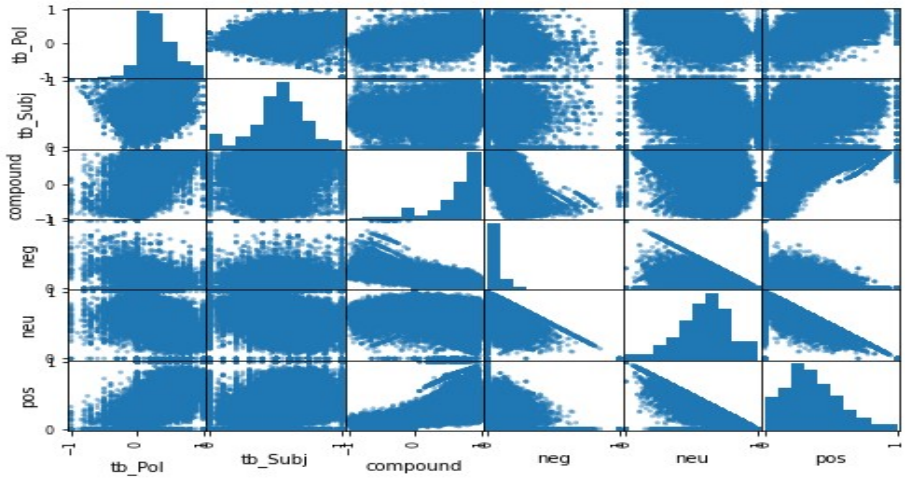
So Random Forest is most accurate than all Because its accuracy-->90%

Model Evaluation

MODEL COMPARISION

```
In [246]: from pandas.plotting import scatter_matrix
```

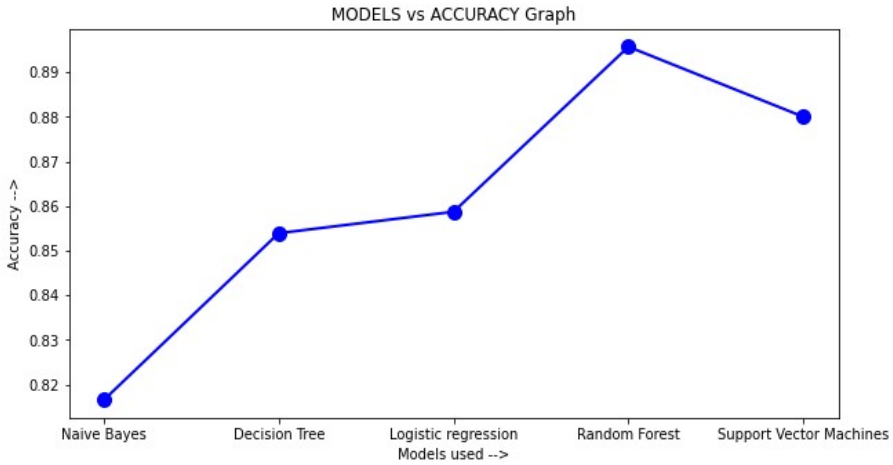
```
In [249]: scatter_matrix(df, figsize=(8, 6))
          plt.show()
```



- **Above is the scatter plot for our dataset**

```
In [299]: acc=[0.8165,0.8539,0.8587,0.8957,0.8800]
```

```
In [315]: acc=[0.8165,0.8539,0.8587,0.8957,0.8800]
          models=["Naive Bayes","Decision Tree","Logistic regression","Random Forest","Support Vector Machines"]
          plt.figure(figsize=(10,5))
          plt.plot(models,acc,'bo',linestyle='solid',linewidth=2, markersize=10)
          plt.xlabel("Models used -->")
          plt.ylabel("Accuracy -->")
          plt.title("MODELS vs ACCURACY Graph")
          plt.show()
```



- First we have loaded the accuracy value to the arry and then performed the function
- Above we can see the comparison of all the algorithms to find which is best according to its accuracy, so we can see here Random Forest is the best of all

CHAPTER 6

RESULTS AND DISCUSSION

This approach aims how to improve quality of sentiment analysis on textual product reviews and simple visual representation of obtained results which will be useful for nontechnical users.Individual customer can take its benefit for decision making and service provider can take advantage to improve quality of service as well as for new product design. Sentiment Analysis is used to determine attitude of mass people towards particular product or service.

What we have done so far:

- Imported all libraries which is required for our project
- Manually created dataset which is text reviews(contains 76540 data of 4 different types of products available in Amazon)
- Data Normalization (Pre-processing)
    - Converting Emojis to their Respective Emotions
    - Removing all Punctuations in the Reviews
    - Removing StopWords
    - Tokenize the Data
    - Stemming
    - Lemmatization
    - Bag of Words(BOW)
- We have applied 4 algorithms and checked the accuracy in terms of precision ,F1score,recall and confusion matrix.
- Since it is sentiment analysis of reviews given by the customer ,we have not included ratings.
- Challenge here is ,we plan to give the score code for each line of the reviews , where we consider words like bad,worstetc as negative.Words like excellent, good,welletc will be set as positive.

- Final output will be in a score code in front of the review line i.e Negative or positive .
- We will try to present it as Dashboard (Front-end)

CHAPTER 7

CONCLUSION

Internet is rich source of reviews on ecommerce products or online services. Customer always prefer to read reviews before paying money to the service provider. But it is hardly possible to read all reviews in today's fast life. Also every review may provide new information of product or feature of product. So there is probability of missing any important review given by consumer.

We are going to identify polarity of review i.e. whether it is positive, negative or neutral. Sentiment analysis will assist us to find out polarity of reviews. Due to large number of reviews we are going to convert textual reviews into visual format by using NLTK it will enhance reliability in decision making.

References

[1] Xing Fang* and Justin Zhan “Sentiment analysis using product review data” Fang and Zhan Journal of Big Data (2015) 2:5

[2] Muhammad Taimoor Khan, Mehr Durrani2, Armughan Ali, IrumInayat, Shehzad Khalid and Kamran Habib Khan “Sentiment analysis and the complex natural language” Khan et al. Complex Adapt Syst Model (2016) 4:2

[3] Ji fang, Bi Chen, “Incorporating Lexicon Knowledge into SVM Learning to Improve Sentiment Classification”, Proceedings of the Workshop on Sentiment Analysis where AI meets Psychology (SAAIP), IJCNLP 2011, pp. 94–100.

[4] Turney, Peter D., “Thumbs up or thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews”, Proceedings of Association for Computational Linguistics, Philadelphia, PA. July 2002, pp. 417-424.

[5] Turney, Peter D., “Thumbs up or thumbs down? Semantic Orientation Applied to Unsupervised Classification of Reviews”, Proceedings of Association for Computational Linguistics, Philadelphia, PA. July 2002, pp. 417-424.

[6] AshishShukla* Rahul MisraM.tech Scholar, CSE Department Assistant Professor, CSE Department , Pranveer Singh Institute of Technology, Kanpur Pranveer Singh Institute of Technology, Kanpur U.P.T.U., Luck now, Uttar Pradesh, India U.P.T.U., Luck now, Uttar Pradesh, India “Sentiment Classification and Analysis Using Modified K-Means and Naïve Bayes Algorithm”

[7] Negar Hariri, Carlos Castro-Herrera, Member, IEEE, Mehdi Mirakhorli, Student Member, IEEE, Jane Cleland-Huang, Member, IEEE, and BamshadMobasher, Member, IEEE “Supporting Domain Analysis through Mining and Recommending Features from Online Product Listings” IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 12, DECEMBER 2013.

[8] H. Cherfi · A. Napoli · Y. Toussaint “Towards a text mining methodology using association rule extraction”.

[9] Opinion mining and sentiment analysis, Bo Pang1 and Lillian Lee2. 1 Yahoo! Research, 701 First Ave. Sunnyvale, CA 94089, U.S.A., bopang@yahooinc.com, 2

Computer Science Department, Cornell University, Ithaca, NY 14853, U.S.A., llee@cs.cornell.edu

[10] Sentiment Analysis and Opinion Mining: A Survey G.Vinodhini* Assistant Professor, Department of Computer Science and Engineering, Annamalai University, Annamalai Nagar-608002. RM.Chandrasekaran, Professor, Department of Computer Science and Engineering, Annamalai University, Annamalai Nagar-608002. India.

[11] Customers behaviour prediction using artificial neural network BichenZheng, Keith Thompson, Sarah S.Lam, Sang Won Yoon, Nathan Gnanasambandam.

[12] An Approach to Sentiment Analysis using Artificial Neural Network with Comparative Analysis of Different Techniques PranaliBorele, Dilipkumar A. Borikar

[13] Review on classification based on artificial neural netwoks, Saravanan K1 and S. Sasithra2.