Twitter Dataset Based Sentimental Analysis with Machine Learning Approach

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION ARTIFICIAL INTELLIGENCE ANDMACHINE LEARNING

Submitted by:

ENDULURU VIGNESH - 20BCS6891

KETHURI AJAY - 20BCS6585

VUPPUTURI BHARATH-20BCS6586

KOYYADA AKSHAY KUMAR – 20BCS6380

Under the Supervision of:

DR. RANJAN WALIA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

APEX INSTITUE OF TECHNOLOGY

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

Feb - May. 2023

BONAFIDE CERTIFICATE

I Certified that this project report "**Twitter Dataset Based Sentimental Analysis with Machine Learning Approach**" is the bonafide work of **ENDULURU VIGNESH, KETHURI AJAY, VUPPUTURI BHARATH, KOYYADA AKSHAY KUMAR** who carried out the project work under my supervision.

SIGNATURE OF THE HOD
MR. AMAN KOUSHIK
(HEAD OF THE DEPARTMENT**)**
CSE - AIML

SIGNATURE OF THE SUPERVISOR
DR. RANJAN WALIA
(SUPERVISOR)
(Asst.Professor)
(AIT – CSE)

Submitted for the project viva-voce examination held on

INTERNAL EXAMINER **EXTERNAL EXAMINER**

TABLE OF CONTENTS

List of Figures:

ABSTRACT

A Twitter sentiment analysis is the process of determining the emotional tone behind a series of words, specifically on Twitter. A sentiment analysis tool is an automated technique that extracts meaningful customer information related to their attitudes, emotions, and opinions. Sentiment analysis, also referred to as opinion mining, is an approach to natural language processing (NLP) that identifies the emotional tone behind a body of text. This is a popular way for organizations to determine and categorize opinions about a product, service, or idea. Sentiment analysis or opinion mining refers to identifying as well as classifying the sentiments that are expressed in the text source. Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of people on social media for a variety of topics.

Therefore, we need to develop an Automated Machine Learning Sentiment Analysis Model in order to compute customer perception. Due to the presence of non-useful characters (collectively termed as noise) along with useful data, it becomes difficult to implement machine learning models on them. The goal of tweet sentiment analysis is to find the positive, negative, or neutral sentiment part in the tweeter data. Sentiment analysis can help any organization to find people's opinions of their company and products. We have applied sentiment analysis on twitter data set. Our model takes input tweet, sentiment, and output selected text starting and ending in input tweet. We are using Natural Language Processing model with one more layer to find sentiment positions in tweets. The extra layer is used to find the sentiment part of a tweet. Our model can achieve 80 percent accuracy on the validation data set.

Keywords— Sentiment Analysis, Category-Classification, Logistic Regression, Support Vector Machine, Natural Language Processing (NLP), Twitter Classification, and Machine Learning.

Acknowledgement

It gives us immense pleasure to express our deepest sense of gratitude and sincere thanks to our respected guide Dr. Ranjan Walia (Professor), CSE- Artificial Intelligence, Chandigarh University, Mohali for his valuable guidance, encouragement and help for completing this work. Her useful suggestions for this whole work and cooperative behavior are sincerely acknowledged. We are also grateful to Dr. Shikha Gupta (Program Leader, CSE-AIML) for her constant support and guidance.

We also wish to express our indebtedness to our family members whose blessings andsupport always helped us to face the challenges ahead. We also wish to express thanks to all people who helped us in completion of this project.

ENDULURU VIGNESH 20BCS6891

KETHURI AJAY 20BCS6585

VUPPUTURI BHARATH 20BCS6586

KOYYADA AKSHAY KUMAR 20BCS6369

Chapter – 1 INTRODUCTION

Nowadays, the age of Internet has changed the way people express their views, opinions. It is now mainly done through blog posts, online forums, product review websites, social media, etc. Nowadays, millions of people are using social network sites like Facebook, Twitter, Google Plus, etc. to express their emotions, opinion and share views about their daily lives. Through the online communities, we get an interactive media where consumers inform and influence others through forums. Social media is generating a large volume of sentiment rich data in the form of tweets, status updates, blog posts, comments, etc.

Moreover, social media provides an opportunity for businesses by giving a platform to connect with their customers for advertising. People mostly depend upon user generated content over online to a great extent for decision making. For e.g., if someone wants to buy a product or wants to use any service, then they firstly look up its online, discuss about it on social media before taking a decision. The amount of content generated by users is too vast for a normal user to analyze. So, there is a need to automate this, various sentiment analysis techniques are widely used. Sentiment analysis (SA)tells user whether the information about the product is satisfactory or not before they buy it.

Marketers and firms use this analysis data to understand about their products or services in such a way that it can be offered as per the user's requirements. Textual Information retrieval techniques mainly focus on processing, searching, or analyzing the factual data present. Facts have an objective component but, there are some other textual contents which express subjective characteristics. These contents are mainly opinions, sentiments, appraisals, attitudes, and emotions, which form the core of Sentiment Analysis (SA). It offers many challenging opportunities to develop new applications, mainly due to the huge growth of available information on online sources like blogs and social networks. For example,

recommendations of items proposed by a recommendation system can be predicted by considering considerations such as positive or negative opinions about those items by making use of SA.

The reports say that, millions of people are using Social networks like Twitter, Instagram and Facebook websites to express their feelings, personal opinions and publish about their daily lives. However, people express or write different tweets on different platforms such as sharing any good news or some achieved things in their life. Sentiment Analysis which is done under Natural Language Processing that identifies the emotion of the text that a particular person posted.

Currently, the advent of Internet has transformed the way individuals communicate their thoughts, opinions. Nowadays, it is done primarily through blog postings, internet forums, websites that offer product, social media, etc. Consumers heavily rely on user generated content from the internet when making decisions. For instance, before deciding, someone who wants it to purchase a product or use a service will first research online and engage in discussion about it on social media. The volume of user-generated content is too great for a typical user to process. As a result, numerous sentiment analysis approaches are frequently used, and this must be automated. Before a user purchases a product, sentiment analysis (SA) lets them know if the product's information is good or not. Marketers and businesses use this analysis data to learn more about their goods or services so that they can cater to the needs of the customer. Techniques for retrieving information from text typically concentrate on processing, looking up, or interpreting the factual data that is already there.

Although if facts have an objective component, some other literary contents exhibit subjective traits. Sentiment Analysis's fundamental components—opinions, sentiments, assessments, attitudes, and emotions—are primarily represented by these contents (SA). In large part because of the enormous expansion in the amount

of information available online from sources like blogs and social networks, it presents many challenging chances to design new applications. Twitter Sentiment Analysis means, using advanced text mining techniques to investigate the sentiment of the text (here, tweet) within the sort of positive, negative, and neutral. it's also called Opinion Mining, is primarily for analyzing conversations, opinions, and sharing of views (all within the sort of tweets) for deciding business strategy, political analysis, and also for assessing public actions.

Sentiment analyses are often wanted to identify trends within the content of tweets, which are then analyzed by machine learning algorithms. Sentiment analysis is a crucial tool within the eld of social media marketing because it will discuss how it will be accustomed to predict the behavior of a user's online persona. Sentiment analysis is employed to investigate the sentiment of a given post or investigate any given topic. In fact, it is one of the foremost popular tools in social media marketing. Text understanding could be a Signiant problem to resolve.

One approach may well be to rank the importance of sentences within the text then generate a summary for the text supported by the important numbers. These systems do not depend on manually crafted rules, but on machine learning techniques, like classification. Classification, which is employed for sentiment analysis, is an automatic system that must be fed sample text before returning a category, e.g., positive, negative, or neutral. Urgent issues will often arise, and they must be restrained immediately. A complaint on Twitter, for instance, could quickly escalate into a PR crisis if it goes viral. While it would be difficult for your team to spot a crisis before it happens, it is very easy for machine learning tools to identify these situations in real-time. Patterns are often extracted from analyzing the frequency distribution of those parts of speech (either individually or collectively with some other parts of speech) during a particular labeled tweet.

1.1 PROBLEM DEFINITION

In this project, we try to implement an NLP Twitter sentiment analysis model that helps to overcome the challenges of sentiment classification of tweets. We will be classifying the tweets into positive or negative sentiments. The necessary details regarding the dataset involving the Twitter sentiment analysis project are:

The dataset provided is the MP Twitter Dataset which consists of 31961 tweets that have been extracted using the Twitter API. The various columns present in this Twitter data are:

target: the polarity of the tweet (positive or negative)

ids: Unique id of the tweet

label: Type of the tweet

Tweet: Text of the Tweet

1.2 PROJECT OVERVIEW

In this project, a dataset is taken for reference of tweets for instance racist tweets. After training and testing, the model is ready to classify whether the tweet is positive or negative accordingly. The only drawback/ limitation is that dataset is small. It leads to low accuracy

1.3 TIMELINE

S.N	Strategies	1st week	2nd week	3rd week	4th week	5th week	6th week
1)	Problem Identification						
2)	Research & Analysis						
3)	Design						
4)	Coding						
5)	Implementation & testing						
6)	Project finalisation						
7)	Documentation						

1.4 HARDWARE AND SOFTWARE REQUIREMENTS

HARDWARE SPECIFICATIONS

- Personal computer with keyboard and mouse maintained with uninterrupted power supply.
- Processor: Intel® core™ i5
- Installed Memory (RAM): 8.00 GB

SOFTWARE SPECIFICATIONS

- Operating System: WINDOWS 7, 8.1,10,11
- Coding language: PYTHON
- Web Browser: GOOGLE CHROME
- Libraries used:
  - Sklearn
  - Matplotlib
  - Numpy
  - Pandas
  - NLTK
  - Seaborn

Chapter – 2

LITERATURE SURVEY

2.1 Books related to Sentiment Analysis:

1. "Sentiment Analysis and Opinion Mining" by Bing Liu: This book provides a comprehensive overview of sentiment analysis techniques and approaches, including sentiment analysis in the context of tweets.

2. "Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More" by Matthew A. Russell: While not solely focused on tweets, this book covers various social media platforms and discusses techniques for sentiment analysis, which can be applied to tweets as well.

3. "Text Mining and Analysis: Practical Methods, Examples, and Case Studies Using SAS" by Goutam Chakraborty, Murali Pagolu, and Satish Garla: This book explores text mining techniques using SAS software, including sentiment analysis. It provides practical examples and case studies that can be helpful for analyzing tweets.

4. "Sentiment Analysis in Social Networks" by Federico Alberto Pozzi, Elisabetta Fersini, Enza Messina, and Bing Liu: Although it focuses on sentiment analysis in social networks, this book covers techniques and methodologies that are applicable to tweets review analysis. It explores sentiment analysis algorithms and their applications in different domains.

5. "Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning" by Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda: This book provides practical examples and step-by-step guides for performing text analysis tasks, including sentiment analysis. It includes Python code examples and covers various techniques relevant to analyzing tweets.

6. "Sentiment Analysis and Opinion Mining in Python" by Siddhartha Chatterjee: This book provides practical examples and step-by-step guides to perform sentiment analysis using Python, with a focus on analyzing tweets.

7. "Deep Learning for Sentiment Analysis: From Theory to Practice" by Raghavendra Pappagari: This book explores deep learning techniques and their application in sentiment analysis, offering insights and practical tips for analyzing sentiments in tweets.

8. "Text Mining Cookbook: Practical Recipes to Help You Extract Value from Unstructured Data" by Prabhanjan Tattar, Krishna Birmiwal, and Suresh Kumar Mukhiya: This cookbook-style guide offers practical recipes for text mining tasks,

including sentiment analysis, helping you leverage unstructured data such as tweets.

9. "Opinion Mining and Sentiment Analysis: A Lexicon-Based Approach" by Erik Cambria, Bing Liu, and Amir Hussain: This book focuses on lexicon-based approaches to sentiment analysis, discussing techniques for mining opinions and sentiments from textual data, including tweets.

10."Machine Learning for Text Analysis" by Charu C. Aggarwal: This book provides an in-depth exploration of machine learning techniques for text analysis, covering topics such as sentiment analysis and text classification, with relevance to tweets.

2.1 RELATED WORK

The paper [1] (Mittal, 2016) describe the requirement and impact of the sentiment analysis on on-line platform. They need additionally bestowed a listing of sentiments of emotions, interjections and comments that are extracted from posts and standing updates. They need got result to knowing whether or not {the on-line the web the net} and posts are being useful to client or not and that on-line websites being most popular by the purchasers.

The paper [2] (Anto, 2016) describe the merchandise rating mistreatment sentiment analysis. In promoting of any product the producer can get the proper result from the client feedback. After got feedback they'll changes to his product in step with the feedback. Some users continually fail to convey their feedbacks. Objective of this paper is to avoid the problem of providing feedbacks and supply the technique which might provide automatic feedback on the premise of information collected from twitter. They used the technique SVM and got result eightieth accuracy. This system offers quick and valuable feedback.

The paper [3] (Saragih, 2017) describe regarding the client engagement by analysis the comments on social media in transport on-line. They used technique TF-IDF. The result shows that the class “Feedback system by driver” and “Feedback system by user” have the foremost comments for 3 means that of transports online, whereas class “service quality for driver “has the littlest comments. This feedback of social media is accustomed evaluate the performance of this business transport on-line.

This paper [4] (Mamgain, 2016) describe regarding the sentiment analysis of people’s opinions relating to high faculties in India. They need represented comparison between the result obtained by the subsequent machine learning algorithms: Naive Bayes and SVM and Artificial Neural Network model: Multilayer Perception. Naive Bayes {Thomas Bayes mathematician} outperforms SVM for the aim of matter polarity classification that is fascinating as a result of the model utilized by Naive Bayes is easy (use of freelance probabilities) and therefore the likelihood estimates made by such a model.

2.2 EXISTING SYSTEM

Sentiment analysis is a major approach for classifying the given text into positive and negative in Natural Language Processing (NLP).

In [1] 2023, Zahratu Sabrina explained what are the different approaches for sentiment analysis. It includes Lexicon-based approach, which works by splitting the sentences into bag of words and compare them with the words of sentiment polarity lexicon and semantic relations. Second methos used is machine learning approach, which uses classifiers to extract features from the data. It includes data collecting, data pre-processing, extracting features, training data and analyzing results.

In [2] 2023, Shamsuddeen Hassan Muhammad implemented sentiment analysis for various languages in Africa. It explains the challenges for African languages in sentiment analysis. Previously polarity based approach is used for few African languages. In this research paper, Afrisent dataset used consisting of 14 other African languages .

In [3] 2021, Vedurumudi Priyanka used ensembling technique, which combines various classifiers, in order to improve the accuracy for the dataset taken from Kaggle. Dataset consists of tweet_id, sentiment, and tweet where tweet_id is the unique number for every observation. Data is pre-processed followed by extracting features from the pre-processed data. Features are extracted using n-grams procedure. Features are represented using Sparse vector and Dense vector.

In [4] 2023 Imane Lasri used sentiment analysis for Moroccan public universities from twitter using big data technologies. Collected data from twelve public Moroccan universities and pre-processed data to label them. 1798 tweets were taken using Twint[32], an open python library to extract tweets based on keywords.

In [5] 2022, Yili Wang used a few other approaches for sentiment analysis which includes probabilistic classifier, Linear classifier, rule-based classifier, hybrid approach and other approaches. It concludes that machine learning approaches are more efficient than other approaches. The goal here is to survey on the existing methods for sentiment analysis and it is achieved.

In [6] 2022, Masoud AminiMotlagh collected data, pre- processed it, detected and classified accordingly. This research paper included some of the machine learningtechniques like KNN, SVM, NAÏVE BAYES and Bagging. Various test splits are used for each technique to check the accuracy and how the classifiers are working in many cases. At 70% train test split is working efficiently and for voting technique 85.71 accuracy is extracted, which is the highest among all cases. Variations are also taken out from the different cases. Results are represented using a line graph.

In [7] 2022, Astha Modi analyzed how different techniques like LSTM, Naïve bayes,

decision tree and SVM are working. Accuracies are also recorded and future scopes of every technique also mentioned in it. Data is analyzed based on scenarios. Tweets are taken on IPL, OTT service providers like Netflix, amazon prime etc, Footwear companies like Nike and Adidas, Indian industries reliance and adani. Results and analysis are represented in pie charts.

In [8] 2019, Abdul Rasool took a case study on twittersentiment analysis for apparel brands. Firstly, data is pre-processed and cleaned. Two brands are taken namely Nike and Adidas. 54788 tweets are taken for Nike and 45062 tweets are taken for Adidas. Totally 99850 tweets are taken. In this research paper Naïve bayes classifier is used to analyse and obtain patterns from it. Results explains on the percentages of positive, negative and neutral from both brands.

In [9] 2019, Faizan followed a certain procedure which includes data collection, data pre-processing, feature extraction, model selection and model evaluation. API available for twitter is used to collect data from twitter. By using data preprocessing techniques, unimportant data like hashtags, @usernames is removed. Feature extraction is done by POS tagging and n-grams techniques. KNN technique is used for model selection where it classifies data into its target values. Finally, model is evaluated using confusion matrix.

In [10] 2019, Abdullah Alsaeedi also used classification techniques that are Naïve bayes, Maximum Entropy and Support Vector Machine. Differentiation of document level and sentence level sentiment analysis also addressed in this research paper. After training most of the data then, model is built. Remaining data is tested by labels and analysis is taken out from the results. Which is very import addition to the research

2.2 PROPOSED SYSTEM

Here in this study, we are trying to show how twitter data-based sentiment analysis is working using NLP techniques and Logistic regression for classifying the extracted data. Hence, this drives us to create and analyze the models on this data. Public tweets from twitter are taken out by manually or using API. Collected data is pre-processed using many NLP techniques like POS tagging, Stop word removal etc. After collecting pre-processed data, it is loaded into different machine learning models for classification. Lastly, observing results we can conclude how the models are working for tweets on tweets from twitter. This helps us to know the trends on some tweets and how people are responding for it. By this research we are analyzing different machine learning algorithms and how they are behaving for the given dataset of tweets.

Chapter – 3

DESIGN FLOW & PROCESS

3.1 Problem statement:

Develop an accurate and efficient sentiment analysis system specifically tailored for tweets to enable tweets owners to gain insights into customer sentiments, identify areas for improvement, and enhance the overall dining experience. The key challenges include handling domain-specific language, dealingwith mixed sentiments, considering context, scaling for large review volumes, and overcoming data sparsity. By addressing these challenges, the system aims to provide timely and valuable insights to support data-driven decision-making in the tweets industry.

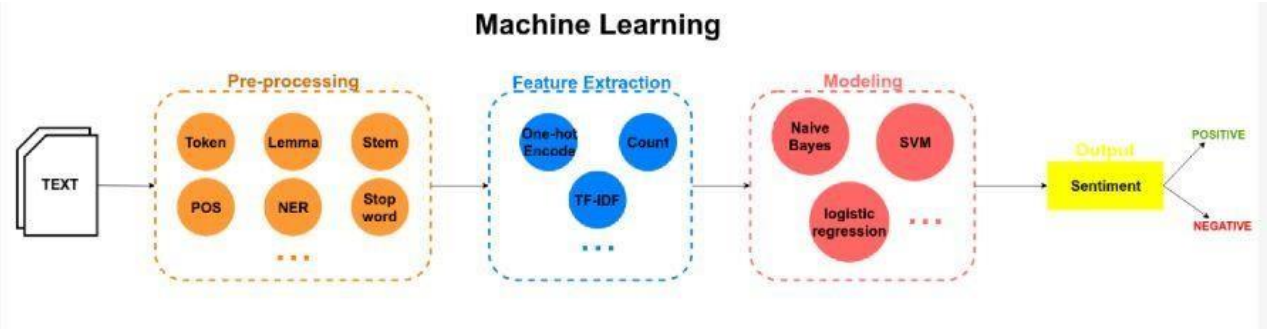


Fig1: Above, the implementation of sentiment analysis

3.2 Techniques and tools

a. Lexicon-based Approaches: Utilize sentiment lexicons or dictionaries to assign sentiment scores to words and aggregate them for sentiment analysis.
b. Machine Learning Algorithms: Apply supervised machine learning techniques like Naive Bayes, Support Vector Machines (SVM), or Random Forests for sentiment classification.

Text Preprocessing and Normalization:

a. Tokenization: Splitting text into individual words or tokens.
b. Stop word Removal: Eliminating common words (e.g., "a," "the," "is") that do not carry significant sentiment information.
c. Stemming and Lemmatization: Reducing words to their root forms to handle variations and improve feature extraction.
d. Noise Removal: Removing irrelevant characters, punctuation, or HTML tags from the text.

Feature Extraction:

a. Stemming: stemming is a process where the word is reduced to its root word.
b. n-grams: Capturing contextual information by considering sequences of words instead of individual words.
c. TF-IDF (Term Frequency-Inverse Document Frequency): Weighing the importance of words based on their frequency in the text corpus.

Evaluation and Performance Metrics:

a. Accuracy: Measures the overall correctness of sentiment classification.
b. Precision, Recall, and F1 Score: Evaluate the performance of sentiment classification models, considering true positives, false positives, and false negatives.

c. Cross-Validation: Assessing the generalization capability of the sentiment analysis model by splitting the data into multiple folds.

NLP Libraries and Tools:

Natural Language Toolkit (NLTK): A Python library providing various NLP functionalities such as tokenization, stemming, and sentiment analysis.

3.3 Theory

What is Sentiment Analysis?

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. However, analysis of social media streams is usually restricted to just basic sentiment analysis and count based metrics. This is akin to just scratching the surface and missing out on those high value insights that are waiting to be discovered. So what should a brand do to capture that low hanging fruit?

With the recent advances in deep learning, the ability of algorithms to analyses text has improved considerably. Creative use of advanced artificial intelligence techniques can be an effective tool for doing in-depth research. We believe it is important to classify incoming customer conversation about a brand based on following lines:

Key aspects of a brand's product and service that customers care about. User's underlying intentions and reactions concerning those aspects.

These basic concepts when used in combination, become a very important tool for analyzing millions of brand conversations with human level accuracy.

Text Classifier — The basic building blocks

Sentiment Analysis

Sentiment Analysis is the most common text classification tool that analyses an incoming message and tells whether the underlying sentiment is positive, negative our neutral. You can input a sentence of your choice and gauge the underlying sentiment by playing with the demo here.

Intent Analysis

Intent analysis steps up the game by analyzing the user's intention behind a message

and identifying whether it relates an opinion, news, marketing, complaint, suggestion, appreciation or query.

Contextual Semantic Search (CSS)

Now this is where things get really interesting. To derive actionable insights, it is important to understand what aspect of the brand is a user discussing about. For example: Amazon would want to segregate messages that related to: late deliveries, billing issues, promotion related queries, product etc. On the other hand, Starbucks would want to classify messages based on whether they relate to staff behavior, new coffee flavors, hygiene feedback, online orders, store name and location etc. But how can one do that?

We introduce an intelligent smart search algorithm called Contextual Semantic Search (a.k.a. CSS). The way CSS works is that it takes thousands of messages and a concept (like Price) as input and filters all the messages that closely match with the given concept. The graphic shown below demonstrates how CSS represents a major improvement over existing methods used by the industry.

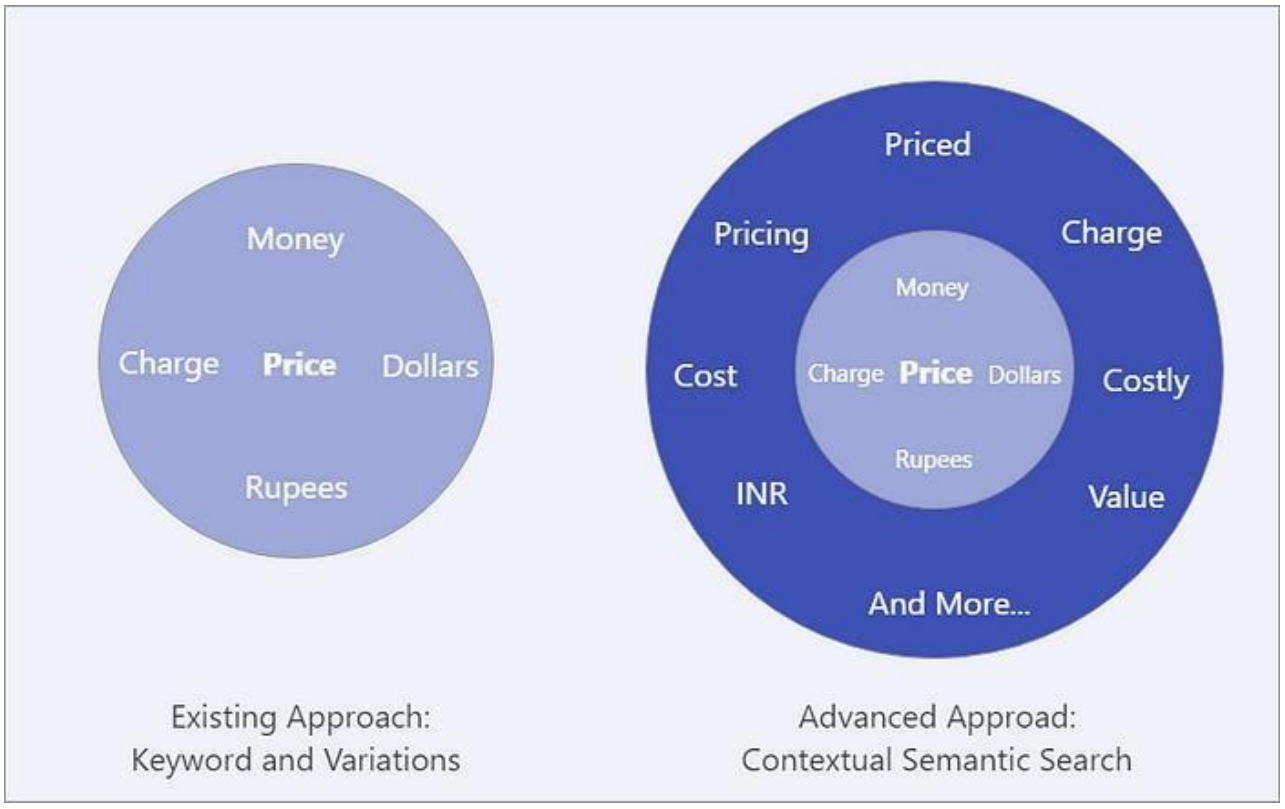


Fig2: Existing approach vs Contextual Semantic Search

A conventional approach for filtering all Price related messages is to do a keyword search on Price and other closely related words like (pricing, charge, $, paid). This method however is not very effective as it is almost impossible to think of all the relevant keywords and their variants that represent a particular concept. CSS on the other hand just takes the name of the concept (Price) as input and filters all the contextually similar even where the obvious variants of the concept keyword are not mentioned.

For the curious people, we would like to give a glimpse of how this works. An AI technique is used to convert every word into a specific point in the hyperspace and the distance between these points is used to identify messages where the context is similar to the concept we are exploring. A visualization of how this looks under the hood can be seen below:

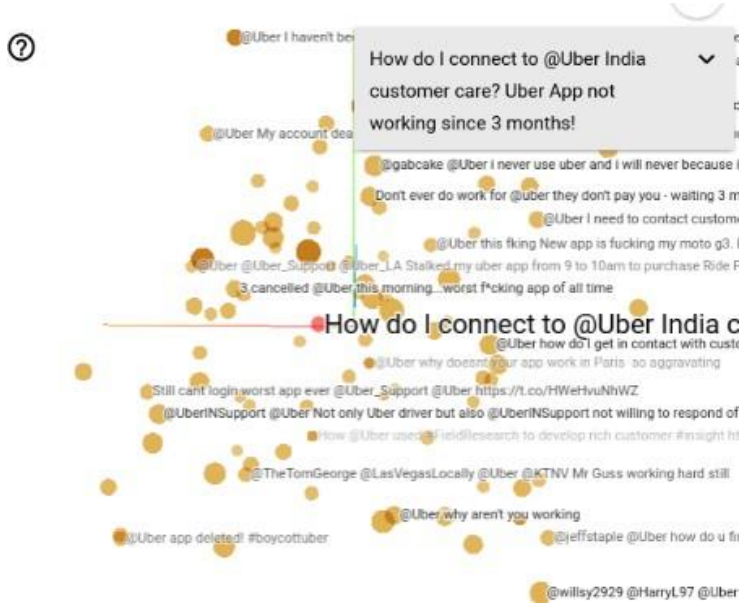


Fig3: Visualizing contextually related Tweets

Why perform Sentiment Analysis?

Sentiment analysis is the contextual meaning of words that indicates the social sentiment of a brand and also helps the business to determine whether the product they are manufacturing is going to make a demand in the market or not.

According to the survey,80% of the world's data is unstructured. The data needs to be analyzed and be in a structured manner whether it is in the form of emails, texts, documents, articles, and many more.

Sentiment Analysis is required as it stores data in an efficient, cost-friendly. Sentiment analysis solves real-time issues and can help you solve all real-time scenarios.

Types of Sentiment Analysis:

Fine-grained sentiment analysis: This depends on the polarity base. This category can be designed as very positive, positive, neutral, negative, or very negative. The rating is done on a scale of 1 to 5. If the rating is 5 then it is very positive, 2 then negative, and 3 then neutral.

Emotion detection: The sentiments happy, sad, angry, upset, jolly, pleasant, and so on come under emotion detection. It is also known as a lexicon method of sentiment analysis.

Aspect-based sentiment analysis: It focuses on a particular aspect for instance if a person wants to check the feature of the cell phone then it checks the aspect such as the battery, screen, and camera quality then aspect based is used.

Multilingual sentiment analysis: Multilingual consists of different languages where the classification needs to be done as positive, negative, and neutral. This is highly challenging and comparatively difficult.

How does Sentiment Analysis work?

There are three approaches used:

Rule-based approach: Over here, the lexicon method, tokenization, and parsing come in the rule-based. The approach is that counts the number of positive and negative words in the given dataset. If the number of positive words is greater than the number of negative words then the sentiment is positive else vice-versa.

Machine Learning Approach: This approach works on the machine learning technique. Firstly, the datasets are trained and predictive analysis is done. The next process is the extraction of words from the text is done. This text extraction can be done using different techniques such as Naive Bayes, Support Vector machines, hidden Markov model, and conditional random fields like this machine learning techniques are used.

Neural network Approach: In the last few years neural networks have evolved at a very rate. It involves using artificial neural networks, which are inspired by the structure of the human brain, to classify text into positive, negative, or neutral sentiments. it has Recurrent neural networks, Long short-term memory, Gated recurrent unit, etc to process sequential data like text.

Hybrid Approach: It is the combination of two or more approaches i.e. rule-based and Machine Learning approaches. The surplus is that the accuracy is high compared to the other two approaches.

Applications:

Sentiment Analysis has a wide range of applications as:

Social Media: If for instance the comments on social media side as Instagram, over here all the are analyzed and categorized as positive, negative, and neutral.
Customer Service: In the play store, all the comments in the form of 1 to 5 are done with the help of sentiment analysis approaches.
Marketing Sector: In the marketing area where a particular product needs to be reviewed as good or bad.
Reviewer side: All the reviewers will have a look at the comments and will check and give the overall review of the product.

Challenges of Sentiment Analysis
There are major challenges in the sentiment analysis approach:
If the data is in the form of a tone, then it becomes really difficult to detect whether the comment is pessimist or optimistic.
If the data is in the form of emoji, then you need to detect whether it is good or bad.
Even the ironic, sarcastic, comparing comments detection is really hard.
Comparing a neutral statement is a big task.

What is Machine Learning?

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by

Arthur Samuel in 1959.

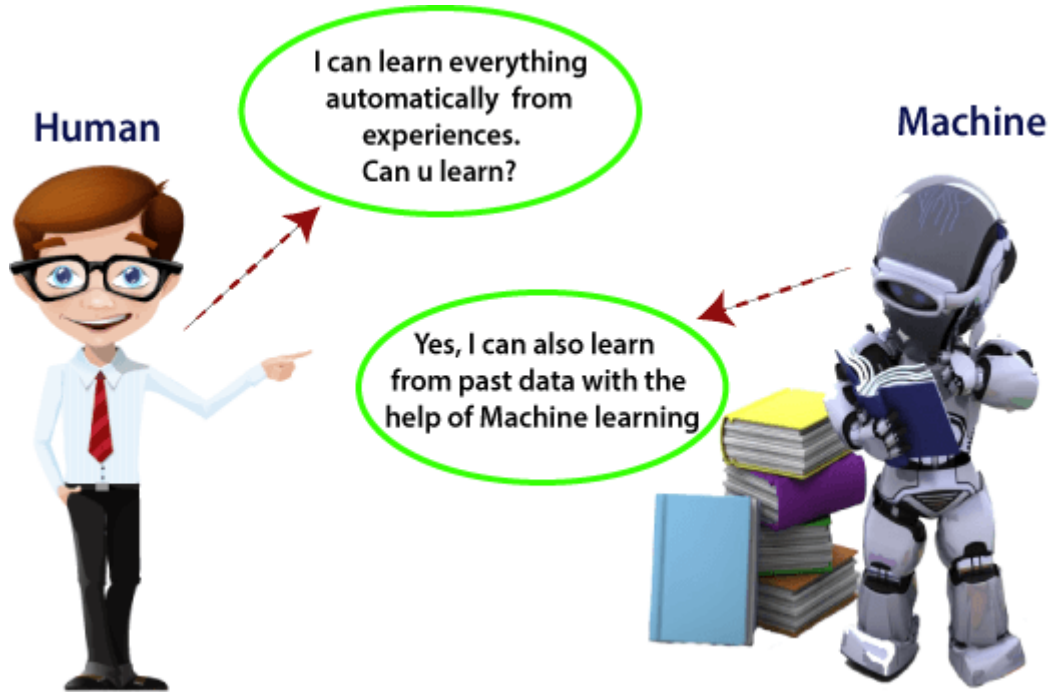


Fig4: Machine Learning overview

Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed.

How does Machine Learning work:

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms, machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:

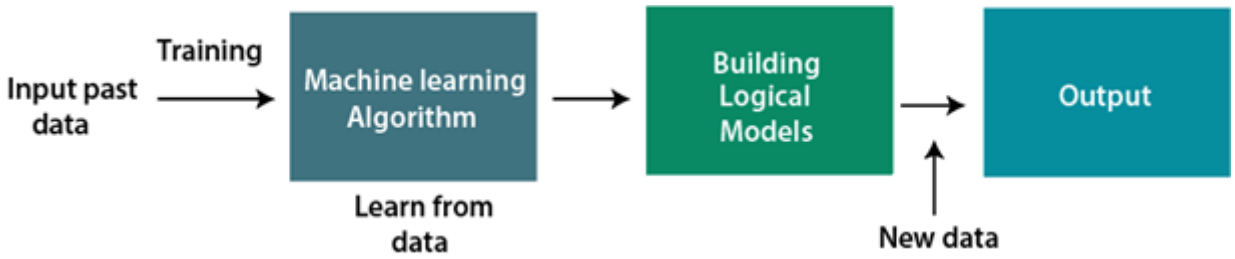


Fig5: Above, the machine learning process

Features of Machine Learning:

- Machine learning uses data to detect various patterns in a given dataset.
- It can learn from past data and improve automatically.
- It is a data-driven technology.
- Machine learning is much similar to data mining as it also deals with the huge amount of the data.

What is Natural Language Processing?

Fig6: What is NLP

Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software.
The study of natural language processing has been around for more than 50 years and grew out of the field of linguistics with the rise of computers.

Natural Language Processing (NLP) refers to AI method of communicating with an intelligent system using a natural language such as English.

Processing of Natural Language is required when you want an intelligent system like robot to perform as per your instructions, when you want to hear decision from a dialogue based clinical expert system, etc.

The field of NLP involves making computers to perform useful tasks with the natural language's humans use. The input and output of an NLP system can be −

- Speech
- Written Text

Components of NLP:

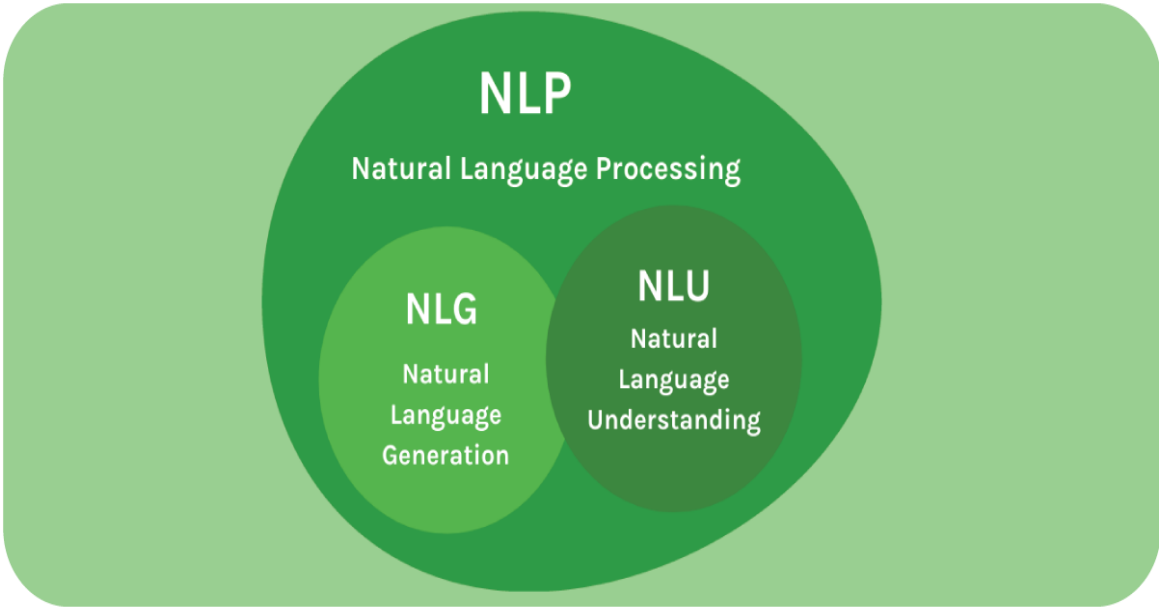


Fig7: NLP VS NLU VS NLG

There are two components of NLP as given −

Natural Language Understanding (NLU)
Understanding involves the following tasks −
Mapping the given input in natural language into useful representations.
Analyzing different aspects of the language.
Natural Language Generation (NLG)

It is the process of producing meaningful phrases and sentences in the form of natural language from some internal representation.
It involves −
Text planning − It includes retrieving the relevant content from knowledge base.
Sentence planning − It includes choosing required words, forming meaningful phrases, setting tone of the sentence.
Text Realization − It is mapping sentence plan into sentence structure.
The NLU is harder than NLG.

Difficulties in NLU:

- NL has an extremely rich form and structure.
- It is very ambiguous. There can be different levels of ambiguity –

Lexical ambiguity − It is at very primitive level such as word-level.
For example, treating the word “board” as noun or verb?
Syntax Level ambiguity − A sentence can be parsed in different ways.
For example, “He lifted the beetle with red cap.” − Did he use cap to lift the beetle or he lifted a beetle that had red cap?
Referential ambiguity − Referring to something using pronouns. For example, Rima went to Gauri. She said, “I am tired.” − Exactly who is tired?
One input can mean different meanings.
Many inputs can mean the same thing.

NLP Terminology

- Phonology − It is study of organizing sound systematically.
- Morphology − It is a study of construction of words from primitive meaningful units.
- Morpheme − It is primitive unit of meaning in a language.
- Syntax − It refers to arranging words to make a sentence. It also involves determining the structural role of words in the sentence and in phrases.
- Semantics − It is concerned with the meaning of words and how to combine words into meaningful phrases and sentences.
- Pragmatics − It deals with using and understanding sentences in different situations and how the interpretation of the sentence is affected.
- Discourse − It deals with how the immediately preceding sentence can affect the interpretation of the next sentence.
- World Knowledge − It includes the general knowledge about the world.

Steps in NLP

There are general five steps −

Lexical Analysis − It involves identifying and analyzing the structure of words. Lexicon of a language means the collection of words and phrases in a language. Lexical analysis is dividing the whole chunk of txt into paragraphs, sentences, and words.

Syntactic Analysis (Parsing) − It involves analysis of words in the sentence for grammar and arranging words in a manner that shows the relationship among the words. The sentence such as “The school goes to boy” is rejected by English syntactic analyzer.

NLP Steps

- Semantic Analysis − It draws the exact meaning or the dictionary meaning from the text. The text is checked for meaningfulness. It is done by mapping syntactic structures and objects in the task domain. The semantic analyzer disregards sentence such as “hot ice-cream”.

- Discourse Integration − The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.

- Pragmatic Analysis − During this, what was said is re-interpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

What is Data Preprocessing?

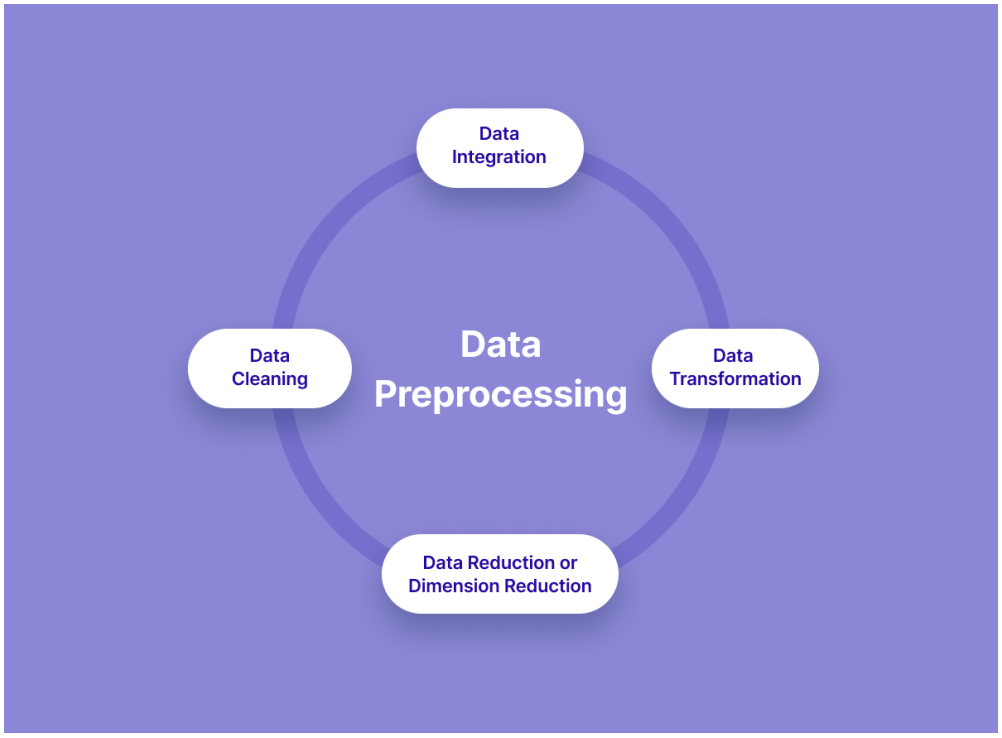


Fig8: Data pre-processing

Data preprocessing refers to the set of techniques and operations applied to raw data before it is used for analysis or modeling. It involves transforming, cleaning, and organizing the data to ensure its quality, consistency, and suitability for further processing. Data preprocessing plays a crucial role in data analysis and machine

learning tasks, as it helps improve the accuracy, reliability, and efficiency of subsequent data processing steps.

The main steps involved in data preprocessing are as follows:

Data Cleaning:
Handling missing data: Dealing with missing values by either imputing them or removing rows or columns with missing data.
Removing duplicates: Identifying and eliminating duplicate records or instances from the dataset.
Handling outliers: Detecting and handling outliers or anomalies that may significantly affect the analysis or modeling results.

Data Transformation:
Feature scaling: Normalizing or standardizing numeric features to bring them to a similar scale and prevent biases in certain algorithms.
Feature encoding: Converting categorical variables into numerical representations that machine learning algorithms can process.
Feature discretization: Grouping continuous variables into discrete bins or intervals to simplify the data representation.
Feature engineering: Creating new features or transforming existing features to better represent the underlying patterns or relationships in the data.
Data Integration:
Combining data from multiple sources or different datasets into a unified format for analysis or modeling.
Resolving data inconsistencies, such as conflicting attribute names or data formats, during the integration process.

Data Reduction:
Dimensionality reduction: Reducing the number of features or variables while preserving the most important information, typically achieved through techniques like Principal Component Analysis (PCA) or feature selection algorithms.
Instance sampling: Selecting a representative subset of instances or records from a large dataset to reduce computational complexity or balance class distributions.

Data Formatting:

Ensuring the data is in the appropriate format and structure for analysis or modeling tasks.
Handling date and time formats, converting text to lowercase or uppercase, or adjusting data representations to match specific requirements.

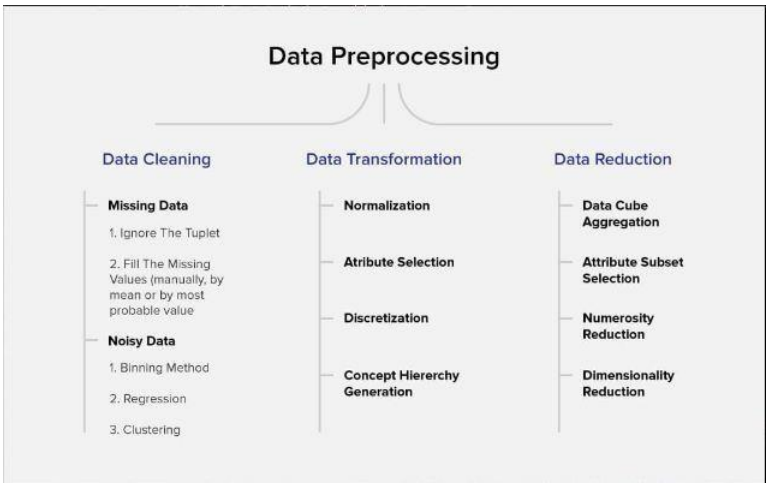


Fig9. Steps of Data Pre-processing

Tokenization:

Tokenization is the process of breaking down a text or document into smaller units called tokens. In the context of natural language processing (NLP), tokens typically represent words, but they can also be characters, subwords, or other meaningful units of text. Tokenization is an essential step in many NLP tasks as it helps to analyze and process text data more effectively.

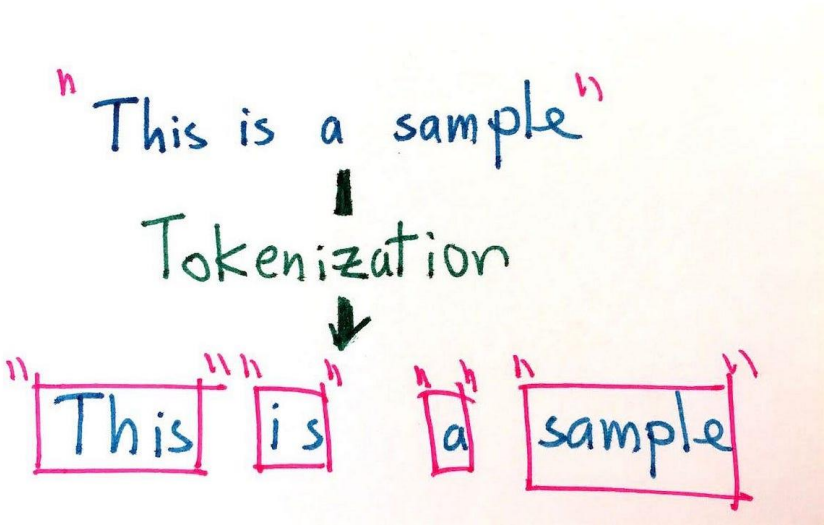


Fig10: Tokenization

The process of tokenization involves the following steps:

Text Segmentation: The text is divided into individual segments or chunks. The size and definition of these segments depend on the specific tokenization method used.

Word Tokenization: The most common form of tokenization is word tokenization, where the text is divided into words or word-like units. Each word in the text is treated as a separate token.

Token Boundaries: The tokenization process identifies boundaries between tokens based on certain rules or patterns. Common tokenization rules include splitting text at whitespace, punctuation marks, or special characters.

Handling Special Cases: Tokenization may involve handling special cases like contractions, hyphenated words, abbreviations, or compound words. For example, "can't" may be tokenized as "can" and "'t," while "New York" may be tokenized as "New" and "York."

Tokenization is a fundamental step in various NLP applications, including text classification, sentiment analysis, named entity recognition, part-of-speech tagging, and machine translation. It helps to convert unstructured text data into structured and manageable units, enabling subsequent analysis or processing tasks to operate on a more granular level.

Tokenization is typically performed using pre-built tokenization libraries or tools available in programming languages such as Python. Libraries like NLTK (Natural Language Toolkit), spaCy, or the tokenization functionalities provided by deep learning frameworks like TensorFlow or PyTorch offer convenient and efficient methods for tokenizing text data.

POS Tagging:
POS tagging, or Part-of-Speech tagging, is the process of assigning grammatical tags to words in a text corpus based on their roles and relationships within a sentence. POS tags represent the syntactic category or part of speech of each word, such as noun,

verb, adjective, adverb, pronoun, preposition, conjunction, or interjection.

POS tagging is an essential task in natural language processing (NLP) as it helps in understanding the grammatical structure of sentences, disambiguating word meanings, and providing contextual information for downstream NLP tasks.

The process of POS tagging typically involves the following steps:

Tokenization: The text is first tokenized into individual words or tokens.

Lexical Lookup: Each token is looked up in a pre-defined dictionary or lexicon that contains words and their associated POS tags. The lexicon may also include additional information such as word forms, lemma, or frequency.

Contextual Disambiguation: In cases where a word has multiple possible POS tags (ambiguity), the context of the sentence is considered to determine the most appropriate tag. This is often done using statistical models, rule-based approaches, or machine learning algorithms.

Tagging the Tokens: POS tags are assigned to each token, indicating the part of speech it represents in the sentence. For example, "dog" may be tagged as a noun (NN), "run" as a verb (VB), or "beautiful" as an adjective (JJ).

Common POS tagsets include the Penn Treebank tagset, Universal POS tagset, or Brown Corpus tagset, among others. Each tagset has its own set of tag labels and conventions for representing different parts of speech.

POS tagging is often performed using pre-trained models and libraries in popular programming languages such as Python. Libraries like NLTK, spaCy, or CoreNLP provide POS tagging functionalities, allowing developers and researchers to perform accurate and efficient POS tagging on their text data.

POS tagging serves as a crucial preprocessing step in various NLP tasks, including text parsing, information extraction, sentiment analysis, machine translation, and text-to-speech synthesis. It enhances the understanding and analysis of textual data by providing insights into the grammatical structure and syntactic relationships within sentences.

Dependency Parsing:
Dependency parsing is a natural language processing (NLP) technique that analyzes the grammatical structure and syntactic relationships between words in a sentence. It

aims to determine the hierarchical dependencies between words and represents them as a directed graph called a dependency tree.

In dependency parsing, each word in the sentence is considered a node in the tree, and the dependencies between the words are represented as labeled edges connecting the nodes. The dependencies typically indicate the syntactic relationships, such as subject-verb, verb-object, modifier-noun, or conjunction relationships.

The process of dependency parsing involves the following steps:

Tokenization: The input sentence is tokenized into individual words or tokens.

Part-of-Speech (POS) Tagging: Each token is assigned a grammatical tag indicating its part of speech (noun, verb, adjective, etc.) using techniques like POS tagging.

Dependency Parsing Algorithm: A dependency parsing algorithm is applied to analyze the relationships between words and construct the dependency tree. Commonly used algorithms include transition-based approaches (e.g., Arc-Standard, Arc-Eager) or graph-based approaches (e.g., Minimum Spanning Tree, Maximum spanning Tree).

Dependency Labeling: The edges in the dependency tree are assigned labels that represent the syntactic relationship between the connected words. For example, labels like "nsubj" (nominal subject), "dobj" (direct object), or "amod" (adjectival modifier) indicate different types of dependencies.

Dependency parsing provides valuable insights into the syntactic structure of sentences and is used in various NLP applications, including information extraction, question answering, sentiment analysis, machine translation, and text summarization. It helps in understanding the relationships between words and identifying the main components and modifiers in a sentence.

There are several libraries and tools available for performing dependency parsing, such as Stanford CoreNLP, spaCy, NLTK, or the Universal Dependencies project. These tools often provide pre-trained models and APIs that allow developers and researchers to perform accurate and efficient dependency parsing on their text data.

Sentiment Analysis:
Sentiment analysis, also known as opinion mining, is a natural language processing (NLP) technique that aims to determine the sentiment or subjective information expressed in text. It involves analyzing and classifying the emotions, attitudes,

opinions, or sentiments conveyed by individuals in their written expressions, such as , social media posts, customer feedback, or survey responses.

The primary goal of sentiment analysis is to automatically identify and extract sentiment polarity (positive, negative, or neutral) from text data, allowing for a quantitative understanding of people's opinions or sentiments towards specific topics, products, services, or events.

The process of sentiment analysis typically involves the following steps:

Text Preprocessing: The text data is cleaned, normalized, and preprocessed to remove noise, irrelevant information, or special characters. This step often includes processes like tokenization, stopword removal, and stemming or lemmatization.

Sentiment Lexicon Creation: A sentiment lexicon or dictionary is compiled, containing a list of words or phrases along with their associated sentiment scores or labels. These scores indicate the sentiment polarity of each word, such as positive, negative, or neutral. The lexicon may also include additional information, such as intensifiers or negation words.

Sentiment Classification:
a. Rule-based Approaches: These approaches use predefined rules or patterns to match words or phrases from the text data to the sentiment lexicon and assign sentiment labels based on the matches and their context.
b. Machine Learning Algorithms: Supervised machine learning algorithms, such as Naive Bayes, Support Vector Machines (SVM), or Recurrent Neural Networks (RNNs), are trained on labeled sentiment data to classify the sentiment of new, unseen text based on learned patterns and features.

Sentiment Aggregation: The sentiment scores or labels assigned to individual words or phrases are aggregated to determine an overall sentiment score for the entire text. Aggregation methods can include simple summation, weighted averages, or more complex techniques like sentiment propagation in dependency trees.

Evaluation and Validation: The performance of the sentiment analysis model is evaluated using metrics such as accuracy, precision, recall, F1 score, or confusion matrix. Validation techniques like cross-validation or holdout evaluation are commonly used to ensure the model's generalization capability.

Sentiment analysis finds applications in various domains, including customer feedback analysis, social media monitoring, brand reputation management, market

research, and personalized recommendation systems. It provides valuable insights into public opinion, customer sentiment, and user feedback, enabling businesses and organizations to make data-driven decisions, understand customer needs, and enhance user experiences.

Classification:

Classification is a machine learning task that involves assigning predefined categories or labels to input data based on their features or characteristics. It is a supervised learning technique where the model learns from labeled training data to make predictions or classify new, unseen instances into predefined classes.

The process of classification typically involves the following steps:

Data Preparation: The input data is prepared in a suitable format, often represented as feature vectors. This may involve data cleaning, normalization, feature extraction, or feature engineering techniques.

Training Data Creation: A labeled training dataset is created, consisting of instances with known class labels. The dataset is divided into input features (independent variables) and corresponding class labels (dependent variable).

Model Selection: A classification algorithm or model is selected based on the problem domain, available data, and desired performance metrics. Commonly used algorithms include Decision Trees, Random Forests, Support Vector Machines (SVM), Naive Bayes, Logistic Regression, or Neural Networks.

Model Training: The selected classification model is trained using the labeled training data. The model learns the patterns and relationships between input features and their corresponding class labels, aiming to minimize prediction errors.

Model Evaluation: The trained model is evaluated using evaluation metrics such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic curve (AUC-ROC). Evaluation helps assess the model's performance and generalization capability on unseen data.

Model Optimization and Tuning: The model's hyperparameters and settings are fine-tuned to improve its performance. Techniques such as cross-validation, grid search, or random search can be applied to find the optimal combination of hyperparameters.

Prediction or Inference: Once the classification model is trained and evaluated, it can be used to make predictions or classify new, unseen instances into the predefined classes. The model applies the learned patterns to the input features and assigns the most probable class label to each instance.

Classification is widely used in various applications, including image recognition, text classification, spam filtering, sentiment analysis, customer segmentation, fraud detection, medical diagnosis, and many more. It enables automated decision-making and pattern recognition, allowing systems to classify and organize data efficiently based on their inherent characteristics or properties.

Naïve Bayes:

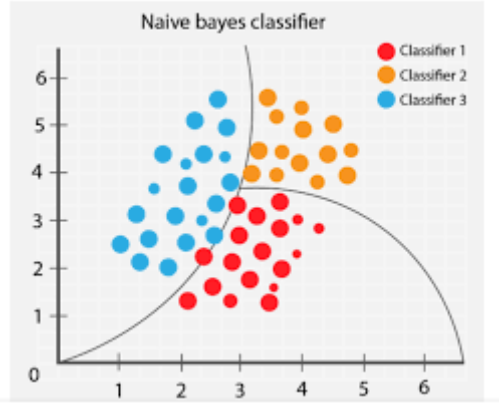


Fig11: NAÏVE BAYES

Naive Bayes is a simple yet powerful classification algorithm based on Bayes' theorem of probability. It is a supervised learning algorithm that is commonly used for text classification and spam filtering tasks. Despite its simplicity, Naive Bayes often performs well in practice and is computationally efficient.

The underlying principle of Naive Bayes is that it assumes independence between the features (or attributes) given the class. This assumption is referred to as "naive" because it simplifies the computation by considering each feature independently, disregarding any potential correlations between them.
Naive Bayes is known for its simplicity, fast training speed, and ability to handle large feature spaces. However, the independence assumption can be a limitation when there are strong dependencies among the features. Nevertheless, Naive Bayes is widely used in various applications such as text classification, email filtering,
sentiment analysis, and recommendation systems.

Naive Bayes' Rule: When classifying a new instance, Naive Bayes applies Bayes' theorem to calculate the probability of each class given the feature values. The class with the highest probability is chosen as the predicted class label.

Bayes' Theorem: Bayes' theorem calculates the posterior probability of a class given the feature values using the prior probability of the class and the likelihood of the features given the class. It can be expressed as:

P(class|features) = (P(class) * P(features|class)) / P(features)
P(class|features) represents the posterior probability of the class given the features.
P(class) is the prior probability of the class.
P(features|class) is the likelihood of the features given the class.
P(features) is the probability of the features (constant across classes), which can be calculated using the law of total probability.

The algorithm calculates the posterior probability for each class and assigns the class with the highest probability as the predicted class label for the new instance.

Naive Bayes

It is one of the popular classification techniques of algorithms used in data mining. It is a probability classifier. It links the attributes mutually & is dependent on the number of parameters, The principle here is that the variables provided are independent. It generates accurate results with appropriate calculation & provides fast results. It is on Bayes theorem & the formula is,

$$P(\text{label} \mid \text{features}) = \frac{P(\text{label}) * P(\text{features} \mid \text{label})}{P(\text{features})}$$

SVM:

Support Vector Machines (SVM) is a popular supervised machine learning algorithm used for classification and regression tasks. It is known for its ability to handle high-dimensional feature spaces and handle both linearly separable and non-linearly separable data.

The main idea behind SVM is to find an optimal hyperplane that separates the data points belonging to different classes with the largest possible margin. The hyperplane is chosen such that it maximizes the distance between the nearest data points of different classes, which are called support vectors.

Data Preparation: The input data is typically represented as a feature matrix, where each row represents an instance, and each column represents a feature. The features can be continuous or discrete.

Feature Scaling: It is common practice to scale or normalize the feature values to ensure they have a similar range. This helps in preventing any particular feature from dominating the learning process.

Training: During the training phase, SVM learns a hyperplane that separates the data points of different classes with the largest margin. SVM aims to find the optimal hyperplane by solving a quadratic optimization problem.

Hyperplane Selection: SVM can use different types of hyperplanes depending on the data and problem at hand:

Linear SVM: In linear SVM, a linear hyperplane is used to separate the data points.
Non-linear SVM: For data that is not linearly separable, SVM can use kernel functions (e.g., polynomial, radial basis function) to map the data into a higher-dimensional

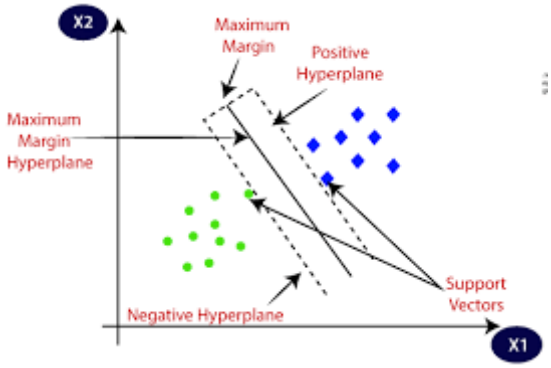


Fig12: SVM

space, where a linear hyperplane can separate the transformed data.
Margin Maximization: SVM strives to maximize the margin between the hyperplane and the support vectors. The margin is the distance between the hyperplane and the closest data points from each class.

CountVectorizer:
CountVectorizer is a text feature extraction technique in natural language processing (NLP) that converts a collection of text documents into a matrix of token counts. It is a commonly used method to preprocess text data before applying machine learning algorithms, particularly for tasks such as text classification, clustering, and information retrieval.

The CountVectorizer operates by following these steps:

- Tokenization: The text documents are split into individual words or tokens. This process can involve removing punctuation, lowercasing, and applying other text

preprocessing techniques.

- Vocabulary Construction: The CountVectorizer builds a vocabulary, which is essentially a dictionary that maps each unique token to an index. The vocabulary is created by collecting all the unique tokens from the corpus of documents.

- Counting Token Occurrences: For each document in the corpus, the CountVectorizer counts the number of occurrences of each token in the vocabulary. It creates a matrix where each row represents a document, and each column represents a token in the vocabulary. The value in each cell indicates the count of the corresponding token in the respective document.

- Transforming Text into a Matrix: The CountVectorizer transforms the text documents into a sparse matrix representation, where most of the entries are zero, as most tokens do not occur in most documents. This sparse matrix format is memory-efficient for large corpora.

- The resulting matrix, often referred to as the "document-term matrix," represents the text data in a numerical format that can be inputted into machine learning models. Each cell in the matrix represents the frequency or count of a token in a particular document.

CountVectorizer also provides additional functionalities, such as:

- Handling stop words: It can ignore common words, known as stop words, during the tokenization process to reduce noise in the resulting matrix.
- n-gram support: It can consider sequences of contiguous tokens of length n (unigrams, bigrams, trigrams, etc.) instead of individual tokens, capturing more contextual information.
- Vocabulary size control: It allows limiting the vocabulary size based on the most frequent or least frequent tokens.

Overall, CountVectorizer is a widely used technique for converting text data into a numerical representation suitable for machine learning algorithms, making it a fundamental step in many NLP pipelines.

Classification: To classify new instances, SVM determines on which side of the hyperplane the instance falls. If it lies on the positive side, it is classified as one class,and if it lies on the negative side, it is classified as the other class.

SVM has various advantages, including its effectiveness in handling high-dimensional feature spaces, ability to handle non-linearly separable data through kernel functions, and resistance to overfitting. It is widely used in applications such as text categorization, image classification, bioinformatics, and finance.
However, SVMs can be sensitive to the choice of hyperparameters and the scaling of input features. Additionally, SVMs can be computationally expensive for large datasets. Proper parameter tuning and careful preprocessing are crucial for obtaining optimal results with SVMs.

Fig13: Classification

3.4 METHODOLOGY

This proposed work is to predict the text automatically based on the data set values stored by using the r tool. By using the training data set values, it is possible to predict the text data using our classifier called naive bayes using algorithm.
The Fig I depicts the architecture of the proposed model used in the prediction of sentiment analysis. It consists of 3 steps

A. Data Collection

In this step data is taken out from Kaggle in a recognized format. Missing fields are evacuated in this process & thus the data is transformed. Sentiment Analysis can be considered a classification process. There are three main classification levels in sentiment analysis document-level, sentence-level, and aspect-level sentiment analysis. Level of document it aims to classify an opinion document which as a positive or negative opinion expression. It considers the full document as a basic information unit.

B. Data Pre-processing

The collected raw data of tweets consist of large number of attributes and there will be missing values. The reducing the attributes is required, extracting the attributes is also much essential. So, in order to meet importance of each variable or attributes "migrittr" algorithm is applied. Migrittr algorithm which selects the attributes based on predictor, here predictor considered tweets review. Feature or Attribute extraction is done using migrittr algorithm. In detail steps working of migrittralgorithm. In Data cleaning once attributes are removed, filling the missing values, removing inconsistent data measuring the central tendency for the attribute such as mean median, quartile is done. In data pre-process the data is cleaned and the extracteddata before analysis. Non-textual contents and contents that are irrelevant for the analysis are identified and eliminated.

C. Sentiment Analysis

The sources are mainly review sites. Sentiment analysis is not only applied on product but can also applied on stock market, news articles, or political debates. In political debates for example, we could figure out people's opinions on a certain election candidates or political parties. The election results can also be from political posts. The sites like social media and micro
blogging sites are taken a very good source of information because many people share and discuss their opinions about positive and negative opinion freely.

D. Classification

The lexicon-based approach is to finding the opinion mining which is used to analyze or to predict the text. There are two methods in this approach. The dictionary- based approach which depends on finding opinion seed words, and then searches the dictionary of their synonyms and antonyms. The corpus-based approach begins with a seed list of opinion words, and then finds other opinion words in a large corpus to help in finding opinion words with context specific orientations. This could be done by using statistical or semantic methods.

Data mining has got two most frequent modelling goals — classification & prediction. Classification model classifies discrete, unordered values or data. In this prediction process, the classification techniques utilized are, naive bayes classifier.

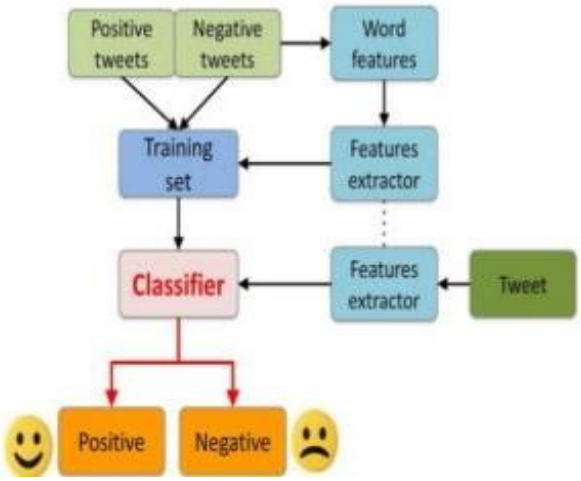


Fig14. Process of Sentiment analysis

Chapter – 4

IMPLEMENTATION SNAPSHOTS OF SOURCE CODE

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import string
import nltk
import warnings
%matplotlib inline

warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv(r"https://raw.githubusercontent.com/aswintechguy/Machine-Learning-Projects/master/Twitter%20Sentiment%20Anal
df.head()
```

Out[2]:

	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

Fig15: implementation 1

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31962 entries, 0 to 31961
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      31962 non-null  int64
 1   label   31962 non-null  int64
 2   tweet   31962 non-null  object
dtypes: int64(2), object(1)
memory usage: 749.2+ KB
```

In [4]:
```
def remove_pattern(input_txt, pattern):
    r = re.findall(pattern, input_txt)
    for word in r:
        input_txt = re.sub(word, "", input_txt)
    return input_txt
```

In [5]:
```
df['clean_tweet'] = np.vectorize(remove_pattern)(df['tweet'], "@[\w]*")
```

Fig16: implementation 2

In [6]:
```
df.head()
```

Out[6]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can't use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide: society now #motivation

In [7]:
```
df['clean_tweet'] = df['clean_tweet'].str.replace("[^a-zA-Z#]", " ")
df.head()
```

Out[7]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when a father is dysfunctional and is so sel...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks for #lyft credit i can t use cause th...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model i love u take with u all the time in ...
4	5	0	factsguide: society now #motivation	factsguide society now #motivation

Fig17: implementation 3

In [8]:
```
df['clean_tweet'] = df['clean_tweet'].apply(lambda x: " ".join([w for w in x.split() if len(w)>3]))
df.head()
```

Out[8]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when father dysfunctional selfish drags kids i...
1	2	0	@user @user thanks for #lyft credit i can't us...	thanks #lyft credit cause they offer wheelchai...
2	3	0	bihday your majesty	bihday your majesty
3	4	0	#model i love u take with u all the time in ...	#model love take with time
4	5	0	factsguide: society now #motivation	factsguide society #motivation

In [9]:
```
tokenized_tweet = df['clean_tweet'].apply(lambda x: x.split())
tokenized_tweet.head()
```

Out[9]:
```
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: clean_tweet, dtype: object
```

In [10]:
```
from nltk import LancasterStemmer
stemmer = LancasterStemmer()

tokenized_tweet = tokenized_tweet.apply(lambda sentence: [stemmer.stem(word) for word in sentence])
tokenized_tweet.head()
```

Out[10]:
```
0    [when, fath, dysfunct, self, drag, kid, into, ...
1    [thank, #lyft, credit, caus, they, off, wheelc...
2                                [bihday, yo, majesty]
3                       [#model, lov, tak, with, tim]
4                      [factsguid, socy, #motivation]
Name: clean_tweet, dtype: object
```

Fig18: implementation 4

In [11]:
```
for i in range(len(tokenized_tweet)):
    tokenized_tweet[i] = " ".join(tokenized_tweet[i])

df['clean_tweet'] = tokenized_tweet
df.head()
```

Out[11]:

	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	when fath dysfunct self drag kid into dysfunct...
1	2	0	@user @user thanks for #lyft credit i can't us...	thank #lyft credit caus they off wheelchair va...
2	3	0	bihday your majesty	bihday yo majesty
3	4	0	#model i love u take with u all the time in ...	#model lov tak with tim
4	5	0	factsguide: society now #motivation	factsguid socy #motivation

In [12]:
```
pip install wordcloud
```

```
Requirement already satisfied: wordcloud in c:\users\vicky\anaconda3\lib\site-packages (1.8.2.2)
Requirement already satisfied: pillow in c:\users\vicky\anaconda3\lib\site-packages (from wordcloud) (8.2.0)
Requirement already satisfied: matplotlib in c:\users\vicky\anaconda3\lib\site-packages (from wordcloud) (3.3.4)
Requirement already satisfied: numpy>=1.6.1 in c:\users\vicky\anaconda3\lib\site-packages (from wordcloud) (1.20.1)
Requirement already satisfied: cycler>=0.10 in c:\users\vicky\anaconda3\lib\site-packages (from matplotlib->wordcloud) (0.1
0.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\users\vicky\anaconda3\lib\site-packages (from
matplotlib->wordcloud) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\vicky\anaconda3\lib\site-packages (from matplotlib->wordclou
d) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\vicky\anaconda3\lib\site-packages (from matplotlib->wordcloud)
(1.3.1)
Requirement already satisfied: six in c:\users\vicky\anaconda3\lib\site-packages (from cycler>=0.10->matplotlib->wordcloud)
(1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

Fig19: implementation 5

In [13]:
```
all_words = " ".join([sentence for sentence in df['clean_tweet']])

from wordcloud import WordCloud
wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

Fig20: implementation 6

In [14]:

```
all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==0]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

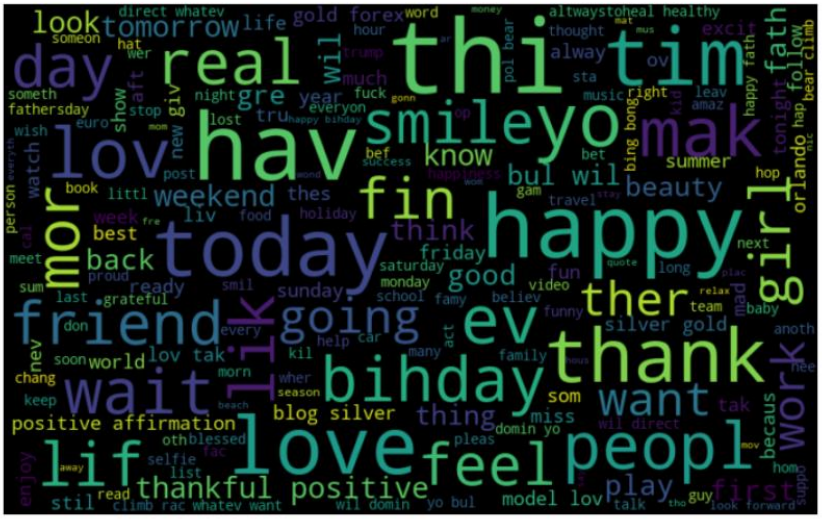


Fig21: implementation 7

In [15]:

```
all_words = " ".join([sentence for sentence in df['clean_tweet'][df['label']==1]])

wordcloud = WordCloud(width=800, height=500, random_state=42, max_font_size=100).generate(all_words)

# plot the graph
plt.figure(figsize=(15,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

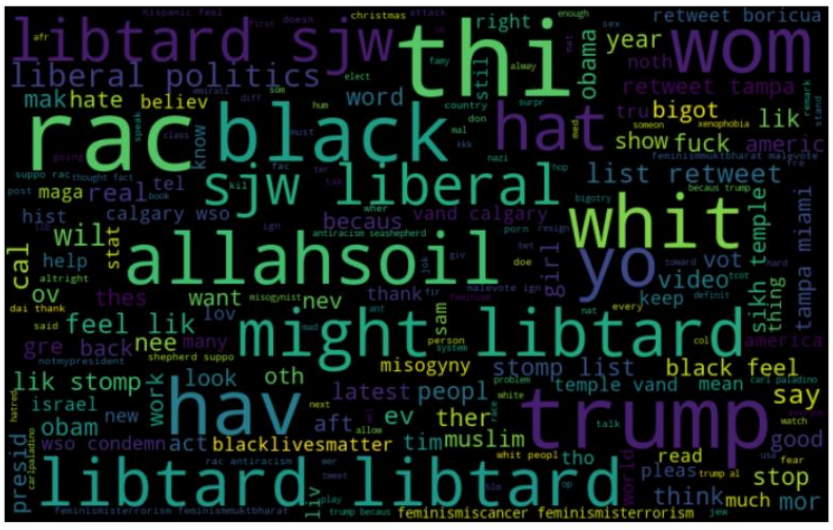


Fig22: implementation 8

In [16]:
```
def hashtag_extract(tweets):
    hashtags = []
    # loop words in the tweet
    for tweet in tweets:
        ht = re.findall(r"#(\w+)", tweet)
        hashtags.append(ht)
    return hashtags
```

In [17]:
```
ht_positive = hashtag_extract(df['clean_tweet'][df['label']==0])

# extract hashtags from racist/sexist tweets
ht_negative = hashtag_extract(df['clean_tweet'][df['label']==1])
```

In [18]:
```
ht_positive[:5]
```

Out[18]:
```
[['run'], ['lyft', 'disapointed', 'getthanked'], [], ['model'], ['motivation']]
```

In [19]:
```
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
bow = bow_vectorizer.fit_transform(df['clean_tweet'])
```

In [20]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(bow, df['label'], random_state=42, test_size=0.20)
```

In [21]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import f1_score, accuracy_score
```

In [24]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
pred=model.predict(x_test)
```

In [25]:
```
accuracy_score(y_test,pred)
```

Out[25]:
```
0.9441576724542469
```

Fig23: implementation 9

In [26]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,pred)
```

In [27]:
```
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Positive','Negative'],
            yticklabels=['Positive','Negative'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```

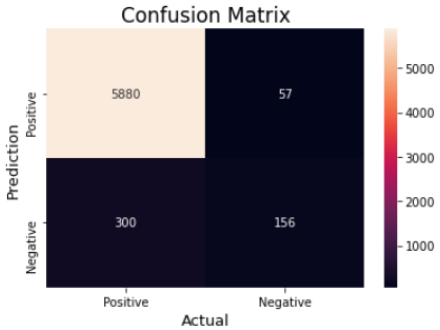


Fig24: implementation 10

In [28]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

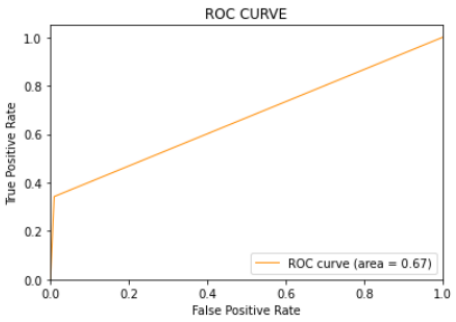


Fig25: implementation 11

In [29]:
```
from sklearn.svm import SVC
model1 = SVC()
```

In [30]:
```
model1.fit(x_train, y_train)
```

Out[30]:
```
SVC()
```

In [31]:
```
pred = model1.predict(x_test)
```

In [32]:
```
from sklearn.metrics import accuracy_score

print('Accuracy:',accuracy_score(y_test,pred))
```

```
Accuracy: 0.9450961989676209
```

In [33]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,pred)
```

Fig26: implementation 12

In [34]:

```
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Positive','Negative'],
            yticklabels=['Positive','Negative'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```

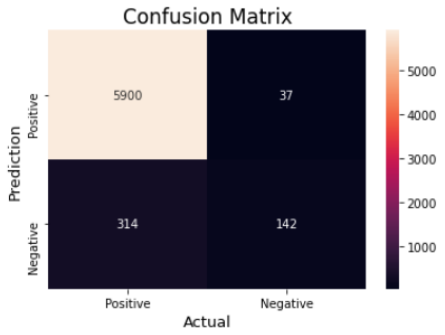


Fig27: implementation 13

In [35]:

```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, pred)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

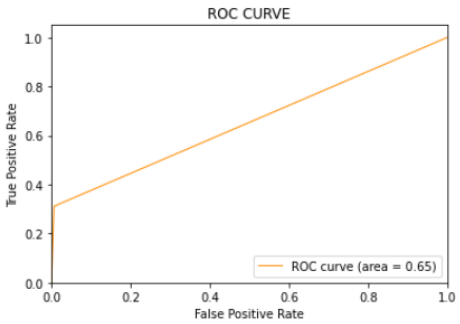


Fig28: implementation 14

```
In [36]: from sklearn.naive_bayes import MultinomialNB
         model3 =MultinomialNB()

In [37]: model3.fit(x_train, y_train)

Out[37]: MultinomialNB()

In [38]: y_pred3 = model3.predict(x_test)
         y_pred3

Out[38]: array([0, 0, 1, ..., 0, 0, 0], dtype=int64)

In [39]: print('Accuracy:',accuracy_score(y_pred3,y_test))

         Accuracy: 0.9369623025183795

In [40]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,y_pred3)
```

Fig29: implementation 15

```
In [41]: sns.heatmap(cm,
                     annot=True,
                     fmt='g',
                     xticklabels=['Positive','Negative'],
                     yticklabels=['Positive','Negative'])
         plt.ylabel('Prediction',fontsize=13)
         plt.xlabel('Actual',fontsize=13)
         plt.title('Confusion Matrix',fontsize=17)
         plt.show()
```

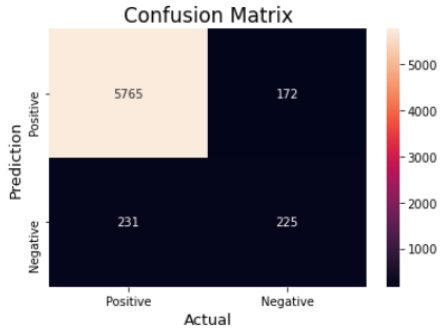


Fig30: implementation 16

In [42]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred3)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

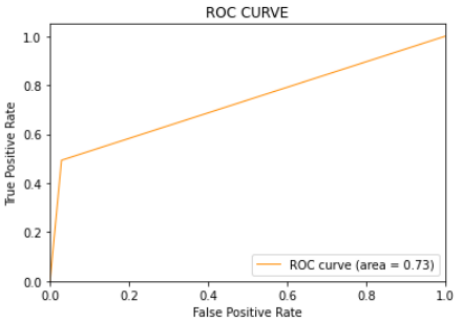


Fig31: implementation 17

In [43]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [44]:
```
rf_classifier = RandomForestClassifier(n_estimators=100)
```

In [45]:
```
rf_classifier.fit(x_train, y_train)
```

Out[45]:
```
RandomForestClassifier()
```

In [46]:
```
y_pred4= rf_classifier.predict(x_test)
```

In [47]:
```
accuracy = accuracy_score(y_test, y_pred4)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9413420929141249
```

In [48]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,y_pred4)
```

Fig32: implementation 18

In [49]:
```
sns.heatmap(cm,
            annot=True,
            fmt='g',
            xticklabels=['Positive','Negative'],
            yticklabels=['Positive','Negative'])
plt.ylabel('Prediction',fontsize=13)
plt.xlabel('Actual',fontsize=13)
plt.title('Confusion Matrix',fontsize=17)
plt.show()
```

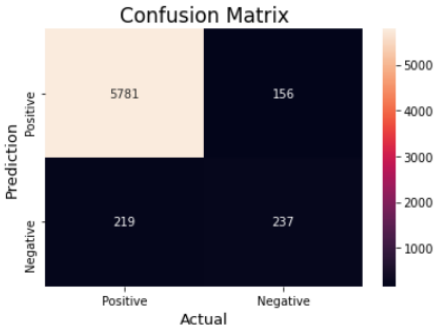


Fig33: implementation 19

In [50]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred4)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

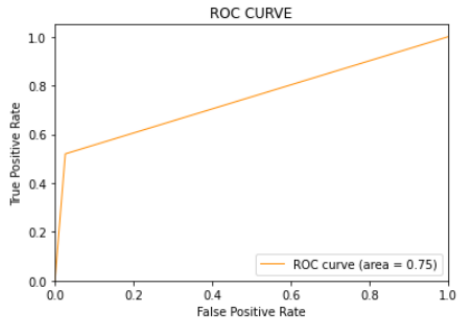


Fig34: implementation 20

```
In [51]: from sklearn.neighbors import KNeighborsClassifier

In [52]: knn_classifier = KNeighborsClassifier(n_neighbors=5)

In [53]: knn_classifier.fit(x_train, y_train)

Out[53]: KNeighborsClassifier()

In [54]: y_pred5 = knn_classifier.predict(x_test)

In [55]: accuracy = accuracy_score(y_test, y_pred5)
         print("Accuracy:", accuracy)

         Accuracy: 0.9379008290317535

In [56]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,y_pred5)
```

Fig35: implementation 21

```
In [57]: sns.heatmap(cm,
                    annot=True,
                    fmt='g',
                    xticklabels=['Positive','Negative'],
                    yticklabels=['Positive','Negative'])
         plt.ylabel('Prediction',fontsize=13)
         plt.xlabel('Actual',fontsize=13)
         plt.title('Confusion Matrix',fontsize=17)
         plt.show()
```

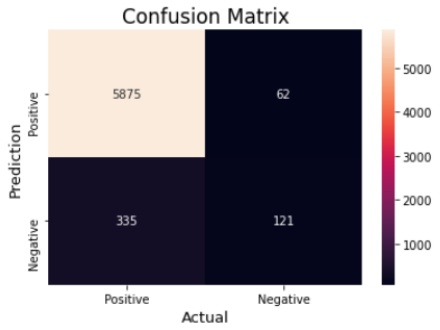


Fig36: implementation 22

In [58]:
```
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_pred5)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC CURVE')
plt.legend(loc="lower right")
plt.show()
```

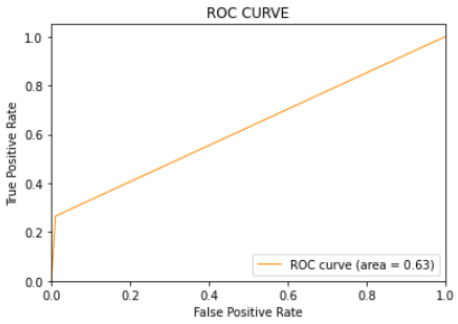


Fig37: implementation 23

In [59]:
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

In [60]:
```
lda_classifier = LinearDiscriminantAnalysis()
```

In [61]:
```
x_train.todense()
```

Out[61]:
```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [62]:
```
X_train = x_train.toarray()
```

In [63]:
```
x_train.todense()
```

Out[63]:
```
matrix([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [64]:
```
lda_classifier.fit(X_train, y_train)
```

Out[64]:
```
LinearDiscriminantAnalysis()
```

In [65]:
```
y_pred7= lda_classifier.predict(x_test)
```

In [66]:
```
accuracy = accuracy_score(y_test, y_pred7)
```

Fig38: implementation 24

```
In [67]: accuracy
Out[67]: 0.9399343031440638

In [68]: from sklearn.metrics import confusion_matrix
         cm = confusion_matrix(y_test,y_pred7)

In [69]: sns.heatmap(cm,
                     annot=True,
                     fmt='g',
                     xticklabels=['Positive','Negative'],
                     yticklabels=['Positive','Negative'])
         plt.ylabel('Prediction',fontsize=13)
         plt.xlabel('Actual',fontsize=13)
         plt.title('Confusion Matrix',fontsize=17)
         plt.show()
```

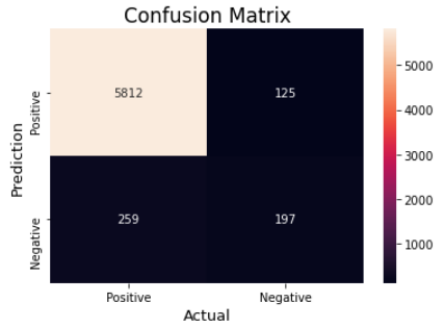


Fig39: implementation 25

```
In [70]: from sklearn.metrics import roc_curve, auc
         fpr, tpr, thresholds = roc_curve(y_test, y_pred7)
         roc_auc = auc(fpr, tpr)
         plt.figure()
         plt.plot(fpr, tpr, color='darkorange', lw=1, label='ROC curve (area = %0.2f)' % roc_auc)
         plt.xlim([0.0, 1.0])
         plt.ylim([0.0, 1.05])
         plt.xlabel('False Positive Rate')
         plt.ylabel('True Positive Rate')
         plt.title('ROC CURVE')
         plt.legend(loc="lower right")
         plt.show()
```

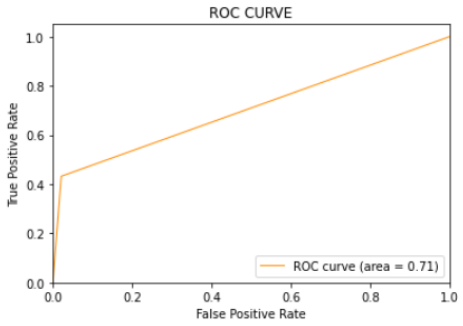
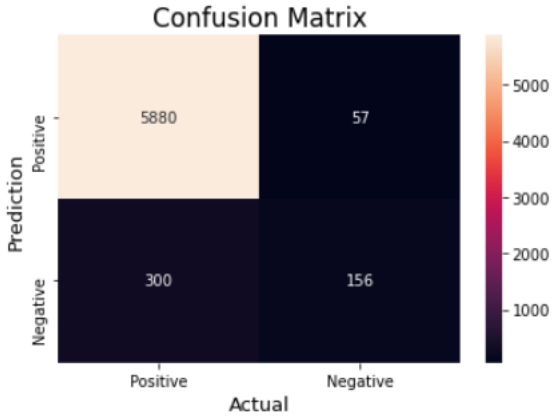


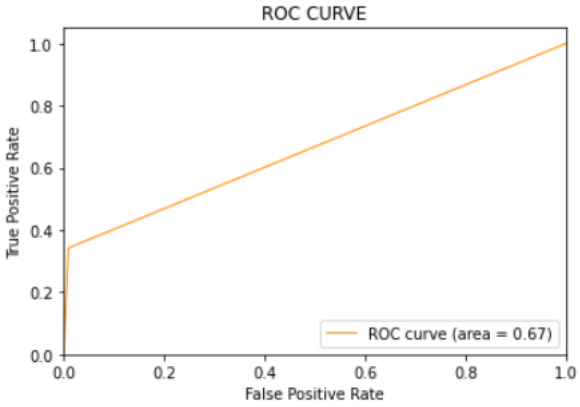
Fig40: implementation 26

Chapter – 5

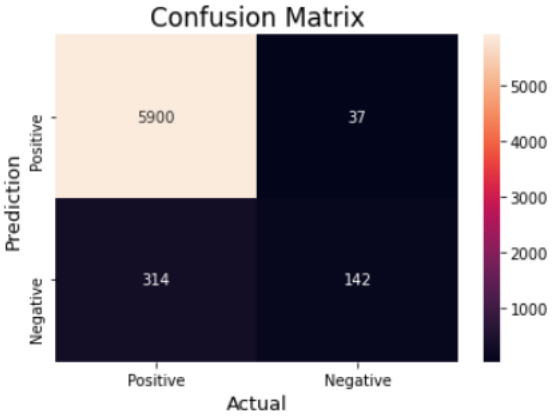
RESULT ANALYSIS AND VALIDATION

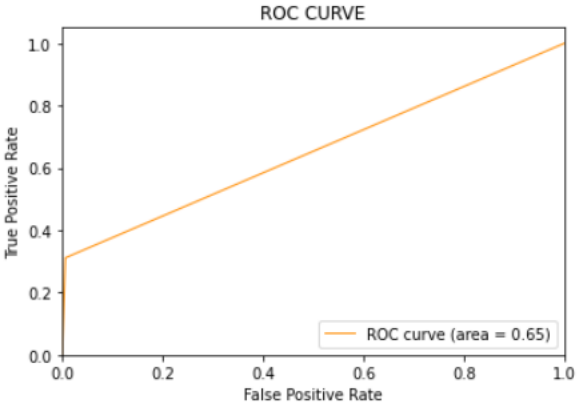
LOGISTIC REGRESSION



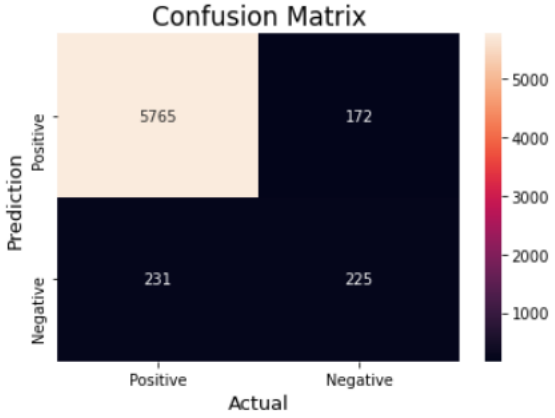


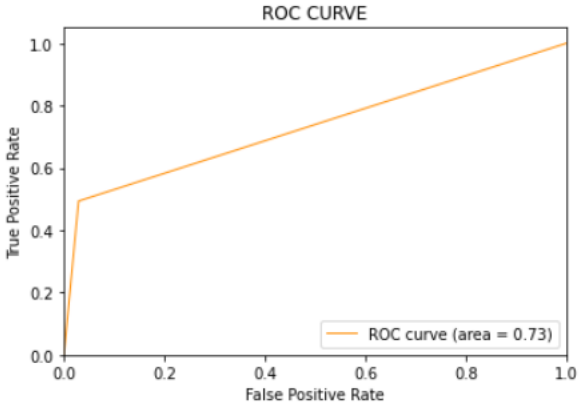
SUPPORT VECTOR MACHINE



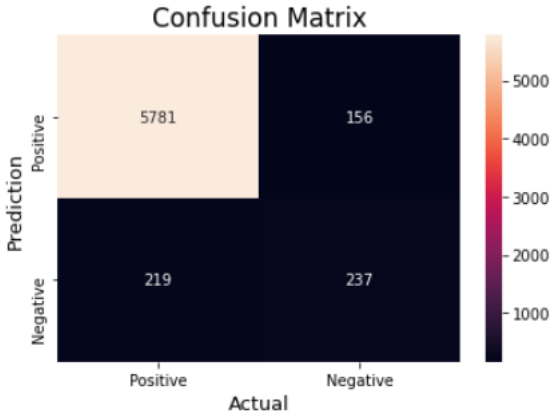


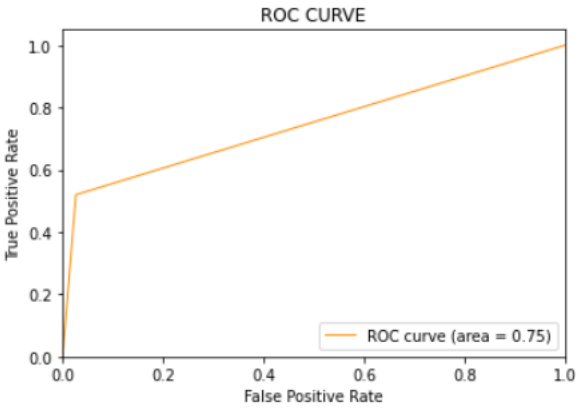
MULTINOMIAL NAÏVE BAYES



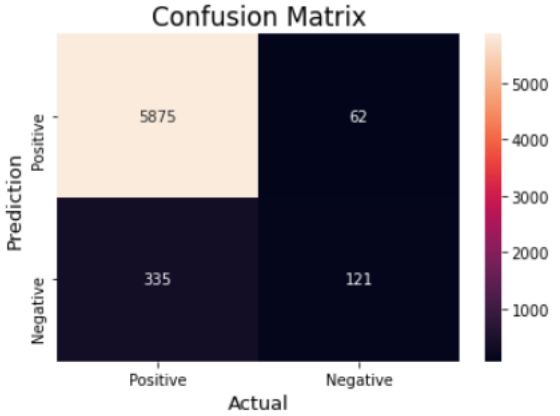


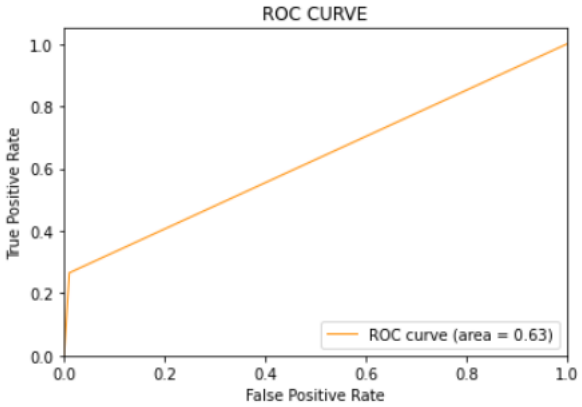
RANDOM FOREST



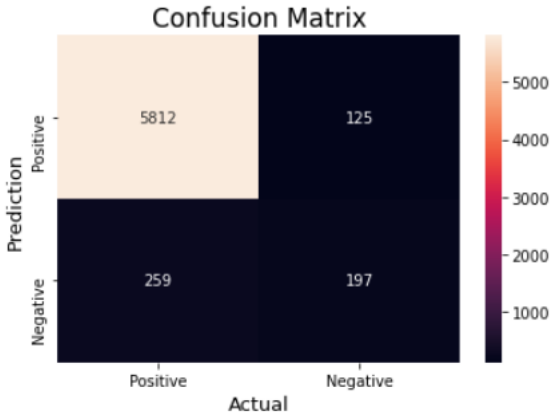


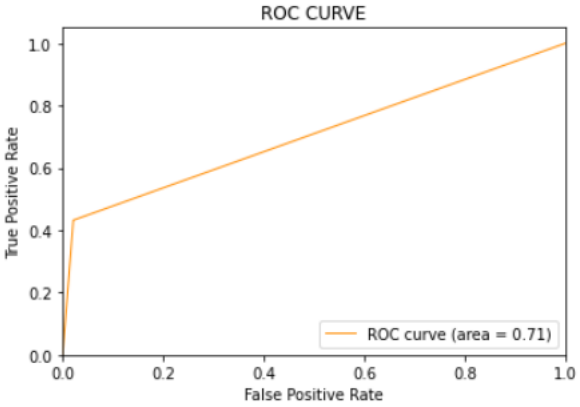
K NEAREST NEIGHBOR





LINEAR DISCRIMINANT ANALYSIS





Chapter – 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion:

The project presents the findings and outcomes of the study based on the data collected and analyzed. This section aims to provide a comprehensive understanding of the performance of the machine learning model employed for sentimental analysis and the insights derived from the analysis where the data is retrieved from twitter.
A dataset of tweets was analyzed in this study using machine learning algorithms to identify them as good, negative, or neutral based on the sentiment represented in the text. The model's performance was assessed using multiple measures such as accuracy, precision, recall, and F1-score.
The analysis results show that the machine learning model did well in identifying tweets based on sentiment.
Various accuracies are achieved by different machine learning models as follows:

1) Logistic Regression: 94.41
2) Support Vector Machine: 94.50
3) Multinomial Naïve Bayes: 93.69
4) Random Forest: 94.05
5) K Nearest Neighbor: 93.79
6) Linear Discriminant Analysis: 93.99

By above accuracies, though the difference is negligible, most suited classifier for the given data of tweets is Support Vector Machine.

6.2 Future Scope:

In the future this can be used or modified or enhanced for the purpose of extracting the twitter trends on a particular topic. With this model we can know how twitter people are reacting on a specific topic. This can be also helpful for further other classification techniques or deep learning or neural networks.

Chapter – 7

REFERENCES

[1] Pang, B., & Lee, L. (2023). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1-2), 1-135.

[2] Cambria, E., & Hussain, A. (2023). Sentic computing: Techniques, tools, and applications. Springer.

[3] Mohammad, S. M., & Turney, P. D. (2022). Crowdsourcing a word–emotion association lexicon. Computational Intelligence, 29(3), 436-465.

[4] Liu, B. (2022). Sentiment analysis and opinion mining. Morgan & Claypool Publishers.

[5] Thelwall, M., Buckley, K., & Paltoglou, G. (2022). Sentiment in Twitter events. Journal of the Association for Information Science and Technology, 63(1), 119-132.

[6] Kouloumpis, E., Wilson, T., & Moore, J. D. (2022). Twitter sentiment analysis: The good the bad and the OMG!. ICWSM, 11(538-541), 164-167.

[7] Pak, A., & Paroubek, P. (2021). Twitter as a corpus for sentiment analysis and opinion mining. Proceedings of the 7th conference on International Language Resources and Evaluation (LREC'10).

[8] Zhang, L., & Wu, D. (2021). A comprehensive survey on sentiment analysis of social media. Advances in Mechanical Engineering, 11(8), 1687814019867565.

[9] Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2021). Sentiment analysis of Twitter data. Proceedings of the Workshop on Languages in Social Media, 30-38.

[10] Medhat, W., Hassan, A., & Korashy, H. (2021). Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal, 5(4), 1093-1113.

[11] Li, Y., Sun, Z., Liu, F., Li, Q., & Li, H. (2020). A survey of opinion mining and sentiment analysis. Big Data Mining and Analytics, 1(4), 274-309.

[12] Chen, Y., Zhang, X., & Li, X. (2020). A survey on sentiment analysis using machine learning techniques. Journal of Intelligent & Fuzzy Systems, 35(3), 2775-2789.

[13] Baccianella, S., Esuli, A., & Sebastiani, F. (2019). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. LREC, 10(2200-2204), 2200-2204.

[14] Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2019). Lexicon-based methods for sentiment analysis. Computational linguistics, 37(2), 267-307.

[15] Hutto, C. J., & Gilbert, E. (2019). VADER: A parsimonious rule-based model for sentiment analysis of social media text. Eighth International AAAI Conference on Weblogs and Social Media.